Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

Laura Silva de Assis
Cientista da Computação – FACICOMP - FIC

Problema de Reagrupamento Capacitado

Dissertação de mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Automação.

Banca Examinadora:

Prof. Dr. Paulo Morelato França (Orientador)
FEEC - Unicamp

Prof. Dr. Christiano Lyra Filho
FEEC - Unicamp

Prof. Dr. Armando Zeferino Milioni
CTA - ITA - IEMB - São Jose dos Campos

Campinas
Março 2009

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

As76p

Assis, Laura Silva de
Problema de reagrupamento capacitado/ Laura Silva de Assis. – Campinas, SP:[s.n.],2009.

Orientador: Paulo Morelato França.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Otimização Combinatória. 2. Heurística. 3. Programação heurística. I. França, Paulo Morelato. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título

Título em Inglês:	Redistricting capacitated problem
Palavras-chave em Inglês:	Combinatorial optimization, Heuristic, Heuristc programming
Área de concentração:	Automação
Titulação:	Mestre em Engenharia Elétrica
Banca Examinadora:	Armando Zeferino Milioni, Christiano Lyra Filho
Data da defesa:	27/03/2009
Programa de Pós Graduação:	Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Laura Silva de Assis

Data da Defesa: 27 de março de 2009

Título da Tese: "Problema de Reagrupamento Capacitado"

Prof. Dr. Paulo Morelato França (Presidente): _____

Prof. Dr. Armando Zeferino Milioni: _____

Prof. Dr. Christiano Lyra Filho: _____

Com todo carinho...
- Aos meus amados pais.
- Aos meus queridos irmãos.

Agradecimentos

"É absolutamente impossível que um homem alcance qualquer realização sem levar outros consigo."

Agradeço de todo o coração...

A Deus, razão da minha vida, pela oportunidade, cuidado, proteção e conforto.

Aos meus pais Maria e Geraldo, pelo amor incondicional, pelo ensino e incentivo.

Aos meus irmãos, Márcio, Flávio e Keila, pelo carinho, preocupação, parceria e cumplicidade.

Aos meus sobrinhos Lili e Hecttor e ao filhinho do José (José Fernando), pois com um sorriso me proporcionaram bons momentos de alegria.

Ao pessoal do GEA pelas reuniões, amizade, principalmente a Eslô pelo carinho, e sua incrível capacidade de encontrar uma solução rápida para os problemas.

Ao meu orientador, professor Paulo França, pela orientação, paciência e incentivo durante o curso.

Ao pessoal da FEEC: Priscila, Rosana, Juliana, Tiago, Fabrício, Kemmi, Cecília pelo apoio e horas de estudos durante as disciplinas.

Ao pessoal do LABORE: Professor Christiano Lyra, Celso, Vinícius, José, Fábio, Hugo, Dani, Marcos, pela companhia, ajudas, sugestões e momentos de descontração compartilhados. De uma forma toda especial gostaria de agradecer ao Fábio uma pessoa muito paciênte e dedicada, pelo grande auxílio e ânimo em todos os momentos.

Ao Cris e ao Baca pela amizade incomparável, por inúmeras ajudas, críticas e sugestões, para vocês dois não encontro palavras de gratidão, levo-os sempre em meu coração.

À Ana Paula e Elma além das disciplinas cursadas juntas, agradeço pela convivência e companheirismo. Também ao Flávio Lúcio pelo socorro em momentos difíceis.

À Márcia, secretária do DENSIS, pelo carisma, paciência e auxílio em todos momentos que precisei.

Aos professores da banca: Professor Christiano Lyra e ao professor Armando Milioni pela atenção, tempo disponibilizado e sugestões.

À UNICAMP pela infra-estrutura.

À FAPESP, pelo apoio financeiro.

Resumo

O objetivo desta dissertação é desenvolver uma metodologia eficiente para solucionar o problema de agrupamento capacitado multicritério (PACM), no qual objetos com pesos associados são dados, os quais devem ser particionados em agrupamentos com capacidade limitada.

Neste trabalho, o PACM está ambientado em um problema de reagrupamento de lotes urbanos, nos quais devem ser realizadas as leituras dos medidores de energia elétrica por concessionárias de distribuição de energia. A operação de leitura dos medidores é realizada sobre lotes geograficamente definidos e é desempenhada sobre rotas percorridas uma vez por mês pelos leituristas.

A motivação deste trabalho é atribuída ao fato de que, com o passar do tempo, o tamanho e o formato dos lotes vão ficando obsoletos, devido a modificações introduzidas na conformação atual, desarranjando o equilíbrio entre os lotes e desatualizando as rotas. Por esse motivo é importante realizar um reagrupamento dos lotes buscando a diminuição dos custos operacionais de leitura, assim como a minimização dos custos e transtornos causados pelas modificações.

O método proposto para resolver o problema abordado nesta dissertação é um algoritmo baseado na metaheurística GRASP (*Greedy randomized adaptive search procedure*). A eficiência do método proposto é testada sobre uma série de instâncias geradas e sobre uma rede real. Os experimentos computacionais demonstram a eficiência do método.

Palavras-chave: Otimização Combinatória, Metaheurística, Reagrupamento Capacitado.

Abstract

The aim of this dissertation is to develop an efficient methodology to solve the multicriteria redistricting capacitated problem (PACM), in which objects with associated weights are given, which must be partitioned into groups with limited capacity.

In this work, the PACM is inserted in to a reassignment problem of urban clusters of clients, in which the readings of the eletric energy measurement must be performed by the company of energy distribution. The reading operation is performed over lots geographically defined is performed once a month by the readers.

The motivation of this work is due to the fact that the size and shape of the lots become obsolete after some time, due to modifications introduced in the current conformation, desarranging the balance between the lots and outdating the routes. For this reason it is important to achieve a reassignment of the lots trying to decrease the operational costs of reading, as well as minimizing the costs and inconvenience caused by the changes.

The proposed method to solve the problem addressed in this dissertation is a algorithm based on GRASP (*Greedy randomized adaptive search procedure*) metaheuristic. The effectiveness of the proposed method is tested on a large number of generated instances and on a real network. Computational experiments demonstrate the effectiveness of the proposed approach.

Keywords: Combinatorial Optimization, Metaheuristic, Capacited Reassignment.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AMP	-	*Adaptive memory Programming*
$ANEEL$	-	Agência Nacional de Energia Elétrica
BL	-	Busca local
BN	-	*Best Neighbor*
FF	-	*First Found*
$FIFO$	-	*First in First out*
$GRASP$	-	*Greedy Randomized Adaptive Search Procedure*
RCL	-	Restricted Candidate List
PAC	-	Problema de Agrupamento Capacitado
$PACM$	-	Problema de Agrupamento Capacitado Multicritério
PD	-	Problema de Distritamento
PDP	-	Problema de Distritamento Político
POM	-	Problema de otimização multiobjetivo
SAT	-	*Satisfiability*

Sumário

Capítulo 1

Introdução

A otimização tem o objetivo de solucionar problemas nos mais diversos contextos e de determinar, de acordo com algum critério, a melhor alternativa dentro de um universo especificado.

A otimização combinatória é o estudo matemático do agrupamento, ordenação ou seleção de um número finito de objetos discretos [Lawler, 2001]. Considerando certos critérios relacionados a um determinado problema, a maior parte dos problemas de otimização combinatória se preocupam em responder pelo menos uma de três perguntas sobre agrupamento:

1. O agrupamento existe?
2. Quantos agrupamentos existem?
3. Qual é o melhor agrupamento?

Entretanto, a existência de um tipo particular de agrupamento não é geralmente o ponto relevante, tão pouco a quantidade de agrupamentos existentes. A questão mais relevante e discutida em problemas para diversas aplicações é encontrar o agrupamento (arranjo, ordenação, seleção) ótimo de maneira eficaz dentre um número muito grande de possibilidades.

Problemas de otimização combinatória aparecem por toda parte, podendo ser aplicados em muitas áreas, tais como problemas de planejamento e programação (*scheduling*) da produção, problemas de corte e empacotamento, roteamento de veículos, redes de telecomunicação, sistemas de distribuição de energia elétrica, problemas de localização, dentre outras.

No contexto da otimização combinatória, os denominados problemas de distritamento (PD) (ou agrupamento) possuem o objetivo de realizar o particionamento de um conjunto de n unidades contíguas de uma região em p territórios, sendo que $p < n$, de forma que encontre a melhor solução possível considerando um ou mais critérios.

A aplicação mais comum de PD é o de distritamento eleitoral, onde estados ou municípios são divididos em distritos eleitorais compactos e com igual número de eleitores. Também

têm recebido certa importância aplicações voltadas a divisão de territórios em distritos de vendas, onde se deseja projetar distritos os mais homogêneos possíveis [França et al., 2007]. Outra característica freqüentemente encontrada em aplicações do problema de distritamento é o compromisso com a atual configuração dos distritos, fazendo com que ele seja designado como um problema de redistritamento.

Existem muitos problemas combinatórios próximos aos problemas de distritamento, tais como, os problemas de *bin packing* ou p-medianas capacitado. Em geral existem características suficientes que diferem estes problemas do PD, por exemplo, considerações geográficas ou geométricas como contiguidade entre os distritos ou compacidade.

O objeto de estudo desta dissertação é a otimização combinatória aplicada ao problema de agrupamento capacitado multicritério (PACM), tratado como um problema de distritamento hierárquico com dois objetivos.

O contexto no qual esta dissertação se insere corresponde à tarefa que concessionárias de distribuição de energia elétrica devem desempenhar mensalmente para medir o consumo de energia gasta nas unidades consumidoras de sua área de concessão. O consumo medido alimenta a fatura que é enviada a cada cliente mensalmente.

A preocupação deste trabalho se encontra nos clientes residenciais, pois estes formam um enorme contingente que requerem uma complexa e sistemática realização de tarefas, começando com uma repartição da área de concessão em unidades regionais que, por sua vez, são divididas nos chamados lotes (territórios) de faturamento.

Em cada lote de faturamento encontram-se definidas as rotas que são percorridas mensalmente por uma equipe de funcionários, os leituristas. A fixação do número de lotes por unidade regional tem como objetivo produzir um melhor aproveitamento logístico das equipes de leituristas, definido de modo a permitir que as equipes se ocupem integralmente de um território para cada dia útil do mês.

Realizar a divisão dos territórios dessa forma proporcionaria um cenário ideal de repartição da área de concessão, visando uma otimização logística da tarefa de leitura dos medidores. Entretanto, é necessário definir dentro de cada território, de forma adequada, a quantidade de rotas e o percurso de cada uma. Essas decisões também causam grandes impactos nos custos operacionais de leitura.

Um problema encontrado é que muitas concessionárias às vezes não dispõem de um plano racional de leitura. As que possuem um plano, sofrem um desequilíbrio entre os lotes e suas rotas que acontece devido ao crescimento vegetativo do mercado, de sua aglomeração, das transformações urbanas e da expansão do seu sistema elétrico.

A tarefa que se deseja cumprir, após a constatação da degradação das condições operacionais da atual configuração dos territórios, é a de redefinir os limites geográficos dos territórios procurando equilibrar suas cargas de leitura, porém, sem desprezar os seus formatos, favorecendo a construção *a posteriori* de rotas eficientes de leituristas.

Pela complexidade e tamanho do problema tratado, o projeto se limita aos problemas de (re)agrupamento da área de trabalho, sendo que um trabalho que obteve uma boa eficiência na definição das rotas dos leituristas dentro dos territórios foi o desenvolvido por [Usberti, 2007].

É importante ressaltar que esta redefinição é realizada sobre uma configuração de territórios pré-existentes e que essas alterações tendem a causar impactos que devem ser minimizados.

No processo de otimização é muito importante realizar uma boa escolha do método de resolução, o qual depende principalmente da razão entre a qualidade da solução gerada pelo método e o tempo gasto para encontrar esta solução. Grande parte dos problemas de agrupamento são intratáveis, ou seja, são problemas para os quais é improvável que se consiga desenvolver um algoritmo exato que possa ser executado em um tempo razoável. Nestes casos, para tornar viável a obtenção de uma boa solução é preciso utilizar métodos heurísticos. Tais métodos, quando bem escolhidos, desenvolvidos e adaptados, são capazes de apresentar soluções de boa qualidade em um tempo razoável.

Existem diversas metaheurísticas que possuem estratégias distintas, podendo-se destacar: Busca Tabu, Simulated Annealing, GRASP e as baseadas em Algoritmos Genéticos.

O objetivo principal deste projeto é desenvolver uma metodologia eficiente para solucionar o problema de agrupamento capacitado multicritério (PACM) citado. O problema analisado é o de reagrupamento de territórios, onde são realizadas as leituras dos medidores de energia elétrica de concessionárias de distribuição de energia. Foi provado em [Garey and Johnson, 1979] que este problema pertence à classe NP-difícil. Por esse motivo não existem algoritmos exatos que conseguem resolvê-lo para instâncias de grande porte. Por isso, essa dissertação devota o estudo a métodos heurísticos e propõe um algoritmo para resolver o PACM baseado na metaheurística GRASP.

Após o desenvolvimento e implementação do método proposto, a eficiência do mesmo será testada através de experimentos realizados e apresentados no Capítulo 5. O algoritmo também será executado para uma rede real obtida junto a uma concessionária de distribuição de energia elétrica do estado de São Paulo.

Este trabalho está organizado de forma que o Capítulo 2 apresenta uma breve síntese de alguns trabalhos relacionados que serviram como base para a dissertação, auxiliando na geração de idéias para o desenvolvimento da dissertação. O Capítulo 3 descreve de uma forma mais detalhada o problema tratado neste trabalho, apresentando uma formulação matemática para ele e explicando como foi representado. O Capítulo 6 apresenta as estruturas de dados utilizadas para representação computacional do problema. O Capítulo 4 aborda o método desenvolvido e utilizado para resolução do PACM hieráquico com dois critérios, discutindo cada parte da metodologia usada e apresentando alguns algoritmos. São apresentados alguns experimentos computacionais no Capítulo 5, onde é realizada a apresentação e análise dos resultados obtidos.

Capítulo 2

Trabalhos Relacionados

Diversos problemas de otimização possuem múltiplas funções objetivo que em sua maioria são conflitantes. Esse tipo de problema é conhecido como problema de otimização multiobjetivo (POM) e pode ser definido como o problema de encontrar um vetor de variáveis de decisão que satisfaça as restrições e otimize o vetor de funções objetivo.

Nos problemas de programação linear multiobjetivo existe um conjunto de soluções eficientes, também conhecidas como não-dominadas. Uma solução é eficiente quando não existe outra solução factível que forneça uma melhora em uma função objetivo sem provocar uma piora em pelo menos uma das demais funções objetivo [Lobianco and Meza, 2007].

Vários trabalhos têm sido desenvolvidos nessa área, especialmente pelo fato de se conseguir uma boa representação para problemas reais utilizando POM.

[Chinchuluun and Pardalos, 2007] realizaram uma investigação sobre otimização multiobjetivo, esboçando diversos resultados teóricos. Existem vários métodos para encontrar soluções eficientes em problemas multiobjetivos, sendo que os métodos mais comuns consistem em um processo que transforma o problema de otimização multiobjetivo em um problema com uma função escalar. Os autores apresentaram três métodos determinísticos:

- Método da ponderação;
- $\in$-restrição;
- L_p-métrico ponderado.

O método da ponderação foi proposto por Zadeh [Zadeh, 1963] e é amplamente utilizado para resolver problemas de otimização multiobjetivo. Neste método os problemas de otimização multiobjetivo são transformados em problemas de otimização escalar para os quais é definido um vetor de coeficientes de ponderação (pesos) que procuram representar a importância relativa de cada critério. Sendo assim, estes coeficientes de ponderação são usados para expressar uma única função objetivo, que é a combinação de todas funções objetivo do problema ponderadas por esses valores.

Considerando um problema de minimização da seguinte forma:

$$\begin{aligned} \min \quad & F(x) \\ S.a. \quad & x \in X. \end{aligned} \tag{2.1}$$

Onde $F(x) = (f_1, f_2, ..., f_k) : X \rightarrow \Re^k$. O novo problema, após aplicar o método da ponderação, pode ser representado da seguinte forma:

$$\begin{aligned} \min \quad & \sum_{i=1}^{k} w_i f_i(x), \\ S.a. \quad & x \in X. \end{aligned} \tag{2.2}$$

Sendo:

- X o vetor de variáveis de decisão;
- *F(x)* o vetor de funções objetivo;
- W o vetor de coeficientes de ponderação, com $\sum_{i=1}^{k} w_i = 1$.

O método $\in$-restrição consiste em selecionar uma função, do vetor de funções objetivo, como substituta para ser a função objetivo do problema. Em geral é escolhida a função que o decisor considera como mais importante, enquanto as demais funções objetivo são tratadas como restrições do problema, definindo para cada uma delas um limitante superior, ou seja valores máximos que o mesmo está disposto a aceitar (em problemas de minimização). O novo problema de otimização passa a ser formulado como:

$$\begin{aligned} \min \quad & f_j(x) \\ S.a. \quad & f_i(x) \leq \in_i, \qquad \forall i = 1, 2, ..., k, \quad i \neq j, \\ x \in X. & \end{aligned} \tag{2.3}$$

Sendo:

- X o vetor de variáveis de decisão;
- $\in$ o vetor de limitantes superiores (para problemas de minimização).

É importante mencionar que, de acordo com alguns autores, determinar valores para $\in_i$ não é uma tarefa trivial [Lobianco and Meza, 2007].

O método L_p-métrico ponderado consiste em escolher um ponto $y \in \Re^k$ e procurar uma solução ótima a qual se aproxime o máximo possível desse ponto. Uma métrica mede a distância entre dois pontos em $\Re^n$. Podem-se utilizar diferentes métricas normalmente denominadas como L_p, onde $p \in [1, \infty) \cup \{\infty\}$. Estas métricas também podem ser ponderadas a fim de encontrar diferentes soluções ótimas de Pareto. Dessa maneira o problema pode ser formulado como:

$$\begin{array}{lll} \min & \left(\sum_{i=1}^{k} w_i |f_i(x) - y_i|^p \right)^{1/p} & \\ S.a. & x \in X, \qquad\qquad w_i > 0 & \forall i = 1, 2, ..., k. \end{array} \tag{2.4}$$

Existem alguns problemas de otimização multiobjetivo ou com um único objetivo que são conhecidos como problemas de agrupamento. O problema que delineia este trabalho é um problema de agrupamento que foi tratado como um problema de distritamento. Por esse motivo são abordados alguns trabalhos correlatos que auxiliaram o desenvolvimento deste trabalho.

O agrupamento de pequenas unidades geográficas em grandes conjuntos territoriais obedecendo a alguns critérios é definido na literatura como distritamento ou *design* territorial [Ríos-Mercado and Fernández, 2007].

[França et al., 2007] apresentam uma comparação entre duas abordagens construtivas para resolver o problema de agrupamento capacitado multicritério (PACM), voltado à situação específica de redefinição de lotes urbanos de faturamento de empresas de energia elétrica. Foi proposto um modelo matemático de otimização restrito para resolver o PACM. O problema de agrupamento capacitado (PAC) é tratado nesse artigo como o problema das p-medianas capacitado, que consiste em encontrar um cliente (mediana) em cada um dos lotes, para o qual a soma de sua distância para todos os outros clientes do lote seja mínima.

A finalidade de [França et al., 2007] é gerar um conjunto inicial de soluções não-dominadas para o PACM. Para isso, foi apresentada uma estratégia que se divide em duas fases. Na primeira fase resolve-se um PAC monocritério em que são geradas diversas soluções minimizando apenas o critério de compacidade dos lotes, onde a forma geográfica dos lotes deve ser a mais compacta possível. Esse critério foi regido pelas seguintes restrições:

- Criação de exatos P territórios;
- Alocação de cada nó do grafo a um, e somente um, nó que seja mediana;
- Equalização dos tamanhos dos lotes com respeito à carga de leitura;

A segunda fase consiste em realizar uma avaliação da solução obtida na primeira com relação ao critério de conformidade dos lotes, buscando-se soluções não-dominadas que alteram o mínimo possível as associações consumidor-lote.

Foram testados dois algoritmos em [França et al., 2007] para resolver o problema proposto. O primeiro método implementado foi uma combinação de GRASP com *Adaptive Memory Programming* - AMP apresentado em [Ahmadi and Osman, 2005], sendo implementada apenas a parte construtiva, a qual gera sementes de medianas usando o conceito de densidade dos nós do grafo e faz alocações nó-mediana com base em uma função de arrependimento. Esse método também intercala fases de construção e desconstrução na busca por soluções cada vez melhores. O segundo método foi encontrado em [Osman and Christofides, 1994] e sua fase construtiva se baseia em três etapas:

1. Escolha da localização das medianas de cada agrupamento;
2. Designação dos nós às medianas;
3. Readequação de cada agrupamento para redefinir a nova posição da mediana.

Segundo [França et al., 2007], ambos os algoritmos obtiveram bons resultados. Embora tenha sido encontrado no primeiro algoritmo um conjunto aproximado de Pareto com valores melhores, o seu esforço computacional é maior que o do segundo. Problemas encontrados pelos autores nos dois métodos foram a dificuldade de obter soluções factíveis com baixo valor de desvio entre as cargas de leitura dos lotes e encontrar soluções sem a ocorrência de enclaves, já que se deseja sempre ter soluções contíguas.

[Laporte and Erkut, 2003] apresentam o problema multicritério de distritamento político e uma formulação matemática. Sua principal contribuição é a apresentação de um algoritmo de busca tabu com memória adaptativa. O algoritmo utiliza um método construtivo para obter uma solução inicial baseado em [Vickrey, 1961], o qual escolhe uma semente para iniciar um território e gradativamente o estende através da adição de unidades adjacentes. Se a solução encontrada possui um número maior de distritos que o número pré-determinado, então distritos que possuem tamanhos menores são unidos (*merge*). Em contra-partida, se foi obtido um número menor de distritos é feita uma divisão dos maiores distritos (*split*).

Após obter uma solução inicial, são utilizadas informações de vizinhança e informações armazenadas nas memórias, as quais são baseadas em recência e freqüência, para mover unidades de um distrito para outro. São apresentados dois tipos de movimentos: movimento tipo1, denotado por *(i,j,l)*, em que a unidade i do território j é movida para o território l e o movimento tipo2, denotado por *(i,k,j,l)*, em que a unidade i do território j é movida para o território l e a unidade k do território l é movida para o território j.

O algoritmo implementado por [Laporte and Erkut, 2003] é aplicado a um problema real referente à cidade de Edmonton, Canadá, o qual encontra melhorias significativas comparadas ao mapa de distritamento corrente desta cidade.

[Ricca and Simeone, 2008] também abordam o problema multicritério de distritamento político e apresentam quatro tipos de algoritmos de busca local (BL): *Descent*, *Tabu Search*

(exato e randômico), *Simulated Annealing* e *Old Bachelor Acceptance*. O principal objetivo é a comparação das diferentes técnicas de busca local tanto em termos de qualidade da solução como em tempo computacional.

Através da realização de alguns experimentos para avaliar o desempenho destes algoritmos os autores concluem que o *Old Bachelor Acceptance* é o algoritmo que produz os melhores resultados na maioria dos casos e que o *Descent* produz os piores, sendo que os demais conseguem, em média, obter bons resultados.

[Pereira et al., 2007] apresentam o problema de distritamento (PD) aplicado ao sistema de preço da rede de transporte público de Paris. Foi implementado um algoritmo híbrido, que corresponde a uma combinação de um algoritmo evolutivo com uma busca local, com o intuito de fazer uma aproximação da fronteira de Pareto. Os autores mostram que como o problema é multicritério, um algoritmo evolutivo é particularmente adequado, pois trabalha com um conjunto de soluções em potencial, as quais podem gerar diversas soluções eficientes em uma única iteração.

O objetivo do algoritmo proposto por [Pereira et al., 2007] é encontrar, em um tempo relativamente pequeno, um conjunto de soluções não-dominadas com alta qualidade. A cada iteração, depois de atribuir um valor de *fitness* para cada indivíduo, são aplicados os operadores de mutação e *crossover*, ao mesmo tempo em que uma lista de soluções não-dominadas é construída. Quando uma nova solução é inserida na lista, uma busca local é aplicada nesta lista com o intuito de melhorar a solução. A combinação com a BL permite verificar a existência de soluções eficientes localizadas perto das soluções obtidas.

[Ríos-Mercado and Fernández, 2007] apresentam uma metaheurística comumente utilizada em problemas de otimização combinatória, denominada GRASP, para resolver o problema de design territorial aplicado a uma empresa de distribuição de bebidas da cidade de Monterrey, México.

O GRASP consiste basicamente em duas fases: construção e pós-processamento. A primeira tenta construir uma solução factível e a segunda tenta melhorá-la. Quando uma solução factível é encontrada com sucesso na primeira fase, o que nem sempre ocorre, a segunda fase consiste em uma busca local com o objetivo de melhorar o valor da função objetivo. Porém quando a solução inicial encontrada na fase construtiva não é factível, executa-se uma fase de ajustamento, a qual, junto com a busca local, procura factibilizar e melhorar a solução corrente.

Como a maior parte do tempo computacional gasto pelo algoritmo se dá na busca local, procurou-se utilizar um filtro para evitar que a busca local seja executada para soluções não promissoras, agilizando o algoritmo.

[Ríos-Mercado and Fernández, 2007] realizaram diversos experimentos em cada uma das fases do algoritmo, com diferentes instâncias de diferentes densidades, e notou-se que a busca local produz melhorias significativas ($75\% - 90\%$) nas soluções obtidas na fase de construção. Quando comparado com soluções já utilizadas, fica claro que a qualidade das

soluções encontradas pelo algoritmo é melhor. O autor conclui que o método proposto é robusto e encontra soluções factíveis com um esforço computacional relativamente pequeno.

A maior parte dos artigos apresentados ajudou na formação de idéias a fim de compreender e resolver o PACM proposto neste projeto. Entretanto a colaboração maior na compreensão do problema foi encontrada em [França et al., 2007] por tratar do mesmo problema, embora utilize métodos diferentes do que foi utilizado neste trabalho.

Para elaboração do método heurístico proposto, as principais idéias foram baseadas em [Ríos-Mercado and Fernández, 2007], por ser um trabalho que se identifica com o problema proposto e oferece um método eficiente para resolvê-lo.

Capítulo 3

Problema de Reagrupamento Capacitado Multicritério

3.1 Introdução

Na resolução de problemas de otimização, busca-se encontrar a melhor solução analizando um ou mais objetivos condicionados por algumas restrições. Estes problemas podem ser visualizados como modelos matemáticos de problemas de decisão. Encontrar a solução para problemas de decisão consiste em realizar a melhor escolha dentre um conjunto de alternativas, onde assume-se a existência de certos critérios segundo os quais a qualidade das alternativas é mensurada.

O problema de otimização apresentado neste trabalho é o PACM, para o qual se destina este capítulo. Também será abordado o ambiente em que o PACM foi aplicado, mostrando os tópicos mais relevantes para a compreensão do problema.

3.2 Problema de Agrupamento Capacitado Multicritério

Problemas de agrupamento consistem basicamente em determinar semelhanças existentes entre objetos que se deseja agrupar, formular a hipótese de agrupamento e definir o esquema de classificação [Sosa, 1996]. O problema de agrupamento capacitado (PAC) é definido a partir de um dado conjunto de clientes, onde cada um deles tem associados pesos ou demandas que devem ser agrupados em p grupos distintos, cada qual com uma capacidade conhecida, de modo que a soma total dos pesos dos objetos atribuídos ao agrupamento não deve ultrapassar a capacidade do agrupamento. O PAC deve atender às seguintes exigências:

- Cada cliente possui um peso;
- Todos os clientes devem ser atribuídos a um, e somente um, agrupamento;

- Os clientes devem ser particionados em p agrupamentos, sendo p um número fixo;
- A soma dos pesos dos clientes de um agrupamento não deve ultrapassar a capacidade desse agrupamento;
- Os agrupamentos devem ser formados de maneira que permaneçam contíguos.
- Deve-se definir um critério a ser otimizado que indique quais são as soluções boas, como por exemplo, o critério geográfico.

O PAC foi tratado neste trabalho como um problema semelhante ao problema de distritamento político (PDP) cujo principal objetivo é particionar uma região em distritos eleitorais, sujeito a algumas restrições. As restrições mais comuns para o PDP incluem: contigüidade, igualdade populacional, compacidade e homogeneidade sócio-econômica (renda média).

O PACM é uma extensão do PAC, sendo que o PACM possui mais de um critério (ou objetivo) que deve orientar o processo de otimização. O PACM também pode ser visto como um problema de partição onde, dada uma certa região com n unidades territoriais, deseja-se particionar essa região de forma a encontrar p territórios que atendam a alguns critérios, sendo $p < n$.

As principais aplicações do PACM têm sido voltadas para os problemas de distritamento eleitoral e para os problemas de divisão territorial de vendas. No primeiro problema, mais presente em países que adotam o voto distrital, o objetivo é dividir uma região eleitoral, como um estado ou um município, em distritos eleitorais que sejam neutros, ou seja, que não favoreçam nenhum partido ou candidato distrital devido à sua conformação geográfica. Em problemas de divisão territorial para vendedores, buscam-se regiões com características semelhantes entre si, de modo a não favorecer determinado vendedor ou estabelecimento [França et al., 2007].

3.3 Descrição do Problema

Esta subseção é voltada para a definição do tipo de problema de agrupamento sobre o qual se delineia o estudo deste trabalho, com ênfase em justificar porque se trata de um problema de agrupamento capacitado multicritério, quais são os critério relevantes e como eles foram tratados no problema.

O problema de agrupamento capacitado, foco deste trabalho, diz respeito à tarefa que concessionárias de distribuição de energia elétrica devem mensalmente desempenhar para medir o consumo de energia gasta nas unidades consumidoras de sua área de concessão.

Deve-se redefinir os limites geográficos dos lotes procurando equalizar suas cargas de leitura, porém sem desprezar os seus formatos, que devem favorecer a construção *a posteriori* de rotas eficientes de leituristas.

Ao se decidir pela redefinição do tamanho e da conformação dos lotes de cada unidade regional de uma concessionária, é necessário considerar alguns critérios:

1. **Critério de homogeneidade:** Os novos lotes devem ser os mais homogêneos possíveis quanto à carga de trabalho da equipe de leituristas, a fim de obter uma minimização dos custos operacionais de mão de obra.

2. **Critério de compacidade:** A forma geográfica dos novos lotes deve ser a mais compacta possível, para que a definição posterior de suas rotas de leitura seja a mais eficiente possível. É do senso-comum que formas de lote alongadas e tortuosas tendem a dificultar o traçado de boas rotas.

3. **Critério de conformidade:** Os novos lotes devem alterar o mínimo possível as atuais associações entre consumidores e suas datas de leitura.

Ao analizar o PACM proposto neste trabalho, os dois primeiros critérios mencionados acima se descortinam imediatamente. Ao realizar a redefinição da configuração dos lotes atuais considerando os dois primeiros critérios, pode-se obter uma solução muito diferente quanto à realocação de datas de leitura, o que não é interessante para a concessionária de energia elétrica, o que justifica a adoção do terceiro critério. Se o número de alterações for grande, o aumento dos custos financeiros provocados pela variação do fluxo de caixa, além de outros transtornos não mensuráveis, a solução proposta pode-se tornar pouco atrativa.

Para entender porque isso ocorre, é importante agregar algumas informações sobre o fornecimento de energia. A Agência Nacional de Energia Elétrica (ANEEL), por meio da Resolução 456, regulamenta as condições gerais de fornecimento de energia elétrica, visando aprimorar o relacionamento entre os agentes responsáveis pela prestação de serviço público de energia elétrica e os consumidores. Dentre diversos outros fatores estabelecidos pela ANEEL, é importante ressaltar os seguintes:

- Uma concessionária se obriga a fazer a leitura dos medidores de energia elétrica em intervalos de 30 dias, os quais podem variar entre 27 e 33 dias;

- A concessionária de energia elétrica necessita apresentar a fatura aos consumidores com um prazo mínimo de 5 dias para que seu pagamento seja efetuado;

- É facultado aos consumidores escolher o dia do pagamento entre 6 diferentes datas uniformemente distribuídas ao longo do mês.

Observando esse regulamento, efetuar grandes mudanças de realocação de datas de leitura pode prejudicar a solução final do problema. Por exemplo, o medidor de um consumidor era inicialmente lido todo dia 15 do mês, porém após a reconfiguração dos lotes esse medidor passou

a ser lido todo dia 5. Como esse intervalo fica fora do prazo de 27 à 33 dias entre leituras consecutivas, como determina a ANEEL, esse leitor deve ser comunicado dessa alteração com antecedência. Além disso, a antecipação da leitura implica em maior demora de seu faturamento, dado que sua data de pagamento não pode ser alterada por iniciativa da empresa. Esse aumento no prazo de faturamento causa uma perda financeira para a concessionária.

Assim, os raciocínios citados produzem um problema de (re)agrupamento capacitado multicritério de grande dimensão, já que em geral as aplicações mais atraentes situam-se em grandes cidades, com milhares de medidores a serem lidos mensalmente.

3.4 Representação do Problema

A representação do PACM apresentado neste trabalho é feita por um grafo conexo não-orientado *G(V, E)* onde V é o conjunto dos n nós e E é o conjunto das m arestas do grafo. A associação entre o grafo e a planta urbana da região que se deseja agrupar é obtida associando um nó a cada cruzamento da planta e uma aresta a cada segmento de rua entre dois cruzamentos. A Figura 3.1 mostra um grafo obtido por uma região de estudo.

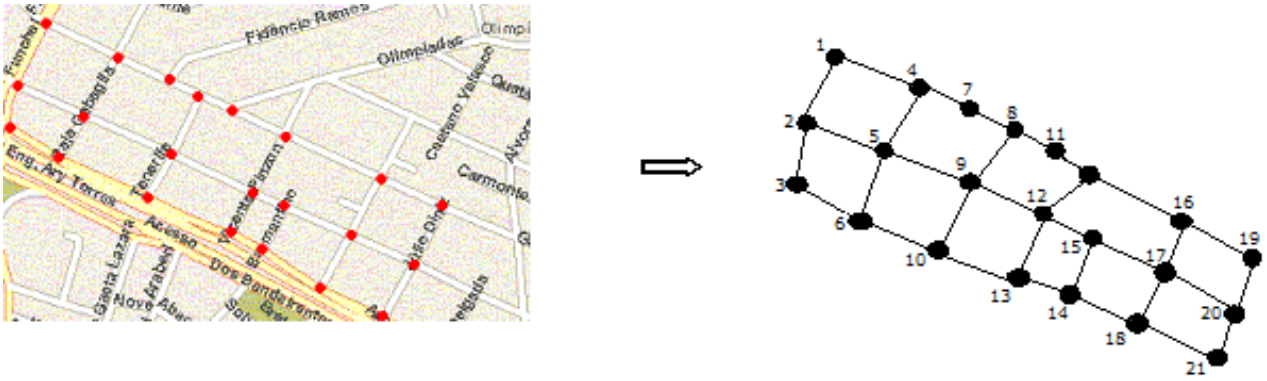


Figura 3.1: Associação entre a planta de uma região e o grafo.

Cada nó i do grafo possui alguns parâmetros associados, que em nosso problema são as coordenadas geográficas e duas atividades mensuráveis. Conhecendo as coordenadas (x, y) de cada nó é possível calcular a distância euclidiana entre cada par de nós. Essa distância é dada por:

$$d_{i,j} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{3.1}$$

Seja w_i^a o valor da atividade a no nó i, sendo que w_i^1 representa o número de medidores e w_i^2 o tempo de leitura desses medidores. Como na prática essas atividades estão relacionadas às arestas, pelo fato dos medidores se localizarem nos imóveis ao longo das ruas, é coerente

que cada w_i^a seja calculado como uma composição proporcional de cada atividade das arestas incidentes ao nó i. A Figura 3.2 mostra uma parte de um grafo com o número de medidores e tempos de leitura associados às aresta.

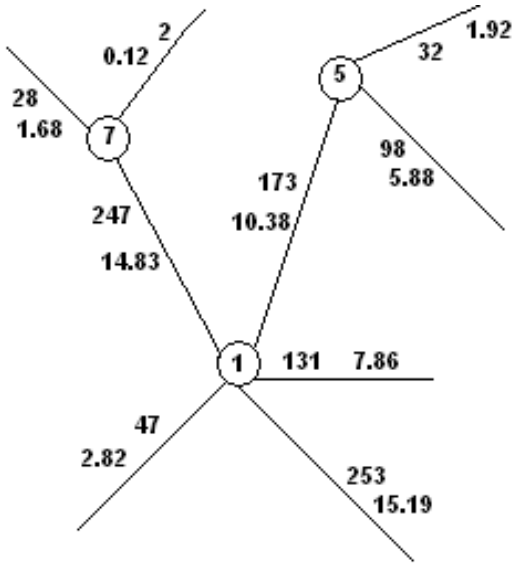


Figura 3.2: Atividades (1 e 2) associadas às arestas do grafo.

O valor das atividades para cada nó é calculado da seguinte forma: considerando o nó 1 da Figura 3.2, existem cinco arestas incidentes a este nó com número de medidores iguais a 131, 253, 47, 247, 173 e tempos de leitura iguais a 7,86; 15,19; 2,82; 14,83; 10,38. Como toda aresta do grafo é incidente a somente dois nós, divide-se o valor das atividades de cada aresta para os nós aos quais ela incide. Dessa forma o valor da atividade 1 para o nó 1 é igual a 425,5 e o valor da atividade 2 para o nó 1 é igual a 25,54, ou seja, esses valores são a soma da metade do valor de cada atividade referente às cinco arestas incidentes ao nó 1. O resultado do cálculo das atividades para o nó 1 é mostrado na Figura 3.3.

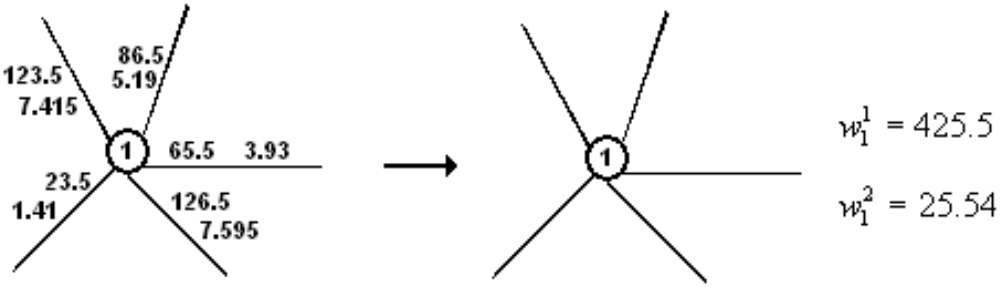


Figura 3.3: Cálculo do valor de cada atividade para um nó.

Um território é formado por um subconjunto V_k de nós onde $V_k \subset V$, sendo que a quantidade total de territórios que devem ser formados é dada por um parâmetro p. Ao se obter uma solução,

cada nó do grafo deve estar atribuído a um território. Dessa forma o conjunto de territórios formados por V_k, sendo $k = 1, ..., p$ definem uma partição de V.

Ao construir os territórios deseja-se também que eles estejam balanceados com respeito ao valor de cada atividade. Para isso define-se o tamanho do território V_k com relação à atividade a dado por:

$$w^a(V_k) = \sum_{i \in V_k} w_i^a, \qquad sendo\ a = 1, 2. \tag{3.2}$$

O ideal seria que a solução composta por p territórios estivesse com todos os territórios perfeitamente balanceados, porém na prática isso é quase impossível devido à estrutura discreta do problema e à restrição de atribuição exclusiva de cada nó. Por essa razão é medido o nível de atividade desejado para cada território, Equação (3.3), o qual não deve utrapassar uma margem de tolerância definida na Equação (3.4), ou seja, essa equação mostra que o tamanho de cada atividade para cada território, não deve utrapassar os limites superior e inferior definidos, sendo que o τ é o parâmetro de tolerância para as atividades.

$$\mu^a = w^a(V)/p \qquad sendo\ a = 1, 2. \tag{3.3}$$

$$(1 - \tau^a)\mu^a \ \leq w^a(V_k) \ \leq \ (1 + \tau^a)\mu^a \tag{3.4}$$

Um outro ponto importante que deve ser considerado na criação dos territórios é que cada território deve formar um subgrafo conexo, ou seja, deve existir um caminho entre cada par de nós, com todos os seus nós pertencentes unicamente ao conjunto de nós deste território.

Este problema pode então ser descrito como encontrar p partições de V que satisfaçam os três critérios mencionados na Seção 3.3.

3.5 Formulação Matemática

O objetivo do PACM tratado neste projeto é redefinir os limites geográficos dos lotes de faturamento de uma concessionária de energia elétrica de acordo com os critérios mencionados na Seção 3.3. Existem várias formas de formular e resolver o PACM. Nesta seção será mostrada uma formulação para o problema e no Capítulo 4 é mostrado um método para resolvê-lo.

Neste trabalho, o PACM foi tratado como um problema de (re)agrupamento capacitado hierárquico com duas funções objetivo. Na primeira etapa de implementação foi resolvido o problema monocritério (PAC), formados pelos critérios de compacidade e de homogeneidade, os quais foram fundidos em uma única função objetivo ponderados por um peso, atribuído pelo decisor, referente à importância do que se deseja priorizar no processo de otimização. Foi

utilizado o método de ponderação [Zadeh, 1963] a fim de transformar os dois critérios em um só. Esse novo objetivo foi denominado como critério geográfico, o qual busca encontrar territórios compactos e equilibrados com respeito à carga de trabalho. A função que representa o critério geográfico é detalhada no Capítulo 4.

É possível atender o critério geográfico minimizando a maior distância euclidiana entre um par de nós de um território e impedindo que as atividades dos nós associadas a cada território k ultrapassem as tolerâncias adimitidas para o tamanho médio da atividade a em cada território [Ríos-Mercado and Fernández, 2007], mantendo dessa maneira os territórios compactos e forçando os lotes a serem balanceados com relação às atividades associadas aos nós.

Após encontrar a melhor solução possível maximizando o critério geográfico, busca-se otimizar o critério de conformidade. Essa etapa da otimização não modifica a solução com respeito ao critério geográfico, por isso a definição do problema como um problema hierárquico com dois objetivos. A seguir são realizadas algumas definições de índices, parâmetros e variáveis [Ríos-Mercado and Fernández, 2007]:

- Índices e conjuntos:

  n Número de unidades territoriais (nós do grafo);
  p Número de territórios (partições do grafo);
  i,j Índice das unidades territoriais, sendo que $i, j \in V = \{1, \cdots, n\}$;
  a Índice das atividades, $a \in A = \{1, 2\}$;
  k e l Índices dos territórios, $k \in K = \{1, \cdots, p\}$ $l \in L = \{1, \cdots, p\}$;
  N^i $= \{j \in V | (i, j) \in E \vee (j, i) \in E\}$ conjunto de nós que são adjacentes ao nó i, ou seja, os vizinhos do nó i;

- Parâmetros:

  w_i^a Valor da atividade a no nó i, $i \in V$, $a \in A$;
  $d_{i,j}$ Distância Euclidiana entre o nó i e o nó j, $i, j \in V$;
  τ^a Tolerância relativa com respeito à atividade a, $a \in A$, $\tau^a \in [0, 1]$;
  $w^a(V_k)$ $\left(= \sum_{j \in V_k} w_j^a\right)$ tamanho do território V_k com relação à atividade a, $a \in A$;
  μ^a $\left(= \frac{w^a(V)}{p}\right)$ média alvo do valor da atividade a para cada território, $a \in A$;

- Variáveis de decisão para f_1:

$$x_{i,j} = \begin{cases} 1, & \text{se o nó } j \text{ está atribuído ao território com centro } i;\ i \in V; \\ 0, & \text{caso contrário.} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{se o centro de um território está localizado no nó } i; \\ 0, & \text{caso contrário.} \end{cases}$$

O modelo matemático para o PAC pode ser escrito como:

$$min \quad f_1(S) = \lambda F(S) + (1-\lambda)G(S) \tag{3.5}$$

$$S.a. \quad \sum_{i \in V} x_{ij} = 1 \qquad j \in V \tag{3.6}$$

$$\sum_{i \in V} y_i = p \tag{3.7}$$

$$x_{ij} \leq y_i \qquad \forall \;\; i,j \in V \tag{3.8}$$

$$\sum_{j \in \cup_{v \in D} N^v \setminus D} x_{ij} - \sum_{j \in D} x_{ij} \geq 1 - |D| \qquad i \in V; D \subset V \setminus (N^i \cup \{i\}) \tag{3.9}$$

$$x_{ij}, \quad y_i \quad \in \{0,1\} \qquad i,j \in V \tag{3.10}$$

$$F(S) = \left(\frac{1}{d_{max}}\right) \left(\max_{k=1,\cdots,p} \left\{ \max_{i,j \in V_k} \; d_{ij} \right\} \right) \tag{3.11}$$

$$g^a(V_k) = \left(\frac{1}{\mu^a}\right) max\{w^a(V_k) - (1+\tau^a)\mu^a, \; (1-\tau^a)\mu^a - w^a(V_k), \; 0\}$$

$$G(S) = \sum_{k=1}^{p} \sum_{a \in A} g^a(V_k) \tag{3.12}$$

Sendo:

- F(S) é a medida de dispersão do território baseada na distância euclidiana;
- d_{max} é a maior distância euclidiana entre dois nós do grafo;
- G(S) controla o balanceamento dos territórios com respeito a cada atividade, dando a soma das inviabilidades;
- λ é o fator de ponderação de F e G, tal que $\lambda \in [0,1]$;
- S é uma solução para o problema;
- N é um subconjunto de V definido no tópico índices e conjuntos.

A função objetivo exibida em (3.5) minimiza a combinação convexa da soma da maior distância euclidiana de cada território com as inviabilidades com respeito às atividades de cada território. Esses dois fatores são ponderados pelo parâmetro λ que é escolhido pelo decisor de acordo com a importância dada por ele às partes da função objetivo.

As restrições (3.6) definem que cada nó do grafo deve ser atribuído a um território. A restrição (3.7) obriga a formação de exatos p territórios. As restrições (3.8) obrigam que os nós sejam alocados somente às medianas. As restrições (3.9) garantem a conectividade dos territórios; essas restrições são similares às usadas em problemas de roteamento para garantir a conectividade das rotas, sendo que existe um número exponencial delas [Ríos-Mercado and Fernández, 2007]. As restrições (3.10) garantem que as variáveis de decisão sejam binárias e as (3.11) e (3.12) definem *F(S)* e *G(S)* respectivamente. É realizada uma descrição detalhada das funções *F(S)* e *G(S)* no Capítulo 4.

Uma atenção especial deve ser dada às restrições (3.9) já que são elas que garantem a conectividade dos territórios da solução. Devido ao fato que estas restrições são muito importantes e não são triviais, elas serão abordadas com mais detalhes.

As restrições de conectividade possuem a seguinte idéia: ao se escolher um subconjunto de vértices D de uma solução, se todos os elementos deste subconjunto pertencerem ao mesmo território, algum outro nó desse território deve ser vizinho a D.

Para compreender essa idéia é necessário esmiuçar esta restrição. Pode-se dividí-la em 3 partes :

$$\underbrace{\sum_{j \in \cup_{v \in D} N^v \setminus D} x_{ij}}_{C} \; - \; \underbrace{\sum_{j \in D} x_{ij}}_{} \geq 1 - |D|_B \qquad \underbrace{i \in V; D \subset V \setminus (N^i \cup \{i\})}_{A}$$

Sendo que:

- Em A, D é um subconjunto de V, o qual não contém o nó i que está sendo analisado no momento nem seus vizinhos;
- Em B, calcula-se a quantidade de nós que pertencem a D e que estão alocados ao território com centro i;
- Em C, calcula-se a quantidade de nós adjacentes à D que pertençam ao território com centro i.

Após compreender cada parte da restrição é necessário analisar três situações para entender porque ela garante a conectividade dos territórios:

1) Quando i não é um centro: São analisados todos os nós do grafo ($i \in V$). Quando o nó i não for um centro a restrição sempre é satisfeita, porque B e C serão iguais a zero, logo $|D| \geq 1$;

2) Quando nem todos os elementos de D pertencem ao mesmo território: Neste caso $|D| > B \geq 0$ e $C \geq 0$, dessa forma tem-se que $C + |D| - B \geq 1$, logo a restrição também é sempre atendida;

3) Quando todos elementos de D fazem parte do mesmo território: Neste caso a propriedade da restrição deve ser atendida, pois se não existir algum vizinho de D pertencente ao território com centro i ($C = 0$) a condição falha, uma vez que ao escolher o conjunto D é desconsiderado o nó i analisado no momento e seus vizinhos.

Como são analisados todos os nós do grafo, dependendo do subconjunto D escolhido, mesmo existindo territórios desconexos na solução, as Restrições (3.9) são satisfeitas, mas sempre haverá no mínimo um caso em que ela falhará. A figura 3.4 ilustra esta situação.

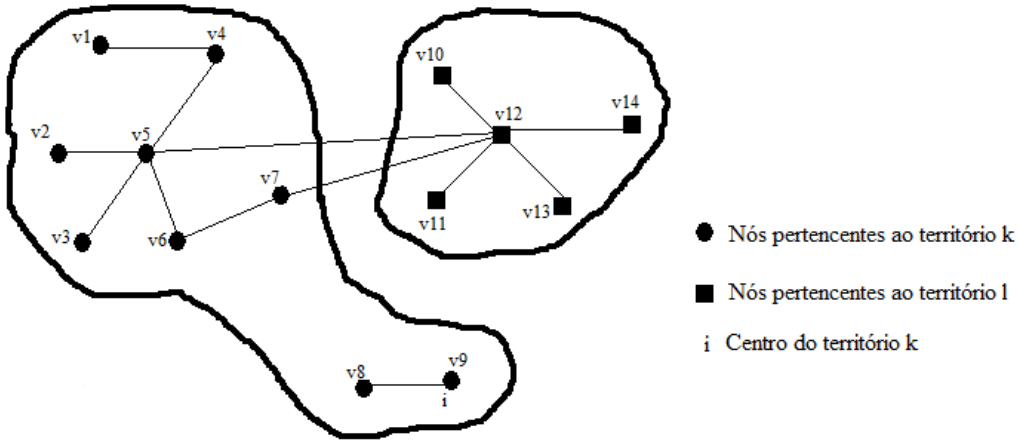


Figura 3.4: Restrição de conectividade, situação 3.

Observando a solução mostrada na Figura 3.4 nota-se que o território k é desconexo. Considerando o subconjunto $D_1 = \{v_1, v_2, v_3, v_4, v_5\}$ a restrição de conectividade não falha, porque existe um nó pertencente a k (v_6) que é vizinho de D_1. Porém como já mencionado anteriormente, sempre haverá um subconjunto D para o qual esta restrição irá falhar. Considerando então o subconjunto $D_2 = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ a restrição falha, pois não existe nenhum nó pertencente ao território k que é adjacente à D_2, sabendo que o nó i e seus vizinhos não fazem parte de D_2 e fazem parte do território k, conclui-se que o território k é desconexo.

Independente de qual nó tenha sido escolhido como centro (nó i da Figura 3.4), quando a solução possui algum território desconexo, sempre existe um subconjunto D para o qual a restrição de conectividade irá falhar. Isso ocorre porque os territórios desconexos são formados por ilhas, então sempre que for escolhido D formado por todos elementos de uma ilha e o centro localizado em outra ilha, a restrição irá falhar, pois não encontrará nenhum vizinho à D.

- Variáveis de decisão para f_2:

$$r_{k,l} = \begin{cases} 1, & \text{se o território } k \text{ passar a ter rótulo } l; \\ 0, & \text{caso contrário.} \end{cases}$$

- Matriz de custo:

$$c_{k,l} = \begin{cases} & \text{número de medidores do território } k \\ & \text{que originalmente estavam com o rótulo } l; \end{cases}$$

O segundo critério avaliado na resolução do problema é o critério de conformidade que pode ser formulado como se segue:

$$max \qquad f_2(S) = \sum_{k=1}^{p} \sum_{l=1}^{p} c_{kl} r_{kl} \tag{3.13}$$

$$S.a. \qquad \sum_{k=1}^{p} z_{kl} = 1 \qquad \forall\ l = \{1, \cdots, p\} \tag{3.14}$$

$$\sum_{l=1}^{p} z_{kl} = 1 \qquad \forall\ k = \{1, \cdots, p\} \tag{3.15}$$

$$z_{kl} \in \{0, 1\} \tag{3.16}$$

A função objetivo exibida em (4.5.1) maximiza o número de medidores que permanecem em seu território de origem. As restrições (3.14) definem que um rótulo deve ser alocado a somente um território, as restrições (3.15) determina que um território pode receber apenas um rótulo e as restrições (3.16) definem a variável z como binária.

A matriz de custo C é uma matriz de dimensão $p \times p$ onde suas linhas representam os territórios e suas colunas os rótulos. Cada posição da matriz guarda o valor do número de medidores do território k que estavam originalmente rotulados como l, ou seja, a posição k,l da matriz retorna o número de medidores que não irão mudar de rótulo caso o território k seja rotulado como l. A Figura 3.5 mostra um pequeno exemplo com um grafo e sua matriz de custo.

O modelo matemático formulado para o segundo critério representa exatamente uma formulação para o problema de designação (*assignment*) cuja formulação pode ser encontrada em [Wolsey, 1998], tornando desnecessário realizar a transformação do problema de rotulamento dos território para o problema de designação. Isso justifica a utilização do método Húngaro para resolver o segundo critério.

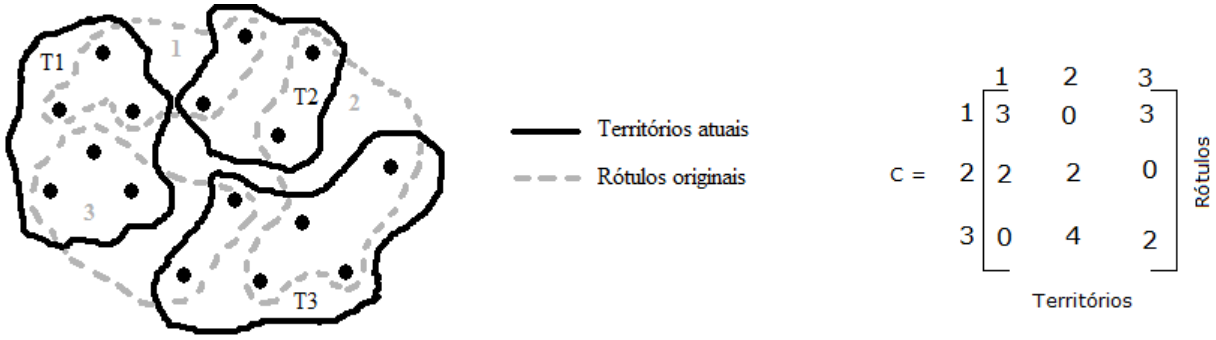


Figura 3.5: Representação de um grafo e sua matriz custo para o roteamento.

Capítulo 4

Heurísticas para o PAC

4.1 Introdução

Este capítulo tem o objetivo de descrever a implementação do algoritmo de solução do problema discutido nesta dissertação. Antes de pormenorizar a implementação propriamente dita, se faz necessário apresentar alguns conceitos sobre heurísticas e metaheurísticas a fim de facilitar a abordagem da metodologia de resolução do problema.

O PAC é um problema de otimização combinatória o qual foi provado pertencer a classe NP-difícil por [Garey and Johnson, 1979]. Devido sua complexidade, algoritmos exatos têm se mostrado incapazes de resolver instâncias de dimensões reais. Por isso a maior parte da literatura está voltada para métodos heurísticos e metaheurísticos.

As heurísticas e as metaheurísticas se apresentam como técnicas de otimização, nas quais as metaheurísticas podem ser consideradas uma evolução dos métodos heurísticos.

4.2 Heurísticas

As heurísticas foram desenvolvidos com a finalidade de resolver problemas que possuem elevado grau de complexidade em um tempo computacional razoável, encontrando para eles uma solução boa e factível [Chaves, 2003]. Para resolver problemas combinatórios difíceis, uma opção seria analisar todas as combinações possíveis a fim de encontrar a melhor delas. Se o problema for pequeno então essa é a melhor forma de se chegar à melhor solução. Porém, os problemas reais são de grande porte, gerando um número extenso de combinações, o que torna inviável encontrar a melhor solução analizando todas as combinações possíveis, dado que o tempo computacional exigido para realização dessa tarefa seria impraticável.

As heurísticas procuram encontrar soluções próximas da ótima em um tempo computacional razoável, porém não conseguem definir se a solução encontrada é ótima, ou quão próxima esta solução se encontra da ótima.

O princípio das heurísticas consiste em realizar um conjunto de buscas através do espaço de soluções do problema. O processo de busca inicializa em um ponto do espaço de soluções e termina quando o algoritmo encontra um ótimo local do problema. Uma boa escolha no processo de inicialização do algoritmo heurístico pode determinar a escolha de um ótimo local e aumentar a chance de se alcançar um ótimo global.

A grande desvantagem das heurísticas reside na dificuldade de fugir de ótimos locais, fato que deu origem às metaheurísticas, que orientam as heurísticas para sairem destes ótimos locais, permitindo a busca em regiões mais promissoras. O grande desafio é produzir, em tempo mínimo, soluções tão próximas quanto possíveis da solução ótima.

4.3 Metaheurísticas

As metaheurísticas surgiram da combinação de métodos heurísticos e apresentam-se como métodos que possuem uma estrutura melhor e com maior eficiência na exploração do espaço de busca que os métodos heurísticos. O termo metaheurística foi introduzido por Glover em 1986, derivado da composição de dois termos gregos: heurística que vem do verbo *heuriskein* que significa "para encontrar", e o sufixo meta que significa "além do nível superior". As metaheurísticas podem ser definidas também como heurísticas modernas [Christian and Andrea, 2003].

As metaheurísticas são procedimentos destinados a encontrar uma boa solução, eventualmente ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico. A principal característica das metaheurísticas é a capacidade que estas possuem de escapar de ótimos locais. As metaheurísticas utilizam diferentes conceitos que exploram um espaço de busca, usando estratégias de aprendizagem e informações para encontrar a solução ótima ou uma próxima da ótima.

O princípio fundamental das metaheurísticas consiste em visitar ou analisar um conjunto reduzido de soluções do espaço de busca, considerando que o espaço de busca é extremamente grande, o que torna inviável computacionalmente visitar todas as possíveis soluções do problema. O processo de busca então deve ser realizado de maneira eficiente e inteligente para que se visite um número menor de soluções e se obtenha soluções ótimas ou próximas da ótima em um tempo computacional aceitável [Christian and Andrea, 2003].

As metaheurísticas são algoritmos estruturados, que podem ser utilizados em diferentes problemas de otimização com poucas modificações para adaptar-se a um problema específico.

As principais características das metaheurísticas são:

- Guiam um processo de busca, explorando-o de maneira eficiente e encontrando uma solução ótima ou próxima da ótima;

- Possuem técnicas de busca que escapam de soluções locais e aprendem com esse processo;
- São aproximados e usualmente não determinísticos;
- São incorporados muitos mecanismos, para evitar o confinamento em determinadas partes do espaço de busca;
- Possuem conceitos básicos que permitem uma descrição nivelada e abstrata sem estar atrelado a um problema específico;
- Podem ser utilizadas em conhecimentos de domínios específicos, e em fórmulas heurísticas para controlar a estratégia em nível superior;
- Usam experiência de busca utilizando algum tipo de memória para guiar o processo de busca.

Existem diversas metaheurísticas utilizadas para resolver os mais variados problemas, as mais usadas são: os Algoritmos Genéticos, *Simulated Annealing*, Busca Tabu, GRASP.

O GRASP é uma metaheurística eficiente comumente aplicada a problemas de otimização combinatória, a qual foi escolhida para ser implementada neste trabalho. As subseções subsequentes abordam conceitos básicos a respeito de busca local (BL) e GRASP. Em 4.3.1, é apresentado uma introdução sobre busca local, sendo que a BL implementada neste trabalho é explicada na Seção 4.4.4. Em 4.3.2 são sintetizadas noções gerais a respeitos do GRASP.

4.3.1 Busca Local

Algoritmos de busca local são utilizados para melhorar soluções iniciais normalmente obtidas por alguma heurística construtiva. Esses algoritmos trabalham de forma iterativa, onde substituem sucessivamente a solução atual por uma solução melhor encontrada em sua vizinhança. O processo da busca termina quando nenhuma solução melhor é encontrada na vizinhança.

Em um algoritmo de busca local é definido para cada solução uma vizinhança que é composta por um conjunto de soluções alternativas. Sendo assim, a partir de uma solução atual, todas as soluções vizinhas podem ser encontradas pelo deslocamento de um nó [Resende and Ribeiro, 2008]. Simplificadamente um algoritmo de busca local pode ser descrito como: dada uma solução corrente, a BL percorre a vizinhança desta solução em busca de outra solução que seja melhor. Se tal solução vizinha for encontrada, esta torna-se a nova solução corrente. Se nenhuma solução melhor for encontrada, então a solução corrente está localizada em um ótimo local em relação à vizinhança estipulada.

A busca local não é útil apenas para melhorar o valor da função objetivo, mas pode contribuir também para o processo de deslocar-se de uma solução infactível para uma solução factível.

No problema de distritamento, o processo de melhoria da solução atual se faz modificando elementos (clientes) entre as regiões que formam a solução. Primeiramente verifica-se se é possível mover um determinado elemento de uma região para uma outra vizinha. Sendo possível, o movimento é realizado. Então verifica-se se esta nova solução é melhor que a atual, sendo melhor, a solução atual é substituída pela que foi encontrada após o movimento. A Figura 4.1 mostra uma solução corrente e suas regiões e o movimento de um elemento de uma região para uma região vizinha.

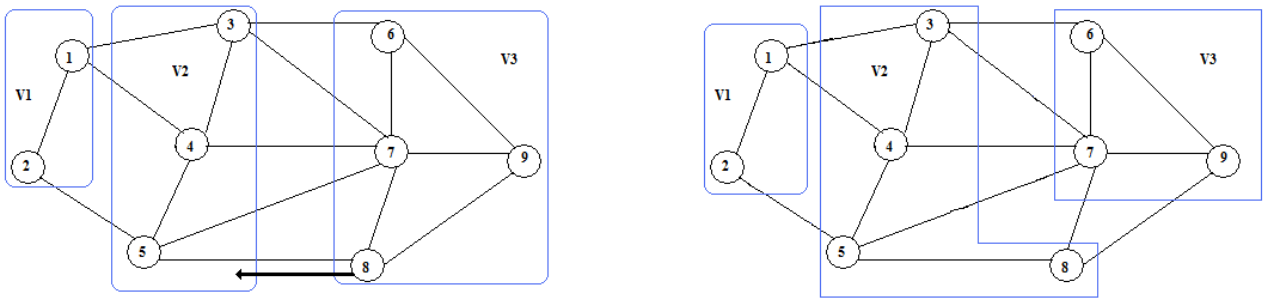


Figura 4.1: Obtenção de uma nova solução após o movimento de um elemento para uma região vizinha.

Algoritmos de busca local são considerados uma boa ferramenta para resolver PDP. Em geral quando o tamanho do problema é muito grande, procedimentos heurísticos de busca local raramente encontram a solução ótima, exceto em alguns casos quando a vizinhança é muito pequena [Ricca and Simeone, 2008].

A agilidade e eficiência da BL depende de diversos aspectos, tais como a estrutura da vizinhança, a técnica de busca utilizada, a estratégia utilizada para avaliar o custo em função do valor dos vizinhos e a solução inicial propriamente dita [Resende and Ribeiro, 2008]. A heurística construtiva é importante em relação a esse último aspecto, construíndo soluções iniciais de alta qualidade para a BL.

Existem duas estratégias comumente utilizadas a fim de realizar a busca na vizinhança, *best-improving* ou *first-improving*. No caso da estratégia *best-improving* todas as soluções vizinhas são verificadas e a solução corrente é substituída pela melhor solução vizinha. Na estratégia *first-improving*, para cada solução vizinha é verificado o valor da função objetivo e a solução atual é substituída pela primeira solução vizinha que possuir um valor melhor que a atual.

4.3.2 GRASP

O GRASP é uma metaheurística simples que combina uma heurística construtiva com uma busca local. Cada iteração é constituída basicamente de duas fases: fase construtiva e fase de busca local. Na primeira fase é construída uma solução, onde os elementos são selecionados

aleatoriamente e adicionados a um conjunto até que se obtenha uma solução factível. Na segunda fase, a vizinhança dessa solução é investigada até que se encontre a melhor solução local [Ríos-Mercado and Fernández, 2007]. A melhor solução dentre as encontradas é mantida como melhor solução encontrada para o problema.

O GRASP foi escolhido para resolver o problema que está sendo discutido neste trabalho por diversas razões, as principais são:

- É uma metaheurística eficiente, de fácil compreensão e implementação e que utiliza um esforço computacional pequeno;
- Dado que o problema possui uma aplicação real, a implementação de um algoritmo simples e eficiente auxilia a companhia em seu entendimento, possibilitando futuras aplicações com dados diferentes;

De um modo geral, pode-se dizer que o GRASP frequentemente fornece um bom compromisso entre a qualidade dos resultados obtidos e o esforço computacional requerido [Ríos-Mercado and Fernández, 2007].

4.4 Métodos e Algoritmos - Otimizando f_1

Esta Seção é dedicada à descrição do método utilizado para resolver o problema de reagrupamento capacitado que rege este trabalho. A metodologia utilizada foi a metaheurística GRASP, a qual será apresentada com detalhes nas seções seguintes. Na otimização com respeito ao critério de conformidade foram utilizadas duas técnicas, uma heurística e o método húngaro, que é um método exato, os quais são detalhados também neste capítulo.

4.4.1 Resolvendo o PACM pelo GRASP

A metaheurística GRASP é bastante conhecida por utilizar características boas tanto de algoritmos puramente gulosos como de procedimentos construtivos aleatórios.

Esse algoritmo consiste basicamente de duas fases: construção e pós-processamento. Na fase de construção o algoritmo constrói uma solução inicial enquanto que na fase de pós-processamento ele tenta melhorar essa solução. Quando a solução encontrada na primeira fase é factível, na segunda fase é realizada uma busca local adequada na vizinhança dessa solução, a fim de melhorar o valor de sua função objetivo.

Entretanto ao executar a fase de construção nem sempre é possível para o algoritmo conseguir encontrar uma solução viável, pois a solução construída pode não ter particionado o grafo em exatos p territórios, violando dessa forma as restrições mostradas em (3.7). Também nesta fase, a função *G(S)* exibida na equação (3.12) não é considerada por completo, apenas se utiliza o limitante superior a fim de evitar formações de territórios com muita carga de trabalho.

```
function GRASP(LimitIt, α, ρ, p)

Entrada: LimitIt := Número máximo de iterções do GRASP;
                 α := Parâmetro de qualidade da RCL;
                 ρ := Parâmetro de fechamento do terrritório;
                 p := Número de territórios.

Saída: A melhor solução factível encontrada - S^best.

1.   S^best ← {};
2.   for (l = 1, ··· , LimitIt) do
3.        S ← Construtivo(α, ρ);
4.        q ← |S|;
5.        if (q ≠ p) then
6.             S ← Ajustamento(S);
7.        S ← Pós − Processamento(S);
8.        if(S melhor que S^best) then
9.             S^best ← S;
10.  endfor;
11.  return S^best;
end GRASP.
```

Algoritmo 1 - GRASP

Por essa razão é necessário modificar a solução obtida na fase de construção de forma a torná-la factível. Para isso, a solução passa pela fase de ajustamento a fim de torná-la viável com respeito as restrições (3.7) e, posteriormente, a fase de pós-processamento que irá melhorar a qualidade da solução atual e reduzir as inviabilidades com respeito à (3.12), por meio de uma busca local.

O algoritmo 1 exibe o pseudo-código de uma implementação genérica do GRASP, ele recebe como entrada uma instância[1] para o PACM, o número máximo de iterações para o GRASP, o parâmetro de qualidade da lista de candidatos restrita (RCL), o número de territórios e o parâmetro de fechamento do território. O algoritmo 1 retorna a melhor solução encontrada.

GRASP possui dois parâmetros principais: um é relacionado com o critério de parada e o outro à qualidade dos elementos da lista restrita de candidatos. O critério de parada usado no algoritmo 1 determina o número máximo de iterações executadas pelo GRASP.

4.4.2 Fase de Construção

Para obter uma solução inicial, a fase de construção começa com a formação do primeiro território atribuindo um nó a ele. Os nós são alocados ao território corrente até atingir o valor limite da atividade ou até que não existam mais nós vizinhos para inserir neste território, assim o território corrente é fechado e um novo território é criado.

[1]Os dados contidos nas instâncias e sua disposição são apresentados no Capítulo 5.

Para iniciar um território a sugestão dada por [Ríos-Mercado and Fernández, 2007] é escolher o nó de menor grau a fim de favorecer a contigüidade, já que estes têm um número menor de vizinhos e são mais difíceis de serem alocados. Portanto se os nós de menor grau forem alocados primeiro, o risco destes nós sobrarem e formarem territórios isolados é minimizado. Para os nós sucessivos ao primeiro, é utilizada uma função gulosa que mede o valor da dispersão baseada na distância euclidiana e a violação das atividades. Para cada nó candidato v, temos a seguinte função gulosa:

$$\phi(v) \quad = \quad \lambda F_k(v) + (1-\lambda) G_k(v), \tag{4.1}$$

Sendo que:

$$F_k(v) \quad = \quad \left(\frac{1}{d_{\max}}\right) \max\left\{ f(V_k), \max_{j \in V_k} d_{vj} \right\}, \tag{4.2}$$

$$G_k(v) \quad = \quad \sum_{a \in A} g_k^a(v), \quad g_k^a(v) = \left(\frac{1}{\mu^a}\right) \max\left\{ w^a(V_k \cup \{v\}) - (1+\tau^a)\mu^a, 0 \right\} \tag{4.3}$$

Onde:

- $d_{\max} = \max_{ij \in V}\{d_{ij}\}$ Representa a maior distância euclidiana entre dois nós do grafo, que é usada para normalizar a função objetivo;
- $f(V_k) = \max_{ij \in V_k} d_{ij}$ É a maior distância euclidiana entre dois nós de um território;
- $g_k^a(v)$ Mede a violação da atividade a no território V_k;
- $w^a(V_k) = \sum_{i \in V_k} w_i^a$ Representa o tamanho de V_k com relação à atividade a;

O somatório de $g_k^a(v)$ representa a soma das inviabilidades referentes ao balanceamento da carga de trabalho dos territórios. No entanto, na equação (4.3), $g_k^a(v)$ representa a inviabilidade referente apenas ao limitante superior, ou seja, o quanto a carga de trabalho do território k ultrapassou o limite superior estipulado. Os dois fatores da equação (4.1) são ponderadas por um parâmetro $\lambda \in \{0, 1\}$; este valor é escolhido pelo decisor de acordo com a importância do que se deseja otimizar, entre compacidade e homogeneidade.

O algoritmo construtivo utiliza o parâmetro α para medir a qualidade dos nós candidatos e assim criar a RCL. Neste algoritmo é usado um valor fixo para α que determina quais nós pertencerão à lista a cada iteração. A escolha de quais nós vizinhos pertencerão à RCL é feita

através do cálculo de uma função que considera tanto a medida de dispersão quanto a violação referente ao balanceamento das atividades dos territórios, sendo que entrarão na RCL apenas os nós cuja função estiverem dentro de um intervalo definido também pelo parâmetro α, de forma que quanto maior o valor de α maior o intervalo considerado.

O valor de α é um dos parâmetros que devem ser ajustados para o GRASP. Os valores de α que levem a uma RCL de tamanho muito limitado, ou seja, utilização de valores para α próximos da escolha gulosa, implicam em soluções finais muito próximas às soluções obtidas de forma puramente gulosa, obtidas com um baixo esforço computacional, porém encontram uma baixa diversidade de soluções construídas. Em contrapartida escolher o valor de α próximo a escolhas puramente aleatórias leva a uma grande diversidade de soluções construídas, porém muitas dessas soluções são de qualidade inferior, tornando mais lento o processo de busca local [Chaves, 2003]. Por esses motivos é importante agregar esses dois aspectos para construção da solução.

Como a RCL normalmente possui mais de um elemento a cada iteração e a escolha de qual elemento irá entrar no território é feita de forma aleatória, podem-se obter diferentes soluções a cada execução da fase de construção.

O parâmetro ρ é usado para determinar o fechamento do território corrente e a abertura de um novo território. Ele é utilizado para calcular o limitante superior das atividades nos territórios, ou seja, usado para calcular o quão perto o valor de cada atividade do território está do desejado.

O algoritmo 2 mostra o pseudo-código da fase de construção. A heurística construtiva recebe como entrada os parâmetros α e ρ e retorna uma solução inicial para o problema.

Na linha 3 do algoritmo 2 o território é iniciado com o nó que possui o menor número de vizinhos, na linha 7 é computado o valor de $\phi(v)$ que é explicitado na equação (4.1), na linha 10 é criada a RCL utilizando o valor de α, onde são atribuídos a essa lista os nós vizinhos ao território V_q, os quais possuem o valor de ϕ dentro do intervalo estipulado. Na linha 14 é verificado se o critério de fechamento do território V_q é satisfeito, se for verdade, o território já está violando, no território corrente, o balanceamento da carga de trabalho, com respeito ao limitante superior. Então o território corrente é terminado e um novo território é iniciado.

O limiar de fechamento é ajustado pelo parâmetro $\rho > 0$, o qual permite maior flexibilidade nas fases seguintes. É notável que nesta etapa o algoritmo 2 não está preocupado com o limitante inferior porque a inserção de nós em um território não tem impacto negativo sobre o limitante inferior, situação que se altera a da busca local [Ríos-Mercado and Fernández, 2007].

4.4.3 Fase de Ajustamento

A fase de ajustamento é utilizada quando a solução construtiva viola a restrição (3.7). O que acontece é que a solução obtida na fase de construção pode não ser factível com respeito a

function Construtivo(α, ρ)

Entrada: α := Parâmetro de qualidade da RCL;
ρ := Parâmetro de fechamento do terrritório;

Saída: Uma solução construtiva - S.

1. $q \leftarrow 1$;
2. $\overline{V} \leftarrow V$;
3. $V_q \leftarrow \{v\}$, onde $v \in argmin\{|N^i| : i \in \overline{V}\}$;
4. $\overline{V} \leftarrow \overline{V} \backslash \{v\}$
5. *while* $(\overline{V} \neq \{\})$ *do*
6. $N(V_q) \leftarrow$ *conjunto de vizinhos de* V_q;
7. *compute* $\phi(v)\ \forall\ v \in N(V_q)$;
8. $\Phi_{min} \leftarrow min_v\{\Phi(v)\}$;
9. $\Phi_{max} \leftarrow max_v\{\Phi(v)\}$;
10. $RCL \leftarrow \{j \in N(V_q) : \phi(j) \in [\Phi_{min},\ \Phi_{min} + \alpha(\Phi_{max} - \Phi_{min})]\}$;
11. *Escolha* $v \in RCL$ aleatóriamente;
12. $V_q \leftarrow V_q \cup \{v\}$;
13. $\overline{V} \leftarrow \overline{V} \backslash \{v\}$;
14. $if((N(V_q) = \{\})\ ou\ (w^a(V_q) > \rho(1 + \tau^a)\mu^a)\ para\ algum\ a)\ then$
15. $q \leftarrow q + 1$;
16. $V_q \leftarrow \{v\},\ onde\ v \in argmin\{|N^i| : i \in \overline{V}\}$;
17. $\overline{V} \leftarrow \overline{V} \backslash \{v\}$;
18. $endif$;
19. *endwhile*;
20. *return* $S = \{V_1, \cdots, V_q\}$;

end Construtivo.

Algoritmo 2 - Heurística Construtiva

essa restrição, assim a fase de ajustamento é executada a fim de torná-la factível.

O número total de territórios encontrados na solução construtiva é denominado por q. Quando a solução viola a restrição (3.7) o que ocorre é que a solução não tem exatos p territórios, ou seja, ela tem $q < p$ ou $q > p$ territórios.

Quando a solução encontrada possui $q > p$ territórios, a fase de ajustamento realiza a operação de *merge*, onde é feita a união do território que possui o menor tamanho com o seu vizinho de menor tamanho. Essa operação reduz o número de territórios em 1 a cada iteração, então ela deve ser realizada até que $q = p$.

Quando a solução possui $q < p$ territórios, é realizada nesta fase a operação de *split*, que consiste em dividir o maior território em dois territórios conectados. Isso é feito aplicando-se recursivamente ao subgrafo do maior território o algoritmo construtivo com $p = 2$. Isso pode fazer com que sejam criados dois territórios ou mais, então essa operação deve ser feita até que $q = p$. Caso o *split* crie mais territórios que o desejado, a fase de ajustamento retorna ao início

function Ajustamento(S, q)

Entrada: S := Solução construtiva;
q := Número de territórios de S;

Saída: Uma solução factível referente a (3.7).

1. *while* $(q \neq p)$ *do*
2. $if(q > p)$ *then*
3. $V_{menorT} \leftarrow V_k,\ onde\ V_k \in argmin\{|S| : k \in \{1, \cdots, q\}\};$
4. $V_{menorTViz} \leftarrow V_l,\ onde\ V_l \in argmin\{|S| : l \in \{1, \cdots, q\},$
$i \in V_l,\ j \in V_k,\ (i,j) \in E\};$
5. $S \leftarrow S \backslash V_{menorT};$
6. $S \leftarrow S \backslash V_{menorTViz};$
7. $V_{menorT} \leftarrow V_{menorT} \cup V_{menorTViz};$
8. $S \leftarrow S \cup V_{menorT};$
9. $q \leftarrow q - 1;$
10. $else\ if(q < p)$ *then*
11. $V_{maiorT} \leftarrow V_k,\ onde\ V_k \in argmax\{|S| : k \in \{1, \cdots, q\}\};$
12. $V \leftarrow V_{maiorT};$
13. $\overline{S} \leftarrow construtivo(\alpha, \rho);$
14. $S \leftarrow S \backslash\ V_{maiorT};$
15. $S \leftarrow S \cup \overline{S};$
16. $q \leftarrow q + |\overline{S}|;$
17. $endif;$
18. *endwhile*;
19. *return* $S = \{V_1, \cdots, V_q\};$

end Ajustamento.

Algoritmo 3 - Fase de Ajustamento

e o *merge* é novamente executado, ou seja, a fase de ajustamento deve ser executada até que a solução tenha exatos p territórios.

Pode-se notar que a operação de *merge* pode ser feita de forma eficiente, porém a de *split* consiste em resolver um outro PAC. Entretanto a prática mostra que soluções obtidas na fase de construção raramente necessitam executar a operação de *split* [Ríos-Mercado and Fernández, 2007].

O algoritmo 3 mostra o pseudo-código da fase de ajustamento. Esse algoritmo recebe como entrada a solução construtiva e seu tamanho retornando uma solução factível para o problema.

Na linha 3 o algoritmo obtém o território de menor tamanho, na linha 4 é encontrado o território vizinho que possui menor tamanho. Esses dois territórios são fundidos em um único território, assim a cada *merge* realizado o número de territórios diminui em um. Na linha 11 o algoritmo encontra o território de maior tamanho. O subgrafo formado por esse território é passado recursivamente para o construtivo para que uma solução construtiva seja encontrada.

O objetivo é utilizar o construtivo, que já garante a conectividade, para dividir o subgrafo formado pelo maior território da solução em dois, porém, é provável que seja divido em mais regiões. Assim a fase de ajustamento realiza as operações de *merge* e *split* até que a solução tenha exatamente p territórios.

4.4.4 Fase de Pós-Processamento

Essa seção é dedicada à descrição da busca local que foi implementada para o PAC hierárquico com dois critérios. A fase de pós-processamento é destinada a modificar a solução atual com o objetivo de melhorá-la. Após realizar o ajustamento na solução construtiva e factibilizá-la, é executado na fase de pós-processamento uma busca local a fim de melhorar o valor da função objetivo desta solução. A função utilizada na busca local é similar a da fase de construção, porém nesta fase são considerados os dados de uma solução completa e o balanceamento das atividades é restringido por um limitante superior e outro inferior.

Para uma dada partição $S = \{V_1, \cdots, V_p\}$ o valor da função que orienta a busca é calculado da forma:

$$\Psi(S) \quad = \quad \lambda F(S) + (1 - \lambda)G(S), \tag{4.4}$$

Sendo que G(S) é dado na equação (3.12) e F(S) é determinado da seguinte forma:

$$F(S) = \left(\frac{1}{d_{max}}\right)\left(\sum_{k=1,\cdots,p}\left\{\max_{i,j\in V_k} d_{ij}\right\}\right)/p \tag{4.5}$$

É interessante notar que as equações (3.11) e (4.5) são bem semelhantes. A diferença entre as duas equações é que a segunda é utilizada no processo de minimização e a primeira, proposta por [Ríos-Mercado and Fernández, 2007], é usada no cálculo do valor final da função objetivo. A equação (3.11) calcula a distância euclidiana máxima de cada território e por fim retorna a distância máxima entre todos os territórios, normalizada pelo d_{max}, ou seja, retorna a medida de dispersão máxima do território. Inicialmente o algoritmo utilizava apenas a equação (3.11), porém notou-se durante a execução da busca local que ao escolher um bom movimento e executá-lo algumas vezes, o valor da função objetivo não se alterava. Isso ocorria quando o movimento reduzia a distância máxima de um determinado território, porém a distância máxima do grafo não se alterava. Por esse motivo decidiu-se utilizar a média do somatório das distâncias máximas, pois se a distância máxima de um determinado território melhora, o valor da função objetivo também melhora.

A Figura 4.2 ilustra a região onde se pode obter uma solução com $G(S) = 0$.

A inviabilidade das atividades em cada território é nula quando todos valores de $g(V_k)$ estiverem dentro do intervalo mostrado na Figura 4.2. Porém isso é muito difícil de ser alcançado, dependendo do valor assumido para τ.

Definição da Vizinhança

Para encontrar soluções vizinhas que sejam melhores que a construtiva corrente, é necessário realizar movimentos de transferência de nós entre os territórios que compõem a solução. Foi utilizado uma vizinhança $N(S)$ que é composta por todas as soluções que podem ser encontradas a partir de S, através do movimento de um nó i de seu território corrente k, para um território vizinho l que possua um nó j tal que $(i, j) \in E$, sem criar um território descontínuo. Esse movimento pode ser denotado por (i, k, l) e é ilustrado na Figura 4.3.

O movimento (i, k, l) é realizado somente se $V_l \cup \{i\}$ é conexo, o que ocorre sempre quando a aresta (i, j) existe, e quando $V_k \backslash \{i\}$ permanecer conexo.

Observando a Figura 4.3, um movimento factível é dado por (i_1, k, l), onde encontra-se uma nova solução possível na vizinhança e ambos territórios permanecem conexos. Porém quando se realiza o movimento (i_2, k, l) a unidade i_1 é desconectada de seu território de origem, por essa razão esse tipo de movimento não deve ser realizado. Neste caso i_2 é denominado como um nó de articulação.

Para encontrar um bom movimento é indispensável verificar se ele possui três características fundamentais:

1. O nó i que será movido deve ser um nó de fronteira;
2. O movimento deve melhorar o valor da função objetivo;
3. O nó i não pode ser um nó de articulação.

Supondo que deseja-se realizar o movimento (i, k, l), para atender a primeira característica é necessário verificar se o nó $i \in V_k$ é adjacente a algum $j \in V_l$. Para atender a terceira característica é necessário verificar se o nó i é um nó de articulação, ou seja, seu movimento não deve desconectar o grafo. Na Figura 4.4 o nó i é um nó de articulação, o qual sendo movido do território k para o l desconecta seu território de origem.

Uma busca em largura foi implementada para verificar quando o movimento candidato possui um nó de articulação. Foram implementadas duas buscas locais ambas baseadas em [Ríos-Mercado and Fernández, 2007], seguindo o princípio da técnica gulosa *first found* (FF). A busca em largura e as duas BLs serão expostas nas Seções seguintes.

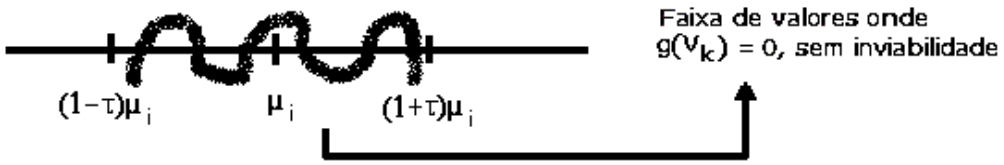


Figura 4.2: Intervalo onde a inviabilidade das atividades é igual a zero.

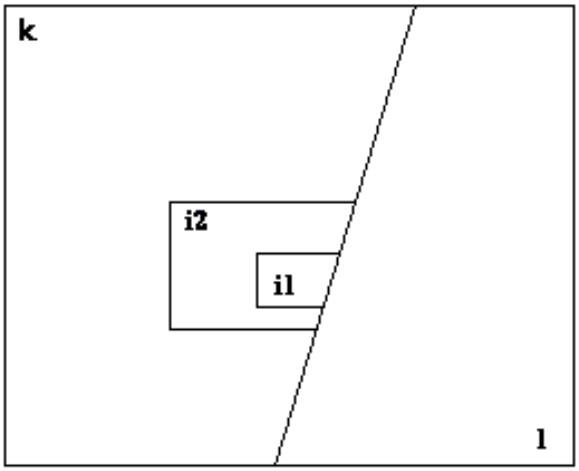


Figura 4.3: Exemplo de um movimento factível (i_1, k, l) e um infactível (i_2, k, l).

Figura 4.4: Nó de articulação - i.

Busca em Largura

O algoritmo de busca em largura[2] foi utilizado neste trabalho a fim de definir quando um nó candidato a um movimento é um nó de articulação, com o objetivo de identificar movimentos que desconectam o grafo.

A busca em largura é um dos algoritmos mais simples utilizados para realizar uma pesquisa em um grafo, é também o arquétipo de muitos algoritmos importantes como algoritmo de Dijikstra e de Prim [Cormen et al., 2002].

Dado um grafo $G = (V, E)$ e um nó de origem b, a busca em largura explora sistematicamente as arestas de G até encontrar cada nó acessível a partir de b. O algoritmo calcula a menor distância, entre b e todos os nós acessíveis a partir dele. A distância é calculada através da quantidade de arestas existentes entre dois nós. O algoritmo cria uma árvore em largura com raíz em b, que contém todos os nós acessíveis. Para qualquer nó v acessível a partir de b, o caminho nessa árvore do nó b até o v corresponde ao caminho mínimo entre esses dois nós em G [Cormen et al., 2002].

A busca em largura possui este nome porque expande uniformemente a fronteira entre nós descobertos e não descobertos, ou seja, todos os nós a uma distância de d arestas de b são descobertos antes de nós localizados a $d + 1$ arestas.

Pode-se dizer que esse algoritmo realiza uma busca exaustiva numa árvore inteira, sem considerar o seu alvo de busca, até que ele o encontre. Ele não utiliza uma heurística. Do ponto de vista do algoritmo, todos os nós filhos obtidos pela expansão de um nó são adicionados a uma fila (FIFO). Em implementações típicas, nós que ainda não foram examinados por seus vizinhos são guardados, por exemplo, em uma lista ligada, a qual é chamada de "aberta". Uma vez examinados, esses nós são colocados numa lista "fechada". A Figura 4.5 mostra um grafo e a ordem de exploração dos nós na busca em largura, tendo 1 como nó origem.

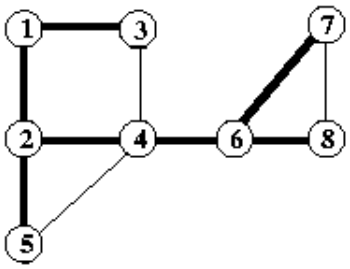


Figura 4.5: Ordem dos nós explorados na busca em largura.

Duas características são atribuídas a busca em largura [Cormen et al., 2002]:

[2]O pseudo código do algoritmo de busca em largura pode ser encontrado em [Cormen et al., 2002].

- A busca em largura é completada apenas se a árvore pesquisada possui um número finito de nós e se for realizada em um grafo conexo, assim o algoritmo consegue alcançar todos os nós do grafo;
- A busca em largura tem complexidade linear[3] $O(V + E)$.

Busca Local 1 (BL1)

Esta busca local executa o primeiro movimento factível encontrado, mesmo que este movimento não seja o melhor no momento, ou seja, aplica a estratégia *first improving*. O objetivo é escolher um bom movimento de forma aleatória, porém garantindo que todos os movimentos tenham a mesma probabilidade de serem escolhidos.

Como já mencionado, um bom movimento deve atender a três características específicas. A BL1 procura por um bom movimento, examinando se possuem as características necessárias na ordem que elas foram apresentadas. Decidiu-se agir dessa maneira dado o fato de que conhecendo quais nós são fronteira, grande parte de movimentos inviáveis são eliminados. Foi testado se o movimento melhora o valor da função objetivo antes de verificar se o nó é de articulação, pelo fato da busca em largura ser mais custosa computacionalmente e por ser mais raro de encontrar nós que são de articulação. Se a ordem fosse invertida provavelmente o algoritmo executaria a busca em largura para diversos nós os quais não melhorariam o valor da função objetivo. A seguir é mostrado como a BL1 escolhe o movimento factível, passando por estes três passos principais:

- **Garantindo característica 1:**

  Uma lista binária de nós de fronteira é construída, a qual guarda o valor 1 (um) quando o nó é fronteira e 0 (zero) se o nó não é fronteira. Para construir esta lista, a estrutura que guarda as informações do grafo é percorrida e para cada nó i o algoritmo obtém seu território e em seguida percorre sua lista de adjacentes. Quando o algoritmo encontra o primeiro adjacente que está localizado em um território diferente ao território do nó i, ambos, o nó i e esse nó adjacente são reconhecidos como nós de fronteira. Assim o algoritmo para de percorrer a lista de adjacência do nó i e passa para a lista do nó $i + 1$. Caso o algoritmo percorra toda a lista de adjacência do nó i e não encontre nenhum adjacente que pertença a um território diferente, então o nó i é reconhecido como um nó que não é fronteira.

  É importante ressaltar que a lista de adjacência de cada nó não é percorrida por inteiro, porque pode ter sido atribuído valor de fronteira a um nó por ele ser adjacente pertencente

[3]Prova da complexidade mostrada em [Cormen et al., 2002].

a outro território de um nó já verificado anteriormente. A lista de adjacência de um nó é percorrida em sua totalidade somente quando o nó que está sendo analisado no momento não for fronteira ou se somente o último nó da lista de adjacência pertencer a um território diferente.

Com o propósito de que todos os nós de fronteira tenham a mesma probabilidade de serem escolhidos como candidatos para um movimento, é criado um vetor de permutação (Vet_1) de 1 a r, sendo r o número de nós de fronteira existentes. Após a criação desse vetor é recuperado o valor que está em sua primeira posição, se esse valor for 5, por exemplo, significa que o primeiro nó candidato ao movimento é o 5^{o} nó de fronteira da lista criada anteriormente.

A fim de encontrar para qual território o nó candidato i pode ser movido, e com o intuito de que os possíveis territórios tenham a mesma probabilidade de serem escolhidos, cria-se um outro vetor permutado (Vet_2) de 1 a t, sendo t o número de nós adjacentes ao nó candidato i que pertençam a territórios diferentes ao de i.

A escolha do território destino é feita de forma semelhante ao escolher o nó candidato, recupera-se o primeiro elemento de Vet_2, esse valor indica o j-ézimo nó adjacente ao nó i, que faz fronteira com ele. Conhecendo esse vizinho, é fácil saber a qual território ele pertence e consequentemente sabe-se para qual território pretende-se mover o nó candidato.

- **Garantindo característica 2:**

Com um nó de fronteira escolhido, o algoritmo verifica se o movimento desse nó melhora o valor da função objetivo. Para determinar isso, é feito o seguinte cálculo:

$$\Delta\Psi(S) = \lambda(\Delta F^{+}(S) + \Delta F^{-}(S)) + (1-\lambda)(\Delta G^{+}(S) + \Delta G^{-}(S)) \tag{4.6}$$

Onde:

$$\Delta F^{-}(S) = \begin{cases} (F(V_k)\backslash v) - F(V_k), & se \quad v \quad \in \quad (i,j), \\ 0, & se \quad v \quad \notin \quad (i,j). \end{cases} \tag{4.7}$$

$$\Delta F^{+}(S) = \begin{cases} (F(V_l) \cup v) - F(V_l), & se \quad v \quad \in \quad (i',j'), \\ 0, & se \quad v \quad \notin \quad (i',j'). \end{cases} \tag{4.8}$$

$$\Delta G^{-}(S) = \sum_{a=1}^{2} \left[g^a(V_k \backslash v) - g^a(V_k) \right]. \tag{4.9}$$

$$\Delta G^{+}(S) = \sum_{a=1}^{2} \left[g^a(V_l \cup v) - g^a(V_l) \right]. \tag{4.10}$$

Sendo que:

- v é o nó que está mudando de território;
- V_k é o território de origem;
- V_l é o território destino;
- (i, j) aresta de maior comprimento do território V_k;
- (i', j') aresta de maior comprimento do território V_l;
- $\Delta F^{-}(S)$ é o valor da diferença entre a medida de dispersão do território k após remover o nó v e antes de removê-lo;
- $\Delta F^{+}(S)$ é o valor da diferença entre a medida de dispersão do território l após receber o nó v e antes de recebê-lo;
- $\Delta G^{-}(S)$ é o valor da diferença entre as inviabilidades do território k após remover o nó v e antes de removê-lo;
- $\Delta G^{+}(S)$ é o valor da diferença entre as inviabilidades do território l após receber o nó v e antes de recebê-lo;

Sendo que $\Delta F^{-}(S)$ é calculado para o território fonte e $\Delta F^{+}(S)$ para o território alvo. Caso a remoção do nó não altere a aresta de maior comprimento que forma $F(V_k)$, do território de origem, então o valor de ΔF^{-} é zero. Caso contrário, ΔF^{-} será a diferença entre o valor de F para o território V_k antes e depois da remoção do nó v. ΔF^{+} é calculado de forma análoga.

De forma semelhante, $\Delta G^{-}(S)$ é calculado para o território origem e o $\Delta G^{+}(S)$ para o destino. A forma de calcular o $g^a(V_k)$ é dada na Equação (3.12), sendo que $g^a(V \backslash v)$ representa a violação correspondente à atividade a no território k com a remoção do nó v e $g^a(V_l \cup v)$ é a violação correspondente à atividade a no território l com a inserção do nó v nesse território.

Quanto menor o valor de $\Delta\Psi(S)$ melhor o movimento, já que nosso objetivo é de minimização, de forma que o algoritmo só realiza o movimento se $\Delta\Psi(S) < 0$, situação que indica uma redução no valor da função objetivo equivalente ao valor do $\Delta\Psi(S)$.

Para $\Delta\Psi(S) > 0$, o movimento não melhora o valor da função objetivo.

- **Garantindo característica 3:**

  Se o movimento escolhido melhora o valor da função objetivo então resta apenas verificar se o nó selecionado para o movimento não é um nó de articulação. Para isso é realizada uma busca em largura no território origem, verificando se este território permanece conexo com a remoção do nó i.

Se o movimento candidato falhar no exame de alguma característica, o processo recomeça, primeiro selecionando o próximo elemento de Vet_2, e depois recuperando os elementos de Vet_1, até encontrar um movimento válido. Se o movimento garantir as três características, então ele é executado. Após um movimento ter sido efetuado, todo o processo recomeça. A busca local termina quando atingir um número máximo de movimentos ou quando não houver mais movimentos bons para serem executados.

Busca Local 2 (BL2)

A BL2 segue o mesmo príncipio da BL1, atendendo às mesmas características ao escolher o movimento, porém ao invés de encontrar movimentos que melhorem $\Psi(S)$ ela tenta encontrar e realizar movimentos que melhorem $G(S)$ e $F(S)$ separadamente, sendo que a melhora de um não deve causar piora no outro. Também é executado o primeiro movimento bom encontrado (*first improving*).

Primeiramente foi escolhido melhorar o $G(S)$, então a busca procura por movimentos que diminuam o valor de $G(S)$ sem provocar uma piora no valor de $F(S)$. Ao encontrar estes movimentos, executa-os e a busca por outro movimento recomeça. Quando o algoritmo não encontra mais nenhum movimento que melhore $G(S)$ sem piorar $F(S)$ então começa a procura por movimentos que melhorem $F(S)$ sem piorar $G(S)$.

A melhoria de cada série (G e F) ocorre alternadamente, ou seja, primeiro tenta-se melhorar $G(S)$ até não poder mais, depois procura melhorar $F(S)$ até não conseguir mais e assim sucessivamente. A busca pára quando passar pelas duas séries e não encontrar nenhum movimento de melhora.

- **Melhoria da série G(S):**
  Primeiramente, cria-se um vetor contendo as inviabilidades de todos os territórios da seguinte forma:

$$inviabilidade_k = \sum_{a \in A} g^a(V_k), \qquad A = \{1, 2\} \; e \; k = 1, \cdots, p \tag{4.11}$$

Com o vetor construído, seleciona o território com maior inviabilidade. Conhecendo esse território, encontram-se os nós de fronteira que pertençam a ele. A forma de escolher um nó de fronteira para o movimento é realizada exatamente da mesma maneira feita na BL1.

Conhecendo a lista de todos os nós fronteira pertencentes ao território que se deseja tornar menos inviável, o algoritmo procura por um movimento que irá melhorar $G(S)$, a partir desses nós. Para verificar se o movimento melhora $G(S)$ é feito o cálculo do ΔG como na BL1. Quando um bom movimento é encontrado, a busca executa-o e o processo recomeça.

Quando o algoritmo percorre toda lista de nós candidatos para todos territórios com $inviabilidade_k > 0$ e não encontra nenhum movimento bom, ele para de tentar melhorar $G(S)$ e passa para série $F(S)$.

- **Melhoria da série F(S):**

Ao começar melhorar a série $F(S)$ busca-se pelo território que possui o maior valor de $F(V_k)$, procurando mover o nó i ou o nó j que deram origem a esse $F(V_k)$, pois somente com a remoção de um desses dois nós é possível diminuir o valor de $F(V_k)$. É escolhido um desses nós aleatoriamente e verificado se ele atende às características para um bom movimento, se falhar em alguma delas o outro nó é testado. O primeiro movimento encontrado é realizado. Caso não encontre um bom movimento para i ou para j, passa para o próximo território com maior valor de $F(V_k)$ e todo o processo é repetido .

Quando o algoritmo não encontrar um movimento que melhore o valor de $F(V_k)$ após procurar em todos os territórios, a série $F(S)$ é encerrada e retorna a série $G(S)$. A BL2 também utiliza o cálculo do ΔF como feito na BL1.

A BL2 termina quando passar pelas duas séries e não encontrar nenhum movimento que melhore uma delas sem piorar a outra ou então quando executar um número máximo de movimentos.

Para as duas buscas locais é necessário realizar a atualização dos nós de fronteira após cada movimento. Como um grande número de movimentos é executado por iteração, calcular todos os nós de fronteira após cada movimento acrescenta um grande esforço computacional ao algoritmo. Então procurou-se otimizar tanto quanto possível a forma de atualizar a lista de nós de fronteira. Após estudar o processo de atualização descobriu-se que é necessário verificar, apenas os nós vizinhos ao nó que sofreu o movimento, se a informação de fronteira modificou, fato que é demonstrado a seguir.

Antes de fazer a demonstração é necessário introduzir as seguintes definições:

Postulado 1: Um nó do grafo é um nó de fronteira se existe pelo menos um nó adjacente a ele que pertença a outro território.

Postulado 2: O movimento (i, k, l) modifica a informação de território de um único nó, que é o nó i que está sendo movido do território k para o território l.

Teorema 1: É necessário verificar se o nó j modifica sua informação de fronteira, se e somente se, j for adjacente ao nó i que está sofrendo o movimento (i, k, l).

Prova:
($\Leftarrow$) Provando que se o nó for adjacente ao nó i, então ele precisa ser verificado:
(condição necessária)

Para prova a condição necessária basta encontrar um único caso que mostre que um nó adjacente ao nó que sofreu movimento modificou sua informação de fronteira após o movimento, ou seja, se ocorre para um nó adjacente então é necessário verificar todos os nós adjacentes. Observando o movimento (i, k, l) ilustrado na Figura 4.6 é possível visualizar que os nós j e z são adjacentes a i. Antes do movimento, j é um nó de fronteira e após o movimento ele deixa de ser fronteira. O nó z antes do movimento não é um nó de fronteira e após o movimento passa a ser fronteira. Logo, é necessário checar os nós adjacentes verificando se a informação de fronteira modificou após a execução do movimento.

($\Rightarrow$) Provando que se o nó precisa ser verificado, então ele é adjacente a i:
(condição suficiente):

Isto é o mesmo que provar que se o nó não for adjacente, ele não precisa ser verificado. Essa demonstração é dividida em duas situações:

1. **Nó j, não adjacente à i, não é um nó de fronteira antes do movimento (i, k, l) ser executado:**

   O que se deseja mostrar neste caso é que o nó j após o movimento continua não sendo um nó de fronteira. Por contradição, suponha que o nó j após o movimento (i, k, l) passa a ser um nó de fronteira. Isso implica que algum nó adjacente a j mudou de território. Através da Definição 2 o único nó que muda de território com o movimento (i, k, l) é o i, então i deve ser adjacente a j, o que é uma contradição à hipótese de não adjacência entre o nó i e o nó j.

2. **O nó j, não adjacente a i, é um nó de fronteira antes do movimento (i, k, l) ser executado:**

   O que se deseja mostrar é que o nó j continua sendo um nó de fronteira após o movimento. Por contradição, suponha que o nó j deixa de ser um nó de fronteira, o que significa que todos os seus vizinhos que não faziam parte do seu território passaram a fazer. No entanto pela Definição 2, o movimento só altera o território de um único nó, que é o próprio i. Isso implica que o nó j possuía somente um vizinho de território distinto ao seu, e esse vizinho teria que ser o nó i. O que é uma contradição a hipótese de não adjacência entre o nó j e o nó i.

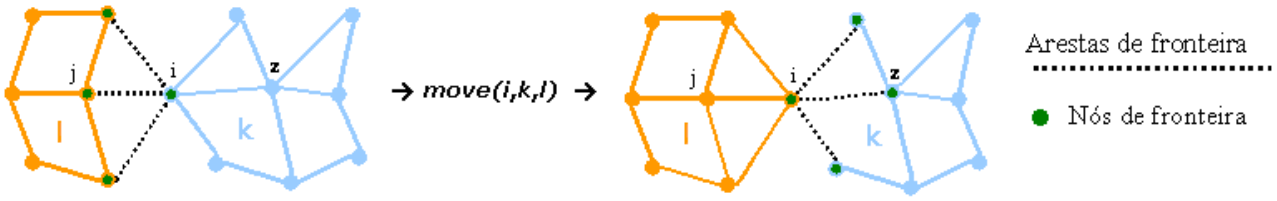


Figura 4.6: Atualização dos nós de fronteira.

4.4.5 Filtragem

A maior parte do tempo consumido pelo algoritmo, certamente está na busca local. Para agilizá-la foi utilizado um filtro proposto por [Ríos-Mercado and Fernández, 2007], com o intuito de evitar a execução da busca local em soluções não promissoras geradas na fase construtiva. Para atingir esse objetivo é armazenado o valor de $\overline{\beta}$, dado por:

$$\overline{\beta} = (\Psi(S) - \Psi(S'))/\Psi(S). \tag{4.12}$$

Sendo que $\overline{\beta}$ representa a redução média obtida pela solução da busca local (S') na iteração anterior com respeito a solução construtiva (S) encontrada na iteração atual.

Após executar as fases construtivas e de ajustamento por 100 iterações, preservando sempre a melhor solução, o algoritmo utiliza a informação de $\overline{\beta}$ para decidir se a solução construtiva deve ou não ser submetida à busca local.

A idéia utilizada é baseada no raciocínio que se um limiar razoável aplicado ao custo da solução construtiva conduz a um valor muito mais alto que o custo da melhor solução encontrada até o momento, é improvável que a busca local consiga produzir uma solução na qual o valor da função objetivo seja menor que o da melhor solução corrente.

Foi utilizado como limiar o valor de $\beta(1-\overline{\beta})$, sendo que β é um parâmetro do algoritmo que mede quão apertado está o limiar. A utilização de um valor alto para β implica que o limiar é satisfeito com menor frequência, conseqüentemente a busca local é executada um número menor

function GRASPFiltro(LimitIt, α, ρ, p)

Entrada: LimitIt := Número máximo de iterções do GRASP;
α := Parâmetro de qualidade da LCR;
ρ := Parâmetro de fechamento do terrritório;
p := Número de territórios.

Saída: Uma solução factível - S^{best}.

1. $S^{best} \leftarrow \{\}$;
2. $soma \leftarrow 0$;
3. $time \leftarrow 0$;
4. $for\ (l = 1, \cdots, LimitIt)\ do$
5. $\quad S \leftarrow Construtivo(\alpha, \rho)$;
6. $\quad q \leftarrow |S|$;
7. $\quad if\ (q \neq p)\ then$
8. $\quad\quad S \leftarrow Ajustamento(S)$;
9. $\quad endif$;
10. $\quad if\ ((l < 100)\ ou\ ((l \geq 100)\ e\ (\beta(1 - \overline{\beta})\Psi(S)) < \Psi(S^{best})))\ then$;
11. $\quad\quad S' \leftarrow BL(S)$;
12. $\quad\quad if\ (\Psi(S') < \Psi(S^{best}))\ then$;
13. $\quad\quad\quad S^{best} \leftarrow S'$;
14. $\quad\quad endif$;
15. $\quad\quad times \leftarrow times + 1$;
16. $\quad\quad soma = soma + (\Psi(S) - \Psi(S'))/\Psi(S)$;
17. $\quad\quad \overline{\beta} \leftarrow soma/times$;
18. $\quad endif$;
19. $endfor$;
20. $return S^{best}$;

end GRASPFiltro.

Algoritmo 4 - Filtro

de vezes. Em contrapartida, se for usado um valor baixo para β então o limiar é satisfeito com maior freqüência e a busca local é executada mais vezes.

A sugestão dada por [Ríos-Mercado and Fernández, 2007] é utilizar valores de β no intervalo $[0, 1]$, sendo que usar $\beta = 0$ corresponde ao caso extremo onde o teste do filtro sempre é satisfeito, naturalmente a BL sempre é executada. O algoritmo 4 mostra o pseudo-código do GRASP juntamente com o filtro.

Até a linha 9 do algoritmo 4 é executada a fase construtiva e de ajustamento, na linha 10 é realizado o teste do filtro para saber se é vantajoso executar a busca local para a solução construtiva, lembrando que durante as 100 primeiras iterações a BL é sempre executada. Após a execução da BL, se ela retornou uma solução melhor que a melhor encontrada até o momento, essa solução é substituída. O valor de $\overline{\beta}$ é atualizado a cada iteração que a BL for executada.

4.5 Métodos e Algoritmos - Otimizando f_2

Após realizar a otimização do critério geográfico, e obter uma boa solução para o mesmo, o objetivo passa a ser otimizar o critério de conformidade. Foram implementadas duas técnicas para otimizar f_2 a fim de rotular os territórios de tal forma que a associação original território-consumidor seja preferencialmente mantida. A primeira técnica implementada foi uma heurística e a segunda foi o método húngaro, essas técnicas são descritas a seguir.

4.5.1 Heurística

A heurística proposta possui basicamente quatro passos principais:

1. Para cada território criado calcule a quantidade de nós que pertenciam aos territórios do agrupamento original. Ou seja, para cada território k da solução atual, é calculado quantos consumidores pertenciam ao território 1, quantos pertenciam ao território 2, e assim sucessivamente para todos os territóros do agrupamento de origem.

2. Escolha um território para ser rotulado, ordene os conjuntos obtidos no passo anterior em ordem decrescente de representatividade, o que significa priorizar os conjuntos com o maior número de consumidores associados. Assim é dado preferência aos territórios que possuem um número maior de medidores que não alteraram de rótulo.

3. Para o território k escolhido no passo anterior avalie c_{kl} para os $j's$ primeiros elementos da ordenação do passo anterior, é escolhido para rotular o território selecionado no passo 2, o rótulo (l) que der o maior valor. Caso percorra os $j's$ primeiros elementos e não encontre um rótulo possível, então escolha o próximo possível, ou seja, que não foi escolhido ainda, a partir de $j + 1$.

4. Se todos os territórios estão rotulados pare, caso contrário retorne ao passo 2.

Nesta heurística busca-se calcular, para cada agrupamento definido, qual o índice de representatividade de cada território da numeração anterior ao reagrupamento. Cria-se um conjunto com a identificação equivalente à numeração anterior e nele são incluídos todos os consumidores que estavam associados a este número de território. Então, ordena-se este conjunto segundo a identificação definida e tomam-se os j primeiros elementos para serem candidatos a fornecer o rótulo para o agrupamento atual.

A escolha entre os j primeiros elementos é feita segundo a função , sendo que f_2, c_{kl} e r_{kl} foram definidos no Capítulo 3.

A cada agrupamento numerado, o rótulo correspondente é excluído da lista de possíveis identificadores de modo que os próximos não usem um rótulo de território já existente. Este processo se repete até que todos os lotes estejam rotulados.

4.5.2 Método Húngaro

O método Húngaro é um algoritmo de otimização combinatória que resolve o problema de designação (*assignment*) em um tempo polinomial. Ele foi desenvolvido e publicado por Harold Kuhn em 1955 quem o nomeou como "método Húngaro" porque o algoritmo foi amplamente baseado no trabalho de dois matemáticos húngaros chamados Dénes König e Jenõ Egerváry [Kuhn, 1995].

Este método utiliza uma matriz-custo não negativa, C de tamanho n x n dada por:

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix}$$

Sendo que o elemento localizado na k-ézima linha e na l-ézima coluna representa o custo de designar o trabalho k ao trabalhador l e o objetivo é encontrar uma alocação de trabalhos e trabalhadores que possua um custo mínimo, a partir da matriz-custo dada.

A soma das n entradas de uma alocação é chamada de custo da alocação, sendo que uma alocação com o menor custo possível é denominada de uma alocação ótima. Antes de apresentar o algoritmo para o método Húngaro é necessário apresentar o teorema da alocação ótima, já que este teorema compreende a idéia principal do algoritmo.

Teorema 2 (Teorema da alocação ótima): Se um número real é somado ou subtraído de todas as entradas de uma linha ou coluna de uma matriz-custo, então uma alocação ótima para a matriz-custo resultante é também uma alocação ótima para a matriz-custo original.

Demonstração:

Seja a matriz $n\ x\ n$:

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1l} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2l} & \cdots & c_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c_{k1} & c_{k2} & \cdots & c_{kl} & \cdots & c_{kn} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nl} & \cdots & c_{nn} \end{bmatrix}$$

Supondo que as entradas da alocação ótima da matriz sejam $c_{1_w1}, c_{2_w2}, \ldots, c_{l_wl}, \ldots, c_{n_wn}$, sendo os índices $1_w, 2_w, \ldots, n_w$ diferentes dois a dois. Logo, o custo mínimo de alocação é a soma de todas estas entradas, da forma:

$$S = c_{1_w1} + c_{2_w2} + \ldots + c_{l_wl} + \ldots + c_{n_wn}. \tag{4.13}$$

Adicionando um valor $u \in \Re$ em todas as entradas de uma coluna da matriz-custo C, obtem-se a seguinte matriz:

$$D = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1l}+u & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2l}+u & \cdots & c_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c_{k1} & c_{k2} & \cdots & c_{kl}+u & \cdots & c_{kn} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nl}+u & \cdots & c_{nn} \end{bmatrix}$$

Utilizando as mesmas entradas da alocação ótima da matriz C, o novo custo de alocação é dado por:

$$S = c_{1_w 1} + c_{2_w 2} + \ldots, (c_{l_w l} + u) + \ldots + c_{n_w n}. \tag{4.14}$$

Pode-se continuar a afirmar que essas entradas correspondem a uma alocação ótima para o problema referente a matriz-custo dada. De fato, qualquer outra seqüência de entradas de D fornece uma soma maior, ou no mínimo igual, a $S + u$, uma vez que, na matriz-custo C a soma mínima é S e em D estão sendo somados $u's$ em todas as entradas de uma coluna. De maneira análoga é feita a demonstração no caso de se adicionar $u \in \Re$ a todas as entradas de uma linha de C.

É possível aplicar o teorema 2, mostrado acima, em uma matriz-custo n x n de tal modo a gerar uma matriz-custo que possua todas as entradas não negativas e, mais ainda, tal que essa matriz possua n zeros sendo que dois deles não estejam na mesma linha ou coluna. Com uma matriz que possua essas características não é difícil encontrar a alocação ótima, pois ela terá soma nula. O algoritmo chamado de Método Húngaro para alocação ótima de facilidades baseia-se nessa idéia.

Aplicado ao problema deste trabalho, tendo n territórios e n rótulos, deseja-se encontrar uma alocação territórios-rótulos de forma que a associação território-consumidor seja mantida o máximo possível.

A matriz-custo é construída da forma definia para $c_{k,l}$ no Capítulo 3, assim o valor recuperado da k-ézima linha com a l-ézima coluna diz quanto custa nomear o k-ézimo território com o l-ézimo rótulo.

Antes de enunciar o algoritmo é importante ressaltar que para aplicar-lo três condições precisam ser atendidas:

- O problema precisa ser um problema de minimização;
- A matriz custo precisa ser quadrada;
- É aconselhável utilizar números inteiros nos valores da matriz-custo a fim de evitar erros de arredondamento.

O método Húngaro pode ser descrito sucintamente nos seguintes passos:

1. Para cada linha k, subtraia o menor valor, obtido dessa linha, de todos os elementos de k;
2. Para cada coluna l, subtraia o menor valor, obtido dessa coluna, de todos os elementos de l;
3. Risque um traço ao longo das linhas e colunas de tal forma que todas as entradas iguais a zero sejam riscadas com um número mínimo de traços. O fator complicador é encontrar esse número mínimo de traços;
4. Se o número de traços para cobrir os zeros é menor que n, continue. Se o número mínimo de traços é igual a n então a matriz possui uma alocação ótima, pare o procedimento;
5. Encontre a menor entrada que não tenha sido riscada. Subtraia esse valor de todas as entradas não riscadas e some a todas entradas riscadas, tanto horizontalmente quanto verticalmente. Retorne ao passo 3.

Como visto anteriormente, uma alocação ótima para a matriz original é uma alocação ótima para a matriz resultante gerada após realizar as operações contidas nos passos 1 e 2. O princípio fundamental desses dois primeiros passos consiste em gerar pelo menos uma entrada igual a zero em cada linha e em cada coluna sendo que as demais entradas são não-negativas. No passo 3, o ponto chave é utilizar um número mínimo de traços que sempre será menor ou igual a n. No passo 4 é verificada a condição de parada. É fácil observar que se o número mínimo de traços é menor que n não é possível identificar uma alocação ótima na matriz-custo obtida.

A partir do teorema da alocação ótima pode-se concluir que n é o número mínimo de traços necessários para cobrir os zeros de uma matriz-custo. De fato, uma alocação ótima nessa matriz será identificada quando existirem n zeros de tal modo que dois deles não estejam em uma mesma linha ou coluna. Ora, nessas condições, são necessários no mínimo n traços para cobrí-los, ou seja, existe uma alocação ótima que corresponde a essas entradas nulas. Esse é o teorema de König, cuja demonstração pode ser encontrada em [Kuhn, 1995].

A representação do problema de rotulamento dos territórios foi feita através de um problema de *assignment*, procurando soluções que otimize f_2. É importante salientar que a solução fornecida pela aplicação de um método exato (Húngaro) para solucionar este problema é um ótimo global, sendo assim esta solução será sempre melhor ou igual a solução obtida através da aplicação de uma heurística.

Capítulo 5

Experimentos Computacionais

5.1 Introdução

Esse capítulo é dedicado à exibição e discussão dos resultados obtidos com a execução dos algoritmos implementados.

Quatro tipos de experimentos são produzidos visando avaliar o desempenho do algoritmo implementado. Os resultados destes experimentos são exibidos nas tabelas apresentadas e discutidas nas seções de cada experimento.

Devido à grande quantidade de dados contidos nas 240 instâncias utilizadas nos experimentos e ao elevado número de linhas do código fonte, não é possível exibir essas informações aqui neste trabalho.

5.2 Dados de Entrada

Os arquivos de entrada possuem alguns dados que descrevem o grafo (a planta da região que se deseja reagrupar). Estes dados são dispostos na seguinte ordem no arquivo:

- **n**: Número de nós do grafo;
- **m**: Número de arestas do grafo;
- **p**: Número de territórios;
- τ**:** Tolerância relativa às atividades;
- (x_i, y_i) **e** p_i**:** Uma lista de tamanho n sendo que x_i e y_i são as coordenadas do nó i e p_i é o território que o nó i está atribuído na configuração inicial.

- $(e_1, e_2)_w$, med_w **e** t_w**:** Uma lista de tamanho m que contém as informações referente às arestas, sendo que $(e_1, e_2)_w$ são os nós adjacentes à aresta w, med_w é o número de medidores e t_w o tempo de leitura desta aresta.

Foram gerados 24 tipos de instâncias com características diferentes em seus dados. Estas características foram classificadas como:

- **Grupo 1 ou 2:** Referente ao número de nós do grafo, instâncias que possuem 512 ou 1024 nós;
- **Família 1 ou 2:** Referente a variação do tempo de leitura (medido em minutos) sob um intervalo apertado $[16, 24]$ ou folgado $[12, 28]$;
- **Classe A ou B:** Referente ao número de medidores, intervalo apertado $[160, 240]$ ou folgado $[120, 280]$;
- **Parâmetro 1, 2 ou 3:** Referente aos valores da tolerância para cada atividade, variação: $\tau = \{0,05; 0,10; 0,30\}$.

A combinação dessas características deram origem a 24 tipos de instâncias. Definidas essas combinações, foram geradas 10 instâncias para cada combinação (tipo de instância), obtendo assim 240 instâncias para a realização dos experimentos. Então, uma instância denominada 1_Grupo1_Familia1_ClasseA_P1, por exemplo, representa a primeira instância do tipo 1 que possui os seguintes dados: 512 nós, o tempo de leitura das arestas variando entre 16 e 24 minutos, o número de medidores de cada aresta variando entre 160 e 240 e $\tau = 0,05$. Todos os grafos do grupo 1 possuem 976 arestas e os do grupo 2, 1984 arestas. A Figura 5.1 ilustra uma instância do grupo 1 e a Figura 5.2 uma do grupo 2.

A geração das coordenadas geográficas de cada nó é feita de forma aleatória tomando cuidado para gerar uma malha uniforme que possua uma boa semelhança à malha de uma cidade, tendo algumas distorções, mas basicamente com a forma "quadrada" para representar os quarteirões. A obtenção de um agrupamento original é realizado executando, exatamente uma única vez, a fase construtiva e de ajustamento para cada instância e guardando a informação dos territórios. Para as instâncias do grupo 1, $p = 10$ e para as do grupo 2, $p = 20$.

5.3 Apresentação dos Resultados

Nesta seção são apresentados os experimentos computacionais realizados nas instâncias descritas na seção anterior. Esses experimentos foram obtidos através dos algoritmos descritos no Capítulo 4, desenvolvidos na linguagem C e utilizados para resolver o problema de reagrupamento capacitado hierárquico com dois critérios.

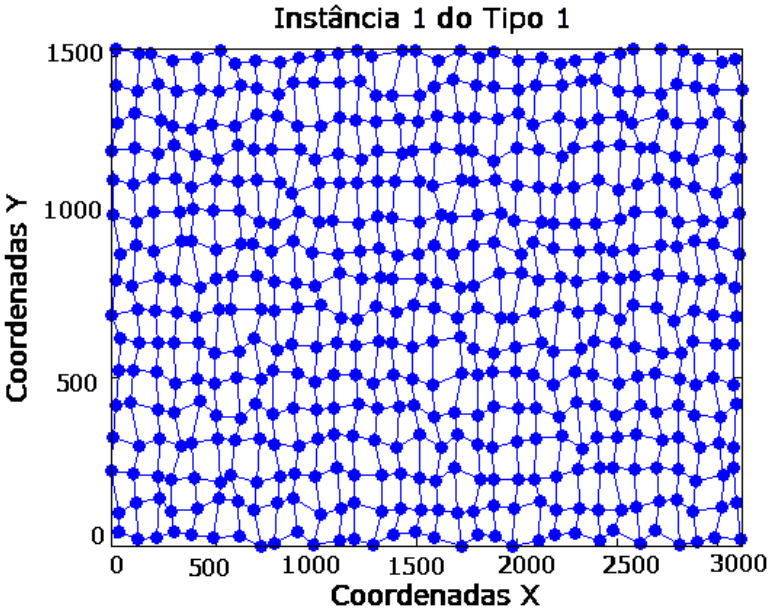


Figura 5.1: Representação de uma instância do Grupo 1.

Os experimentos foram realizados em um PC Intel, Core2 Duo, 3GHz, 3061Mb de RAM e o sistema operacional utilizado foi $WindowsVista^{TM}$.

5.3.1 Experimento 1: Ajuste de Parâmetro

O experimento 1 foi realizado com o objetivo de analisar a sensibilidade do algoritmo, em relação a escolha do parâmetro de fechamento dos territórios (ρ), utilizado na fase construtiva para decidir quando o algoritmo deve parar de atribuir nós ao território que está sendo construído no momento.

Para realizar essa análise foram separadas 20 instâncias para o teste, onde 10 delas pertencem ao grupo 1 e as outras 10 ao grupo 2. O GRASP foi executado sem a fase de pós-processamento, descrito na Seção 4.4.4, utilizando um limite de iterações igual a 1000. Como o algoritmo possui um grau de aleatoriedade, a cada iteração é obtida uma solução distinta, sendo que a melhor solução é sempre preservada.

Os resultados obtidos com este experimento são exibidos nas tabelas 5.1 e 5.2, onde são apresentados os valores da função objetivo (Ψ), medida de dispersão (F), o grau de inviabilidade (G) e o tempo médio gasto pelo algoritmo exibido em segundos. Foi atribuído ao parâmetro de ponderação da função objetivo e ao parâmetro de qualidade da RCL os valores: $\lambda = 0,2$ e $\alpha = 0,3$. Os parâmetros ρ e α variam de acordo com os valores $\rho = \{1,0; 0,8; 0,6\}$ e $\tau = \{0,3; 0,2; 0,1; 0,05\}$.

Os dados exibidos nas Tabelas 5.1 e 5.2 representam os resultados alcançados executando as instâncias de teste para cada parâmetro indicado nas tabelas. Analisando esses resultados, percebe-se que o algoritmo se comporta de forma semelhante para ambos os grupos de instâncias.

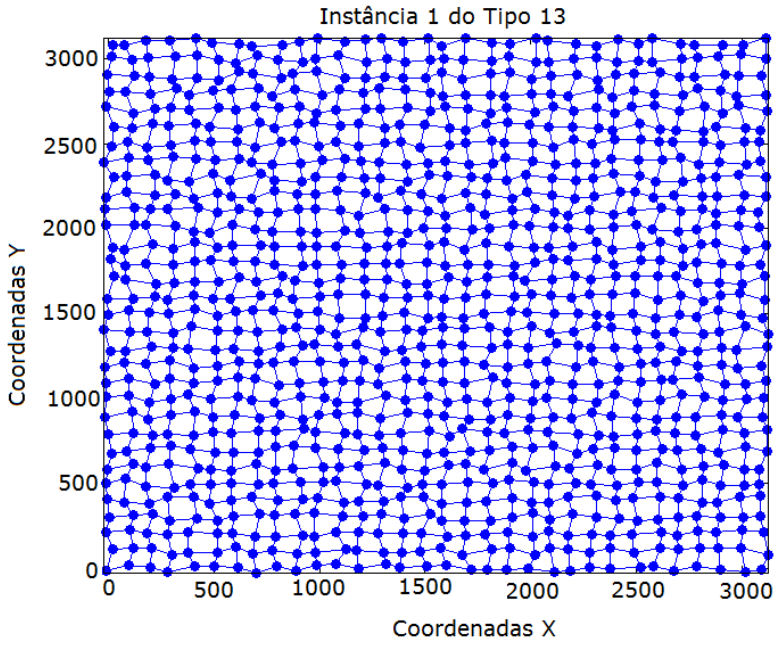


Figura 5.2: Representação de uma instância do Grupo 2.

Nota-se que tanto para instâncias menores (com 512 nós) quanto para instâncias maiores (1024 nós), os melhores resultados são encontrados quando se utiliza $\rho = 0,8$, exceto quando $\tau = 0,05$ nas intâncias do grupo 2 para as quais $\rho = 1,0$ fornece os melhores resultados. A segunda melhor escolha é alcançada com $\rho = 0,6$ quando $\tau = \{0,3; 0,2\}$ e $\rho = 1,0$ quando $\tau = \{0,1; 0,05\}$.

Estes resultados, observando principalmente a segunda melhor escolha, mostram que quando a tolerância para as restrições (3.9) é maior, em média valores pequenos de ρ conseguem alcançar melhores resultados, já quando se tem uma tolerância apertada, valores maiores de ρ obtêm melhores resultados. Esse fato pode ser notado tanto para instâncias pequenas (Tabela 5.1) quanto para instâncias maiores (Tabela 5.2).

É importante ressaltar que o valor de τ não está sendo avaliado neste experimento, pois ele não é um parâmetro ajustável, mas sim um dado de entrada da instância. O valor de τ indica a dificuldade da instância, já que este modifica a função objetivo referente ao primeiro critério. Assim, quanto menor o seu valor mais difícil é encontrar soluções sem inviablilidades.

Analisando as linhas que mostram os resultados referente à inviabilidade das atividades em ambas tabelas, pode-se notar que quanto menor o valor de τ, maior a inviabilidade. Esse fato pode ser explicado porque, quanto menor o valor de τ, menor a tolerância, ou seja, menor o intervalo de soluções factíveis com respeito a G(S) (Figura 4.2). Isso torna mais difícil encontrar soluções em um intervalo pequeno, dado que soluções encontradas fora desse intervalo são penalizadas na função objetivo. Quando o valor de τ é maior, esse intervalo de factibilidade também é maior, sendo que soluções factíveis para um intervalo gerado por um valor de τ grande são penalizadas quando se tem um intervalo gerado por um τ pequeno.

Esse fato pode ser justificado examinando o comportamento da heurística construtiva. Como

Instâncias de 512 nós	$\tau = 0,3$			$\tau = 0,2$		
	$\rho = 1,0$	$\rho = 0,8$	$\rho = 0,6$	$\rho = 1,0$	$\rho = 0,8$	$\rho = 0,6$
$\Psi(S)$						
Melhor	0,92	**0,47**	0,53	2,13	**0,09**	1,47
Médio	2,10	**0,63**	0,86	2,34	**0,43**	1,76
Pior	2,33	**0,76**	0,98	2,61	**0,76**	1,94
$F(S)$						
Melhor	**0,31**	0,33	0,37	**0,33**	**0,33**	0,38
Médio	**0,33**	0,35	0,48	**0,35**	0,36	0,43
Pior	**0,36**	0,45	0,58	**0,40**	0,44	0,50
$G(S)$						
Melhor	1,07	**0,50**	0,58	2,57	**0,05**	1,73
Médio	2,55	**0,70**	0,96	2,83	**0,45**	2,10
Pior	2,82	**0,86**	1,11	3,18	**0,87**	2,32
Tempo(s)	9,55	7,96	6,54	8,99	7,33	6,31
	$\tau = 0,1$			$\tau = 0,05$		
	$\rho = 1,0$	$\rho = 0,8$	$\rho = 0,6$	$\rho = 1,0$	$\rho = 0,8$	$\rho = 0,6$
$\Psi(S)$						
Melhor	1,57	**0,15**	3,10	1,51	**1,33**	3,47
Médio	2,02	**0,44**	3,30	1,84	**1,79**	3,85
Pior	2,56	**0,86**	3,63	**2,35**	2,42	3,96
$F(S)$						
Melhor	**0,31**	0,33	0,37	**0,32**	0,33	0,36
Médio	**0,34**	0,38	0,40	0,39	**0,37**	0,41
Pior	**0,42**	0,48	0,43	0,45	**0,41**	0,54
$G(S)$						
Melhor	1,86	**0,10**	3,77	1,79	**1,57**	4,22
Médio	2,44	**0,46**	4,02	2,20	**2,14**	4,71
Pior	3,11	**0,96**	4,44	**2,85**	2,93	4,48
Tempo(s)	8,20	6,92	6,07	7,89	6,74	5,98

Tabela 5.1: Avaliação do parâmetro ρ para instâncias do grupo 1

mencionado na Seção 4.4.2, a heurística construtiva considera somente o limitante superior. Quando o algoritmo cria um território, os nós vão sendo adicionados ao território corrente enquanto o valor da atividade do território não utrapassar o limite. Assim, os territórios deixam a fase de construção sem inviabilidade, porém com o valor das atividades bem próximo ao limitante. Na maioria das vezes, o algoritmo construtivo cria soluções com um número de territórios maior que o previamente definido pelo decisor, violando assim a restrição (3.7). Torna-se necessário então que esta solução passe pela fase de ajustamento, obrigando a solução atender esta restrição.

Quando o algoritmo construtivo é executado com um valor baixo de τ, o número de territórios criados utrapassa p muito mais do que quando o valor de τ é alto. Isso ocorre porque o intervalo de tolerância é menor no primeiro caso. Por exemplo, para uma instância do grupo 1, executando

Instâncias de	$\tau = 0,3$			$\tau = 0,2$		
1024 nós	$\rho = 1,0$	$\rho = 0,8$	$\rho = 0,6$	$\rho = 1,0$	$\rho = 0,8$	$\rho = 0,6$
$\Psi(S)$						
Melhor	4,50	**0,72**	1,32	3,97	**0,06**	2,51
Médio	4,71	**0,89**	1,50	4,11	**0,06**	2,92
Pior	5,04	**1,01**	1,69	4,29	**0,05**	3,26
$F(S)$						
Melhor	0,27	**0,25**	0,31	0,26	**0,25**	0,30
Médio	0,29	**0,28**	0,34	0,29	**0,27**	0,34
Pior	0,40	**0,33**	0,41	0,36	**0,30**	0,40
$G(S)$						
Melhor	5,52	**0,83**	1,57	4,87	**0,00**	3,06
Médio	5,82	**1,04**	1,79	5,06	**0,00**	3,56
Pior	6,24	**1,19**	2,02	5,29	**0,00**	3,99
Tempo(s)	27,40	23,90	21,71	25,97	22,97	21,18
	$\tau = 0,1$			$\tau = 0,05$		
	$\rho = 1,0$	$\rho = 0,8$	$\rho = 0,6$	$\rho = 1,0$	$\rho = 0,8$	$\rho = 0,6$
$\Psi(S)$						
Melhor	2,96	**1,50**	5,64	**2,13**	3,73	6,28
Médio	3,19	**1,64**	5,84	**2,44**	4,15	7,16
Pior	3,54	**1,85**	6,11	**2,83**	4,56	7,53
$F(S)$						
Melhor	**0,27**	0,30	0,32	**0,27**	0,30	0,32
Médio	**0,30**	0,35	0,34	**0,29**	0,34	0,34
Pior	0,39	0,41	**0,37**	**0,32**	0,40	0,39
$G(S)$						
Melhor	3,57	**1,78**	6,94	**2,59**	4,58	7,77
Médio	3,91	**1,97**	7,21	**2,98**	5,10	8,86
Pior	4,36	**2,22**	7,55	**3,46**	5,60	9,32
Tempo(s)	24,67	22,24	20,78	24,00	21,93	20,66

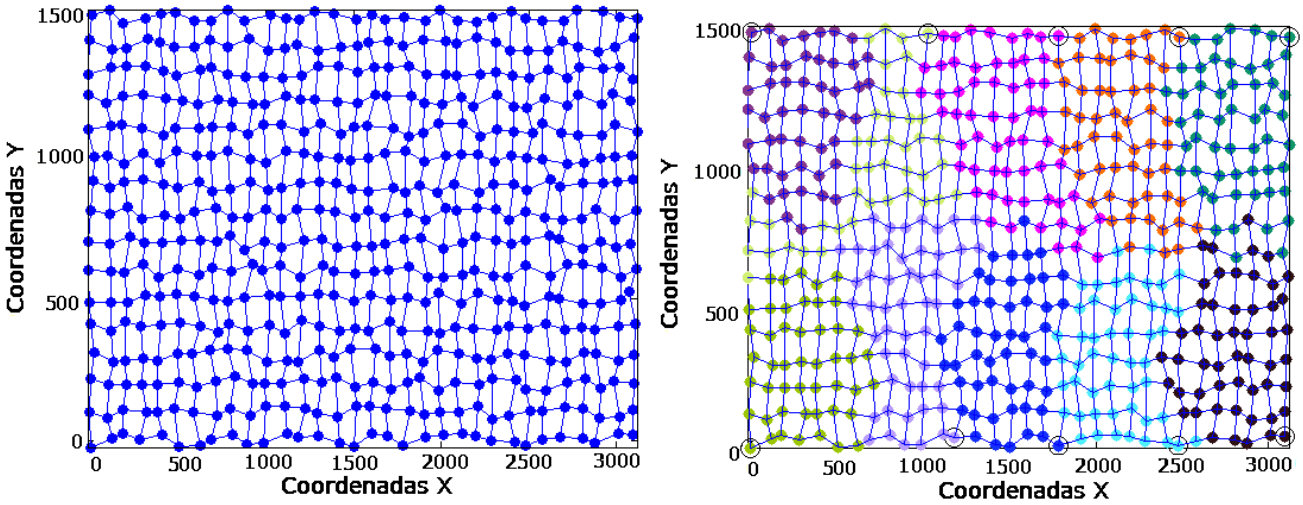
Tabela 5.2: Avaliação do parâmetro ρ para instâncias do grupo 2

o algoritmo construtivo com $\tau = 0,05$ o número de territórios criados variam entre $[18, 25]$, já quando esse algoritmo é executado com $\tau = 0,3$, para a mesma entrada, o número de territórios variam entre $[14, 18]$. É coerente que o processo de executar a operação de *merge* em 25 territórios com o objetivo de alcançar 10 territórios resultará em uma solução com um nível de atividade muito acima do desejado quando comparada a solução obtida pelo mesmo processo para transformar 18 territórios em 10.

Por esse motivo, as soluções encontradas a partir de $\tau = 0,05$ são piores, já que o algoritmo construtivo cria um número bem maior de territórios do que o exigido pelo decisor, fazendo com que na fase de ajustamento ocorram um número maior de operações de *merge*, deixando a solução mais longe do nível de atividade desejada para os territórios que quando é utilizado $\tau = 0,3$, onde a quantidade de *merge* realizados é bem menor, obtendo dessa forma uma solução

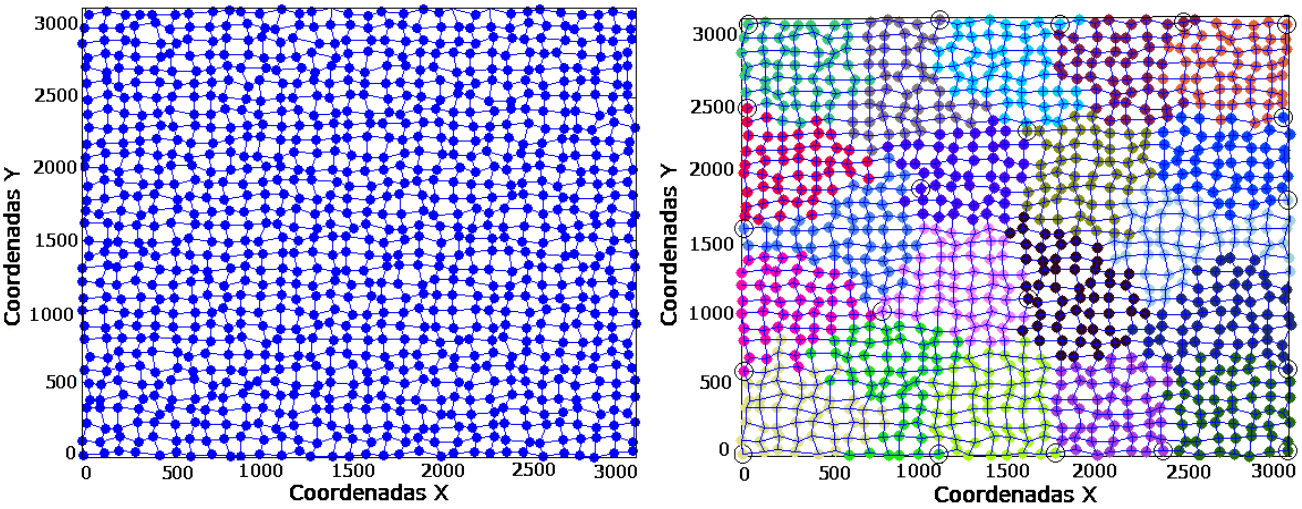
menos inviável.

As Figuras 5.3 e 5.4 ilustram a melhor solução encontrada para uma instância após executar o GRASP sem a BL por 1000 iterações utilizando os parâmetros $\rho = 0,8$ e $\tau = 0,2$.



(a) Dados de Entrada - Isntância 5 do Grupo 1 (b) Solução Construtiva de 5.3(a)

Figura 5.3: Grafo de entrada e a solução encontrada pelo algoritmo construtivo para uma instância do Grupo 1.



(a) Dados de Entrada - Instância 2 do Grupo 2 (b) Solução Construtiva

Figura 5.4: Grafo de entrada e solução encontrada pelo algoritmo construtivo para uma instância do Grupo 2.

5.3.2 Experimento 2: Comparação das BL's sem Filtro

O experimento 2 foi efetuado com o intuito de avaliar os dois procedimentos de busca local que foram implementados. Ambas BL's utilizam o mesmo princípio (*first found*, FF) para

examinar a vizinhança da solução corrente a procura de soluções melhores. Foram realizados alguns testes computacionais, os quais não são reportados na dissertação que mostraram que o princípio FF encontra soluções tão boas quanto o BN (*best neighbor*) e as vezes até melhores, porém com um tempo computacional inferior. Esse fato é confirmado com o mencionado por [Ríos-Mercado and Fernández, 2007], sendo que o autor implementou os dois pricípios chegando a conclusão que o FF era melhor.

A diferença básica entre as duas BL's implementadas neste trabalho é que a BL1 procura por movimentos que melhoram a Função (4.4) e a BL2 por movimentos que melhoram F(S)(3.12) e G(S) (4.5) separadamente, como descrito na Seção 4.4.4.

Neste experimento foi executado o GRASP com a BL1 e com a BL2 separadamente. De forma diferente ao primeiro experimento foram utilizadas todas as 240 intâncias sendo que o valor de τ é atribuído à instância, como descrito na Seção 5.2. Os demais parâmetros utilizados foram: $\rho = 0,8$; $\alpha = 0,3$, $\lambda = 0,2$ e $\rho = 0,8$. Foi utilizado para o GRASP *LimiteIteracoes*=1000 e para as buscas *LimiteMovimentos*=200.

Os resultados obtidos para este experimento são exibidos nas Tabelas 5.4 e 5.5 para a BL1 e BL2 respectivamente, sendo que em cada linha é mostrado o resultado referente a cada tipo de instância. No primeiro conjunto de três colunas, são exibidos valores de Ψ encontrados com o cálculo da Função (4.4), no segundo são mostradas as medidas de dispersão ($F(S)$), no terceiro o valor das inviabilidades com relação às atividades em cada território ($G(S)$) e na última coluna a média do tempo exibida em segundos. Nos conjuntos de três colunas tem-se que S = superior, M = médio e I = inferior. O valor médio é calculado pela média aritmética das soluções obtidas para as 10 instâncias de cada tipo.

Para facilitar a visualização e compreensão dos dados apresentados nas tabelas de resultado dos próximos experimentos, as características que descrevem cada tipo de instância são mostradas na Tabela 5.3.

Esta tabela juntamente com o descrito na Seção 5.2 deixa claro a correspondência existente entre os resultados mostrados nas tabelas e as instâncias que deram origem a estes resultados, sendo que os resultados apresentados nas tabelas seguintes utilizam os valores mostrados.

Observando os resultados mostrados nas Tabelas 5.4 e 5.5, pode-se perceber que ambas BL's conseguiram obter bons resultados. Para constatar isso basta comparar os resultados médios obtidos antes e depois de aplicar as BL's através das Tabelas 5.1 e 5.2. Entretanto, é interessante notar que para instâncias que possuem uma tolerância menor ($\tau = 0,05$), a BL1 obteve melhores resultados e para instâncias mais folgadas ($\tau = 0,10$ e $\tau = 0,30$), a BL2 alcançou melhores resultados, muito embora na maioria das vezes os resultados obtidos pelas duas buscas foram bem próximos.

Quando $G(S)$ está mais restrito, ou seja, com valores menores para τ, a BL1 consegue alcançar melhores resultados. Uma hipótese para justificar isso é pelo fato da BL1 permitir a piora do valor de $F(S)$ para obter uma melhora maior com $G(S)$, levando a crer que dessa forma

Tipo da instância	Grupo 1 ou 2	Família 1 ou 2	Classe 1 ou 2	Tolerância(%) 5, 10 ou 30
1	1	1	A	5
2	1	1	A	10
3	1	1	A	30
4	1	1	B	5
5	1	1	B	10
6	1	1	B	30
7	1	2	A	5
8	1	2	A	10
9	1	2	A	30
10	1	2	B	5
11	1	2	B	10
12	1	2	B	30
13	2	1	A	5
14	2	1	A	10
15	2	1	A	30
16	2	1	B	5
17	2	1	B	10
18	2	1	B	30
19	2	2	A	5
20	2	2	A	10
21	2	2	A	30
22	2	2	B	5
23	2	2	B	10
24	2	2	B	30

Tabela 5.3: Modelo de Representação dos Resultados

a BL1 consegue escapar de alguns ótimos locais. Já a BL2 realiza movimentos de melhora $F(S)$ e $G(S)$, sem permitir a deterioração de nenhuma dessas medidas. Com a análise dos resultados apresentados, percebe-se que quando $\tau = 0,05$ a BL1 em média consegue um resultado melhor, mas não significa que a BL2 não conseguiu alcançar um resultado tão bom quanto o da BL1. É possível observar que, na maioria das vezes, os menores valores encontrados pelas buscas (campo I) são iguais, porém a BL2 consegue encontrar esse resultado em um número menor de vezes. Por outro lado, quando a tolerância é maior, a BL2 consegue encontrar resultados melhores que a BL1, mesmo sendo pequena.

O tempo computacional de ambas BL's é razoavelmente pequeno, sendo que a BL1 precisa de um tempo maior que a BL2 para alcançar a solução final. Observando os tempos computacionais mostrados nas Tabelas 5.4 e 5.5 e comparados aos exibidos nas Tabelas 5.1 e 5.2, é notável o aumento no tempo de execução causado pelo acréscimo da busca local no algoritmo.

Com esses resultados, pode-se concluir que quando se tem instâncias mais apertadas, a BL1 oferece melhores soluções e quando se tem instâncias mais folgadas, a BL2 consegue obter melhores soluções.

	Critério 1 - BL1 sem Filtro									
	$\Psi(S)$			F(S)			G(S)			t(seg.)
Inst	S	M	I	S	M	I	S	M	I	
1	0,26	0,14	0,07	0,40	0,34	0,30	0,25	0,09	0,00	33,69
2	0,06	0,06	0,06	0,32	0,30	0,28	0,00	0,00	0,00	31,09
3	0,34	0,16	0,06	0,37	0,33	0,28	0,35	0,11	0,00	29,90
4	0,20	0,13	0,06	0,43	0,35	0,29	0,16	0,07	0,00	32,51
5	0,07	0,06	0,05	0,33	0,30	0,27	0,00	0,00	0,00	30,84
6	0,42	0,24	0,06	0,36	0,31	0,29	0,45	0,22	0,00	29,23
7	0,65	0,17	0,06	0,39	0,33	0,28	0,73	0,13	0,00	33,37
8	0,08	0,06	0,06	0,42	0,31	0,28	0,00	0,00	0,00	29,92
9	0,39	0,18	0,06	0,45	0,35	0,30	0,41	0,14	0,00	30,22
10	0,52	0,22	0,06	0,40	0,36	0,31	0,55	0,19	0,00	33,31
11	0,07	0,06	0,06	0,31	0,30	0,28	0,00	0,00	0,00	32,08
12	0,42	0,14	0,06	0,39	0,34	0,29	0,44	0,09	0,00	30,52
13	1,28	1,06	0,65	0,38	0,31	0,26	1,52	1,25	0,74	150,33
14	0,47	0,28	0,10	0,36	0,32	0,27	0,50	0,27	0,06	144,73
15	0,06	0,05	0,05	0,29	0,26	0,23	0,00	0,00	0,00	148,37
16	1,33	1,05	0,67	0,32	0,30	0,25	1,57	1,24	0,78	164,78
17	0,48	0,29	0,10	0,38	0,33	0,26	0,51	0,28	0,04	154,64
18	0,06	0,05	0,05	0,28	0,25	0,23	0,00	0,00	0,00	153,80
19	1,25	0,98	0,51	0,44	0,34	0,29	1,48	1,14	0,53	166,30
20	0,47	0,37	0,24	0,41	0,33	0,29	0,49	0,38	0,21	151,23
21	0,05	0,05	0,05	0,27	0,26	0,24	0,00	0,00	0,00	144,12
22	1,54	1,15	0,75	0,39	0,33	0,28	1,86	1,36	0,86	159,04
23	0,50	0,37	0,23	0,36	0,31	0,27	0,51	0,38	0,21	158,43
24	0,06	0,05	0,05	0,31	0,26	0,24	0,00	0,00	0,00	144,80
Média	0,45	0,30	0,17	0,36	0,31	0,27	0,49	0,30	0,14	92,38

Tabela 5.4: Avaliação da BL1

5.3.3 Experimento 3: Avaliação do Filtro

Como mencionado anteriormente e confirmado através dos testes realizados, a maior parte do tempo consumido pelo algoritmo é atribuído à busca local. Procurando melhorar o desempenho do algoritmo em relação ao tempo de execução, foi adicionado ao GRASP um filtro, guiado por $\overline{\beta}$, como apresentado na Equação 4.4.5.

O filtro é utilizado para evitar a execução da busca local em soluções construtivas que não são promissoras. Para aplicar o filtro nas soluções construtivas, a cada iteração, é calculado o valor de $\overline{\beta}$, exibido na Equação 4.12, que representa a redução média obtida pela busca local na iteração anterior, com respeito a solução obtida pela heurística construtiva na iteração atual.

Durante as primeiras 100 iterações do algoritmo, todas as soluções construtivas passam pela busca local, sempre armazenando a melhor solução encontrada. Após essas primeiras iterações,

	Critério 1 - BL2 sem Filtro									
	$\Psi(S)$			F(S)			G(S)			t(seg.)
Inst	S	M	I	S	M	I	S	M	I	
1	0,53	0,20	0,06	0,39	0,31	0,28	0,57	0,17	0,00	23,85
2	0,06	0,06	0,05	0,31	0,29	0,27	0,00	0,00	0,00	24,28
3	0,30	0,16	0,06	0,34	0,31	0,30	0,30	0,12	0,00	21,17
4	0,30	0,19	0,06	0,38	0,32	0,29	0,52	0,16	0,00	23,63
5	0,06	0,06	0,05	0,31	0,29	0,27	0,00	0,00	0,00	24,01
6	0,41	0,12	0,06	0,38	0,32	0,28	0,41	0,08	0,00	21,93
7	0.65	0,29	0,06	0,40	0,32	0,27	0,71	0,28	0,00	23,92
8	0,07	0,06	0,06	0,31	0,29	0,28	0,00	0,00	0,00	23,22
9	0,28	0,12	0,06	0,44	0,32	0,29	0,26	0,07	0,00	22,75
10	0,69	0,31	0,07	0,39	0,35	0,30	0,78	0,29	0,00	36,61
11	0,06	0,06	0,06	0,30	0,29	0,28	0,00	0,00	0,00	25,34
12	0,19	0,08	0,06	0,33	0,31	0,28	0,15	0,02	0,00	21,29
13	1,77	1,22	0,82	0,33	0,29	0,25	2,14	1,45	0,97	105,80
14	0,57	0,19	0,06	0,34	0,31	0,27	0,65	0,16	0,00	102,12
15	0,05	0,05	0,04	0,27	0,24	0,22	0,00	0,00	0,00	107,86
16	1,55	1,33	0,89	0,38	0,32	0,25	1,87	1,59	1,03	108,30
17	0,36	0,25	0,05	0,33	0,30	0,25	0,37	0,23	0,00	111,75
18	0,05	0,05	0,05	0,31	0,25	0,24	0,00	0,00	0,00	115,13
19	1,54	1,38	1,15	0,37	0,30	0,27	1,86	1,65	1,36	103,65
20	0,42	0,30	0,13	0,38	0,30	0,26	0,45	0,30	0,09	113,44
21	0,05	0,05	0,05	0,26	0,24	0,23	0,00	0,00	0,00	108,88
22	1,56	1,35	1,04	0,33	0,31	0,26	1,87	1,62	1,22	114,89
23	0,47	0,29	0,05	0,36	0,31	0,26	0,53	0,29	0,00	109,20
24	0,05	0,05	0,05	0,27	0,25	0,23	0,00	0,00	0,00	108,77
Média	0,50	0,34	0,21	0,34	0,29	0,26	0,56	0,35	0,19	66,74

Tabela 5.5: Avaliação da BL2

o valor de cada solução construtiva é analisada pelo filtro. Se essa for uma solução promissora ela passa pela busca local, caso contrário ela é descartada.

Foi utilizado $\beta \times (1 - \overline{\beta})$ como limiar, sendo que, $0 \leq \beta \leq 1$. A escolha do valor inicial usado para β foi baseado em [Ríos-Mercado and Fernández, 2007], o qual mostra que $\beta = 0,6$ produz resultados bem próximos ao encontrado quando o filtro não é incorporado ao GRASP, além de produzir uma redução considerável no tempo de execução do algoritmo. Este experimento utiliza a BL2 e tem definido os seguintes parâmetros: $LimiteIteracoes = 1000$, $LimiteMovimentos = 500$, $\lambda = 0,2$, $\alpha = 0,3$ e $\rho = 0,8$.

O algoritmo foi executado para as 240 instâncias e os resultados obtidos são apresentados na Tabela 5.6.

Analisando os resultados apresentados na Tabela 5.6, pode-se perceber que o uso do filtro

	Critério 1 - BL2 com Filtro									
	$\Psi(S)$			F(S)			G(S)			t(s)
Inst	S	M	I	S	M	I	S	M	I	
1	0,47	0,27	0,06	0,56	0,36	0,28	0,51	0,25	0,00	8,75
2	0,07	0,06	0,05	0,34	0,30	0,27	0,00	0,00	0,00	10,02
3	0,33	0,15	0,06	0,33	0,31	0,29	0,34	0,12	0,00	10,33
4	0,59	0,31	0,06	0,43	0,34	0,29	0,63	0,30	0,00	8,68
5	0,08	0,06	0,05	0,42	0,31	0,27	0,00	0,00	0,00	9,71
6	0,36	0,21	0,06	0,38	0,32	0,28	0,36	0,18	0,00	10,36
7	0,49	0,34	0,06	0,43	0,32	0,28	0,51	0,34	0,00	9,02
8	0,08	0,06	0,06	0,34	0,31	0,28	0,02	0,00	0,00	9,16
9	0,39	0,17	0,06	0,45	0,34	0,30	0,41	0,13	0,00	10,57
10	0,84	0,44	0,06	0,39	0,35	0,28	0,96	0,46	0,01	9,09
11	0,19	0,08	0,06	0,55	0,36	0,28	0,13	0,01	0,00	8,49
12	0,36	0,26	0,07	0,45	0,34	0,29	0,38	0,25	0,01	10,35
13	1,85	1,65	1,23	0,37	0,32	0,26	2,35	1,99	1,45	30,47
14	0,62	0,42	0,05	0,42	0,30	0,25	0,69	0,45	0,00	31,74
15	0,09	0,06	0,04	0,27	0,24	0,22	0,05	0,01	0,00	32,14
16	1,87	1,66	1,20	0,33	0,30	0,26	2,27	2,00	1,43	30,56
17	0,53	0,30	0,09	0,39	0,32	0,29	0,59	0,30	0,04	30,44
18	0,09	0,06	0,05	0,31	0,25	0,23	0,06	0,01	0,00	32,48
19	1,75	1,55	1,34	0,34	0,29	0,27	2,10	1,86	1,42	29,68
20	0,59	0,33	0,06	0,46	0,30	0,24	0,67	0,34	0,00	28,51
21	0,08	0,05	0,04	0,26	0,24	0,22	0,04	0,00	0,00	31,37
22	1,92	1,76	1,50	0,35	0,29	0,26	2,65	2,13	1,80	30,61
23	0,59	0,38	0,15	0,35	0,30	0,24	0,66	0,40	0,11	28,94
24	0,07	0,05	0,05	0,29	0,26	0,24	0,02	0,00	0,00	32,07
Média	0,59	0,44	0,27	0,38	0,30	0,26	0,68	0,48	0,26	20,14

Tabela 5.6: Avaliação do Filter

para limitar quais soluções construtivas passam pela busca local foi satisfatório, comprovando seu desempenho.

Esse fato é notado comparando os resultados obtidos na Tabela 5.6 com os da Tabela 5.5. A primeira mostra os resultados alcançados executando o GRASP com a BL2 e a segunda exibe os resultados executando o mesmo algoritmo, porém com o filtro acoplado.

Os resultados encontrados adicionando o filtro ao GRASP indicam a eficiência do filtro (Tabela 5.6) para limitar o número de vezes que a busca local é executada, dado que foram encontrados resultados com valores médios muito pouco piores aos da Tabela 5.5, porém com um esforço computacional muito menor comparado ao esforço do algoritmo execultado sem o filtro. O tempo médio de execução foi reduzido em 40% para instâncias do grupo 1 e em 30% para instâncias do grupo 2, sendo que a busca local é aplicada em média a apenas $14,25\%$ das

iterações para as instâncias menores (grupo 1) e a $10,63\%$ das iterações para instâncias maiores (grupo 2).

5.3.4 Experimento 4: Avaliação do Critério de Conformidade

Este experimento foi realizado com o objetivo de avaliar os resultados alcançados pelo algoritmo após adicionar ao problema o segundo critério de otimização (critério de conformidade) apresentado no Capítulo 3, bem como mostrar a evolução das soluções obtidas pelo algoritmo.

Foram utilizados dois métodos para otimizar o critério de conformidade, a heurística e o método Húngaro, ambos discutidos na Seção 4.5.

Foi executado o GRASP utilizando a BL2 e o filtro, para os dois métodos. Os parâmetros utilizados foram: $LimiteIteracoes = 1000$, $LimiteMovimentos = 500$, $\lambda = 0,2$, $\alpha = 0,3$ e $\rho = 0,8$. A heurística é executada apenas uma vez, assim como o Húngaro, já que este é um método exato. O resultado alcançado é atribuído aos territórios, obtendo dessa forma uma nova rotulação para a solução presente.

Os resultados obtidos são exibidos na Tabela 5.7.

	Solução inicial							Critério 1							Critério 2							
	$F(S)_{inicial}$			$G(S)_{inicial}$			t(s)	F(S)			G(S)			t(s)	$f_2^{Heuristica}(\%)$			t(s)	$f_2^{Hungaro}(\%)$			t(s)
Inst	**S**	**M**	**I**	**S**	**M**	**I**		**S**	**M**	**I**	**S**	**M**	**I**		**S**	**M**	**I**		**S**	**M**	**I**	
1	0,53	0,43	0,36	7,36	5,39	4,59	7×10^{-3}	0,41	0,38	0,32	0,41	0,24	0,00	9,10	64,06	56,31	44,36	0,00	65,51	58,94	50,65	0,00
2	0,54	0,47	0,40	4,89	4,49	3,94	8×10^{-3}	0,35	0,30	0,28	0,00	0,00	0,00	8,70	68,93	60,20	47,70	0,00	68,93	61,47	53,53	0,00
3	0,36	0,33	0,30	2,97	2,83	2,66	5×10^{-3}	0,32	0,30	0,28	0,50	0,31	0,01	10,50	75,50	69,14	63,88	0,00	75,50	69,64	65,85	0,00
4	0,51	0,44	0,36	6,34	5,06	4,49	6×10^{-3}	0,39	0,33	0,31	0,57	0,38	0,18	8,68	68,87	56,27	44,70	0,00	68,87	60,66	51,81	0,00
5	0,47	0,42	0,39	4,83	4,13	3,59	3×10^{-3}	0,36	0,30	0,28	0,05	0,00	0,00	9,71	65,05	58,71	50,30	0,00	67,96	60,49	52,55	0,00
6	0,34	0,32	0,31	3,35	2,87	2,61	9×10^{-3}	0,35	0,31	0,29	0,26	0,09	0,00	10,57	67,76	61,97	53,32	0,00	67,76	64,51	59,66	0,00
7	0,47	0,42	0,35	7,28	5,19	4,46	4×10^{-3}	0,42	0,36	0,31	0,61	0,31	0,07	9,16	68,68	57,39	46,53	0,00	68,68	59,98	50,63	0,00
8	0,54	0,44	0,37	5,04	4,27	3,68	4×10^{-3}	0,33	0,30	0,27	0,01	0,00	0,00	9,02	67,68	58,80	47,92	0,00	67,68	60,61	54,42	0,00
9	0,30	0,27	0,24	7,08	6,28	4,89	6×10^{-3}	0,46	0,33	0,30	0,07	0,01	0,00	10,35	49,73	42,19	28,69	0,00	50,63	46,31	43,91	0,00
10	0,50	0,46	0,42	7,20	5,75	4,63	9×10^{-3}	0,43	0,37	0,31	0,64	0,45	0,16	9,36	65,93	52,22	36,82	0,00	65,93	58,09	53,26	0,00
11	0,38	0,42	0,48	5,61	4,19	3,61	9×10^{-3}	0,42	0,33	0,29	0,02	0,00	0,00	8,68	65,55	57,82	46,19	0,00	65,55	59,27	54,56	0,00
12	0,37	0,33	0,31	2,97	2,75	2,47	8×10^{-3}	0,43	0,32	0,29	0,06	0,02	0,00	10,25	73,01	70,66	67,08	0,00	73,01	70,66	67,08	0,00
13	0,30	0,35	0,41	11,39	10,09	9,01	$2,2 \times 10^{-2}$	0,39	0,32	0,27	2,39	2,08	1,41	31,74	59,96	50,44	44,26	0,00	61,42	56,17	52,13	0,00
14	0,44	0,36	0,32	9,44	8,03	6,72	$2,2 \times 10^{-2}$	0,38	0,31	0.26	0,64	0,46	0,14	30,44	57,36	48,64	37,19	0,00	61,06	55,70	51,10	0,00
15	0,42	0,35	0,27	7,15	6,56	6,08	$2,2 \times 10^{-2}$	0,26	0,24	0,22	0,05	0,00	0,00	33,15	65,12	61,05	57,20	0,00	66,42	63,13	61,21	0,00
16	0,47	0,37	0,33	10,66	9,54	8,61	$2,3 \times 10^{-2}$	0,35	0,29	0,26	2,41	1,80	0,79	30,42	56,66	48,08	40,80	0,00	57,62	54,89	50,66	0,00
17	0,51	0,38	0,32	10,09	8,47	8,07	$2,0 \times 10^{-2}$	0,39	0,31	0,27	0,67	0,37	0,05	29,68	62,45	51,33	39,32	0,00	62,45	57,17	51,77	0,00
18	0,40	0,35	0,26	6,94	6,72	6,38	$2,5 \times 10^{-2}$	0,32	0,26	0,24	0,04	0,01	0,00	33,37	64,73	61,71	52,68	0,00	65,76	63,24	58,70	0,00
19	0,38	0,35	0,32	10,67	9,87	9,02	$2,2 \times 10^{-2}$	0,37	0,30	0,26	2,38	1,96	1,48	30,51	59,29	53,14	48,60	0,00	60,64	56,51	51,25	0,00
20	0,41	0,35	0,32	10,07	8,33	7,23	$2,1 \times 10^{-2}$	0,32	0,28	0,24	0,67	0,43	0,00	28,56	60,22	52,63	43,11	0,00	60,22	57,71	53,79	0,00
21	0,37	0,32	0,25	6,83	6,37	5,33	$2,0 \times 10^{-2}$	0,27	0,25	0,23	0,01	0,00	0,00	33,15	66,52	61,42	55,40	0,00	67,62	64,00	59,98	0,00
22	0,49	0,36	0,31	11,83	10,34	8,96	$2,1 \times 10^{-2}$	0,35	0,30	0,26	2,35	2,05	1,56	30,69	66,32	52,08	44,09	0,00	66,32	56,85	49,87	0,00
23	0,41	0,36	0,31	8,86	8,02	7,06	$2,3 \times 10^{-2}$	0,36	0,31	0,26	0,67	0,46	0,01	30,96	58,60	50,36	41,61	0,00	61,49	55,77	51,37	0,00
24	0,38	0,31	0,27	7,01	6,43	5,73	$2,2 \times 10^{-2}$	0,28	0,25	0,22	0,00	0,00	0,00	33,29	63,42	61,29	58,86	0,00	65,58	63,24	60,78	0,00
Média	0,42	0,37	0,33	7,32	6,33	5,57	$1,4 \times 10^{-2}$	0,36	0,30	0,27	0,64	0,47	0,24	20,12	64,22	56,41	47,52	0,00	64,93	59,79	54,65	0,00

Tabela 5.7: Avaliação da Evolução do Algoritmo

Os dados exibidos na Tabela 5.7 localizados nas colunas referente à "solução inicial" foram obtidos executando a heurística construtiva e a fase de ajustamento uma única vez, de forma a adquirir uma configuração inicial de territórios e seus respectivos rótulos que represente uma situação semelhante à real para ser realizado o reagrupamento. São exibidos os valores das inviabilidades dos territórios ($G(S)$), medida de dispersão ($F(S)$) e o esforço computacional. Pode-se observar que as soluções iniciais possuem valores pobres para as medidas $F(S)$ e $G(S)$, principalmente observando os valores referente à $G(S)$.

Em seguida são exibidos os resultados alcançados para as mesmas medidas, agora porém referente ao Critério 1, no qual foi executado o GRASP com a BL2 e o filtro com os parâmetros já mencionados. A otimização do critério 1 trouxe grandes melhoras à solução inicial com um esforço computacional pequeno.

Por fim são mostrados os resultados alcançados executando cada solução do critério 1 com a heurística e com o método Húngaro. Estes resultados representam a porcentagem de medidores que permaneceram no mesmo território depois do reagrupamento.

Para os resultados obtidos otimizando o critério 2, quanto maior for a porcentagem de medidores que não alteraram seu rótulo, melhor o valor da função objetivo. Analisando estes resultados, pode-se observar que tanto a heurística quanto o método Húngaro obtiveram bons resultados, porém o método Húngaro alcançou melhores resultados que a heurística, o que de fato era o esperado já que o segundo é um método exato. No entanto, a heurística conseguiu encontrar valores próximos ao Húngaro, principalmente para as instâncias do grupo 1. O tempo de execução de ambos métodos para todas as instâncias foi muito baixo não agregando um esforço computacional relevante ao algoritmo. Visto que o método Húngaro alcança melhores soluções e seu tempo computacional é baixo, não se justifica o uso de uma heurística para resolver o critério de conformidade.

A Figura 5.5 ilustra o desenvolvimento da solução obtida pelo algoritmo em cada etapa do processo de otimização. Ela mostra cada passo do desenvolvimento do algoritmo para uma instância do grupo 1. É exibido primeiramente a configuração inicial da região estudada e seus rótulos originais na Figura 5.5(a), esse grafo é uma instâcia para o algoritmo. Na Figura 5.5(b) é mostrada a melhor solução encontrada após executar a fase construtiva e de ajustamento por 1000 iterações. A Figura 5.5(c) mostra a solução excutando o algoritmo também por 1000 iterações, como a figura anterior, porém é acrescentado a busca local nesta etapa. Por último é mostrada na figura 5.5(d) o resultado final do algoritmo.

É possível observar a melhora ocorrida no critério geográfico, a solução final apresenta o desenho dos territórios mais compactos que os mostrados na configuração inicial. A solução apresentada na Figura 5.5(c) e 5.5(d) é a mesma solução comparada a forma geográfica de cada uma delas, o que as diferencia são os rótulos dados a cada uma. É importante salientar que os rótulos atribuídos às Figuras 5.5(b) e 5.5(c) são meramente posições da estrutura de dados, pois nelas o critério de conformidade ainda não foi aplicado ao problema.

Como mencionado anteriormente o algoritmo foi executado para uma rede real. Foram obtidos alguns dados referentes a uma área da cidade de São Paulo, estes dados foram fornecidos pela AES Eletropaulo. As imagens[1] exibidas na Figura 5.6 mostram esta área. A Figura 5.6(a) representa uma área da cidade de São Paulo e seus clientes associados. Foram selecionadas as duas regiões que possuem a maior quantidade de consumidores agrupados para o tratamento da não-conectividade (Figura 5.6(b) e Figura 5.6(c)). A união dessas duas regiões resultou na região exibida na Figura 5.6(d), que foi utilizada na execução do algoritmo. As demais regiões ficam isoladas no mapa e como os tamanhos dessas regiões são pequenos elas não foram consideradas.

O algoritmo foi executado para a região exibida na Figura 5.6(d). Para esta rede os parâmetros do algoritmo foram: $LimiteIteracoes = 1000$, $LimiteMovimentos = 500$, $\lambda = 0,2$, $\alpha = 0,3$, $\rho = 0,8$.

Os dados exibidos na Tabela 5.8 mostram os resultados para a região exibida na figura 5.6(d) onde os pesos das arestas variam entre: $medidores = [120, 280]$ e $tempo = [12, 28]$. E os dados exibidos na Tabela 5.9 mostram os resultados para a mesma região, porém os pesos atribuídos às suas arestas variam entre: $medidores = [1, 100]$ e $tempo = [20, 2000]$.

	Solução inicial			**Critério 1**			**Critério 2**	
Tau	$F(S)_{inicial}$	$G(S)_{inicial}$	t(s)	F(S)	G(S)	t(s)	$f_2^{Hungaro}(\%)$	t(s)
0,05	0,35	5,60	$6,5 \times 10^{-2}$	0,29	0,10	66,03	73,69	1×10^{-3}
0,10	0,37	5,86	$7,0 \times 10^{-2}$	0,29	0,14	64,38	70,17	1×10^{-3}
0,30	0,25	6,73	$7,8 \times 10^{-2}$	0,29	0,00	78,53	64,72	1×10^{-3}

Tabela 5.8: Resultado Rede São Paulo - pesos apertados

	Solução inicial			**Critério 1**			**Critério 2**	
Tau	$F(S)_{inicial}$	$G(S)_{inicial}$	t(s)	F(S)	G(S)	t(s)	$f_2^{Hungaro}(\%)$	t(s)
0,05	0,31	5,12	$6,7 \times 10^{-2}$	0,29	1,23	64,88	71,88	1×10^{-3}
0,10	0,34	7,76	$6,8 \times 10^{-2}$	0,29	0,13	64,52	69,65	1×10^{-3}
0,30	0,32	6,98	$7,6 \times 10^{-2}$	0,35	0,00	78,05	69,40	1×10^{-3}

Tabela 5.9: Resultado Rede São Paulo - pesos folgados

Assim como os demais experimentos, os resultados encontrados através da execução do algoritmo para a rede de São Paulo, exibidos nas Tabelas 5.8 e 5.9, comprovam a eficiência e eficácia do algoritmo implementado, o qual conseguiu alcançar um bom resultado com um esforço computacional pequeno, tanto para a rede com atividades leves quanto para atividades mais pesadas.

Um fato que deve ser ressaltado é que para uma variabilidade maior nos valores das atividades (número de medidores e tempo de leitura), a instância fica mais difícil, por isso os valores obtidos para $F(S)$ e $G(S)$ tendem a piorar em relação aos valores encontrados para uma rede com menor variabilidade do valor dessas atividades.

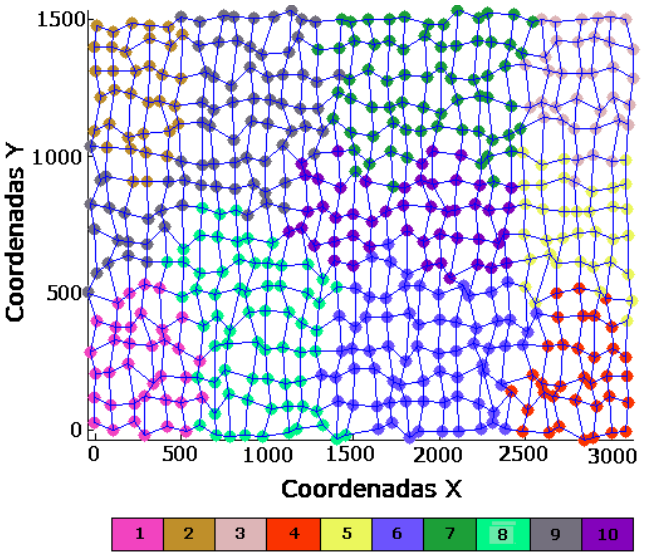
[1]Imagens fornecidas pela AES Eletropaulo.

A região utilizada no processo de otimização não fornece um grafo muito maior que os grafos de teste do Grupo 2. O grafo para a região representada pela Figura 5.6(d) possui 1659 nós e 2408 arestas sendo que foi utilizado $p = 20$. Porém uma grande diferença desse grafo para os utilizados até o momento é a forma geográfica que os nós estão dispostos. Na rede de São Paulo encontramos uma malha tortuosa, na qual o algoritmo poderia encontrar dificuldades ao otimizar o critério geográfico, no entanto isso não ocorre, pois os valores encontrados para $F(S)$ são tão bons quanto para os grafos com malhas mais bem comportadas.

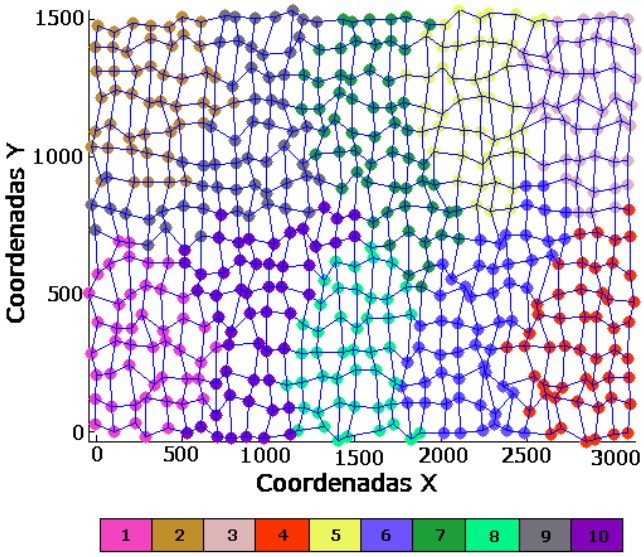
Para facilitar a visualização da solução a Figura 5.7 mostra os resultados obtidos em cada etapa do processo de otimização, utilizando a instância que gerou o resultado apresentado na 3ª linha da Tabela 5.8.

A Figura 5.7(a) mostra a configuração inicial, que é considerada como entrada do algoritmo. A Figura 5.7(b) exibe a melhor solução encontrada após a execução da heurística construtiva e da fase de ajustamento. A Figura 5.7(c) mostra a melhor solução encontrada após a execução do algoritmo com o acréscimo da busca local. Por último é mostrado na Figura 5.7(d) o resultado final do algoritmo aplicando o método Húngaro para rotular os territórios.

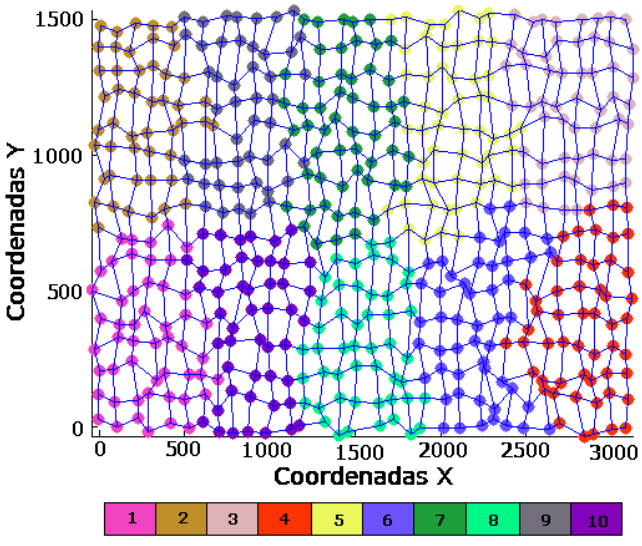
É interessante observar a melhora no critério geográfico começando pela solução inicial (Figura 5.7(a)) até chegar à solução final (Figura 5.7(d)). Observando as duas últimas figuras (5.7(c) e 5.7(d)), nota-se que a configuração dos territórios não se altera; somente suas cores são alteradas. Pois, para gerar a solução exibida na Figura 5.7(d), utilizou-se a solução do critério 1 e aplicou-se o método Húngaro para rotular os territórios.



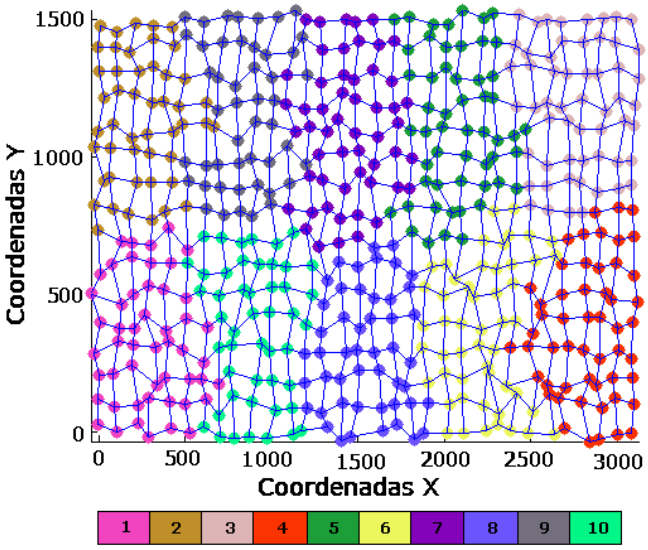
(a) Instância inicial com agrupamentos e rótulos originais.



(b) Solução obtida com a heurística construtiva



(c) Solução obtida com BL2



(d) Solução Final com método Húngaro

Figura 5.5: Soluções encontradas durante as etapas do algoritmo.

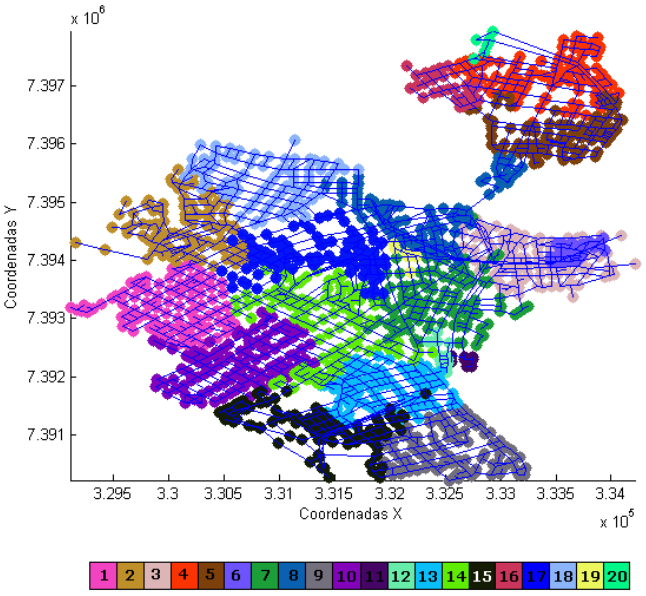
(a) Área região de São Paulo com clientes associados.

(b) Região 1

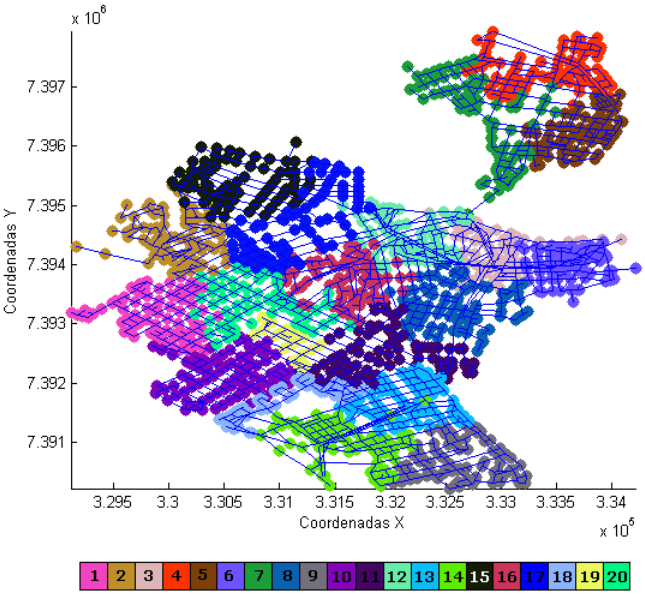
(c) Região 2

(d) Região 1 e 2

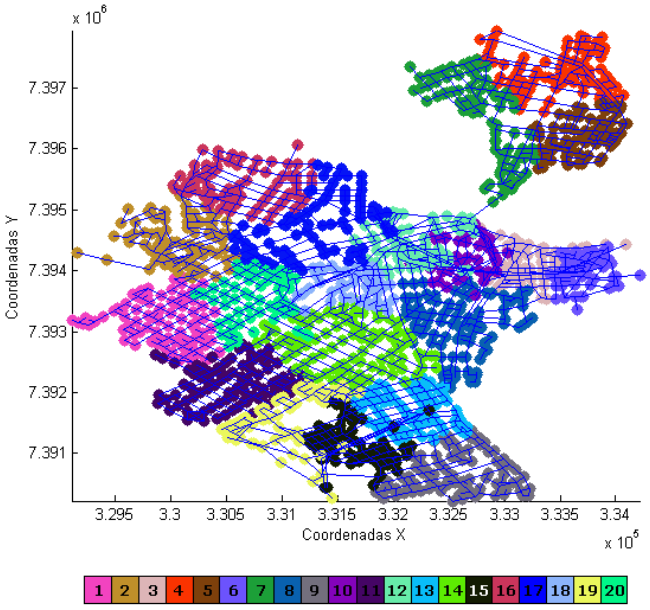
Figura 5.6: Área Piloto e Regiões tratadas pelo algoritmo.



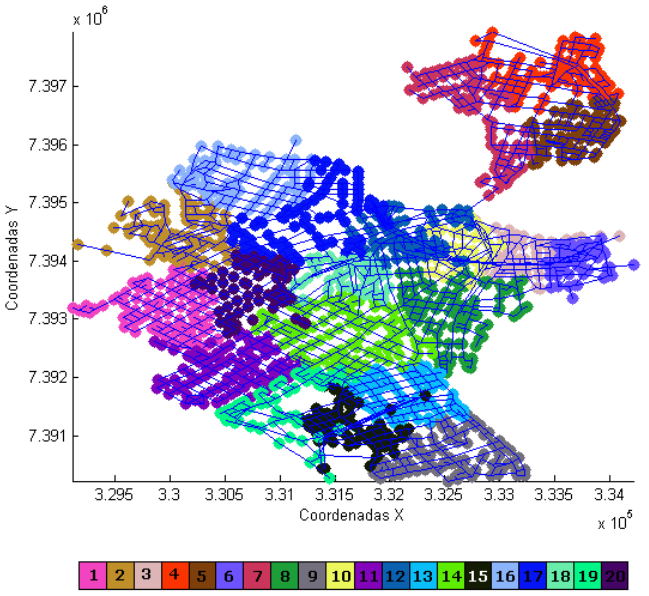
(a) Rede de São Paulo com agrupamentos e rótulos originais.



(b) Solução obtida com a heurística construtiva



(c) Solução obtida com BL2



(d) Solução Final com método Húngaro

Figura 5.7: Soluções encontradas durante as etapas do algoritmo.

Capítulo 6

Conclusão

O principal objetivo deste trabalho foi desenvolver uma metodologia eficiente para resolver o PACM aplicado ao problema de reagrupamento de lotes de faturamento para concessionárias de distribuição de energia elétrica. O PACM foi tratado neste trabalho como um problema de reagrupamento capacitado hierárquico com 2 critérios.

Para resolver o critério geográfico foi desenvolvida uma metaheurística baseada no GRASP que contém: uma heurística construtiva para geração de soluções iniciais; uma fase de ajustamento que procura factibilizar o problema referente ao nível de atividade de cada território; uma busca local que procura melhorar o valor da função objetivo da solução obtida pela heurística construtiva.

Para avaliar o desempenho da metodologia implementada para o critério 1 foram realizados alguns experimentos os quais mostraram a eficiência do algoritmo obtendo bons resultados. Na maioria dos resultados o algoritmo conseguiu alcançar soluções sem inviabilidades referente às atividades dos territórios.

Para resolver o critério de conformidade, sendo que o objetivo do problema passou a ser encontrar a melhor rotulação possível para a solução obtida com a otimização do critério geográfico. Para otimizar o novo problema, foi implementada uma heurística que procura rotular os novos territórios causando o mínimo de alteração possível referente à conformação anterior. Também foi utilizado o método Húngaro, que é um método exato para resolver problemas de designação.

Foi realizada a avaliação e comparação dos dois métodos utilizados para otimizar o critério de conformidade, sendo que ambos obtiveram bons resultados. Embora a heurística em diversas vezes tenha conseguido alcançar a mesma solução que o método Húngaro, na maior parte as soluções produzidas pelo método Húngaro superaram as soluções encontradas pela heurística, o que é esperado ja que o primeiro é um método exato. Os resultados mostraram que não há necessidade de usar uma heurística para resolver este problema, já que o método exato nos fornece a solução ótima em um tempo computacional pequeno.

Com o objetivo de avaliar o parâmetro de fechamento (ρ) dos territórios para a fase construtiva, foram realizados alguns experimentos computacionais utilizando os valores $\rho = 1,0;\quad 0,8;\quad 0,6$ e $\tau = 0,3;\quad 0,2;\quad 0,1;\quad 0,05$, os quais mostraram que o valor de ρ que produz as melhores soluções na maioria dos experimentos foi $0,8$. Observando a segunda melhor escolha verificou-se uma dependência do melhor valor para o parâmetro ρ com o dado de entrada τ (tolerância para as atividades) da instância: quanto maior o valor de τ, melhores resultados foram obtidos para menores valores de ρ e vice-versa.

De modo geral, este trabalho contribuiu com idéias e estratégias para resolução do PACM. Embora tais estratégias tenham sido aplicadas ao problema de reagrupamento capacitado de lotes de faturamento para concessionárias de distribuição de energia elétrica, elas também podem ser utilizadas em outras aplicações.

A maior preocupação foi a de conseguir realizar o reagrupamento de forma eficiente evitando, sempre que possível, a formação de territórios desconexos, já que esta é uma das maiores dificuldades apresentadas em trabalhos presentes na literatura, além de ser difícil encontrar métodos eficientes para resolver este problema. Este trabalho apresentou uma maneira eficiente de evitar a formação de territórios desconexos. No processo construtivo, um nó só é adicionado a um território quando há adjacência entre ambos. Na busca local, é feito uma pesquisa em largura em todos territórios candidatos a perder um nó, verificando se ele continuará conexo após a remoção.

Além das instâncias geradas para teste o algoritmo também foi executado para uma rede real, conseguindo bons resulados. O maior desafio encontrado era que o algoritmo conseguisse otimizar, de forma eficiente, o critério de compacidade dessa rede, já que ela não forma um grafo bem comportado como os grafos das instâncias de teste. Apesar dessa caracterísca do grafo o algoritmo não encontrou dificuldades, conseguindo alcançar resultados tão bons quanto os das instâncias de teste.

A heurística proposta mostrou-se eficiente, gerando boas soluções com um elevado grau de compromisso com a compacidade, homogeneidade e conformidade, sendo que o esforço computacional é relativamente pequeno e garante a conectividade dos territórios gerados.

Apêndice

Informações Complementares sobre Estruturas de Dados

Neste apêndice é apresentada um breve introdução sobre estruturas de dados, procurando esclarecer as estruturas que foram utilizadas no trabalho.

Estrutura de Dados

Introdução

Os algoritmos manipulam diversos dados, como eles são organizados caracterizam sua forma ou estrutura. As estruturas de dados são chamadas tipos de dados compostos que dividem-se em dois grupos distintos: homogêneos, que são os vetores e matrizes, e heterogêneos, que são os registros. As estruturas homogêneas são conjuntos de dados formados pelo mesmo tipo de dado primitivo. As estruturas heterogêneas são conjuntos de dados formados por tipos de dados primitivos diferentes em uma mesma estrutura que são denominados como campos do registro. A Figura 6.1 exemplifica esses dois tipos de estruturas de dados.

A boa representação dos dados de um problema é fundamental para que se tenha um bom desempenho na implementação de algoritmos. A escolha de uma estrutura de dados apropriada pode facilitar a resolução de problemas complicados.

Este capítulo se destina a descrever a estrutura de dados utilizada para armazenar as informações do problema, a fim de manuseá-las computacionalmente facilitando a resolução do mesmo. Porém antes de apresentar a estrutra utilizada se faz necessário mostrar algumas estruturas clássicas que foram utilizadas como base para as estruturas criadas neste trabalho.

Vetores

Os vetores podem ser considerados como as estruturas de dados mais simples que é possível implementar, são estáticas, ou seja, compostas por um número fixo de elementos de um determinado tipo de dados. Sua forma mais simples é o vetor unidimensional que pode ser definido como um conjunto finito e ordenado de elementos homogêneos. Por finito entende-se que o tamanho do vetor pode ser grande ou pequeno, mas deve existir previamente, ordenado

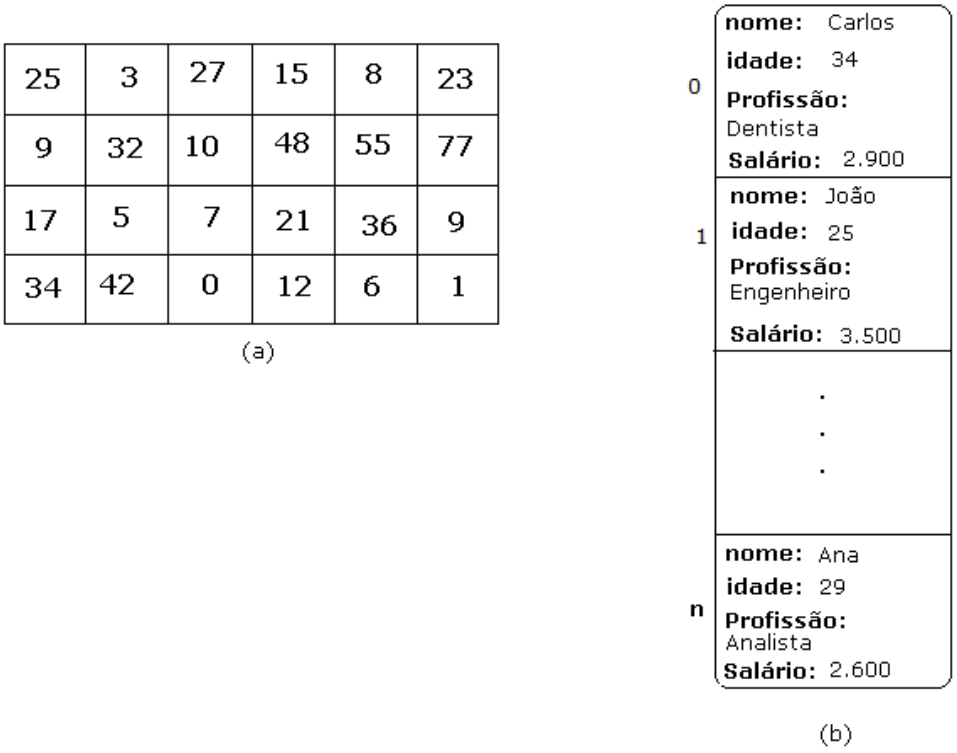


Figura 6.1: Estruturas de dados homogêneas (a) e heterogêneas (b).

indica que deve existir o elemento zero, o primeiro, o segundo, e assim sucessivamente e por homogêneos entende-se que o vetor pode guardar dados do tipo inteiro ou caracteres, por exemplo, mas não ambos simultâneamente [Tenenbaum et al., 1995]. Cada elemento do vetor é acessado por sua posição dada por um índice que geralmente utiliza uma seqüência de números inteiros. As matrizes são vetores bidimensionais, os quais são úteis para descrever objetos que são fisicamente bidimensionais. A Figura 6.2 mostra exemplos de vetores unidimensionais e bidimensionais.

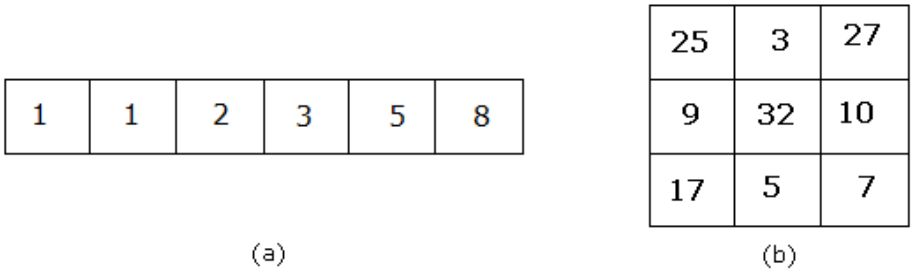


Figura 6.2: Exemplos de vetores unidimensionais (b) e bidimensionais (a).

Conforme a necessidade, pode-se definir vetores com quantas dimensões forem necessárias.

Os vetores possuem a vantagem de que os seus elementos são acessíveis de forma rápida, mas têm uma notável limitação, são de tamanho fixo, não podem ser incrementados ou diminuídos

sem implicar complexos processos de cópia e a remoção de elementos pode ser custosa se for desejável que o vetor não tenha espaços vazios entre elementos.

Filas

As filas são conjuntos ordenados de itens a partir do qual podem-se inserir itens numa extremidade e eliminar itens na outra extremidade. Estas estruturas de dados são comumente conhecidas como FIFO(*first in, first out*), nome que descreve sua principal característica, onde o primeiro elemento a entrar na fila é o primeiro a sair [Cormen et al., 2002]. As filas são conjunntos dinâmicos que procuram representar exatamente uma fila de elementos. Essas estruturas são implementadas em larga escala para representar filas de espera, pois os elementos são colocados e retirados (ou processados) por ordem de chegada.

A fila possui um início e um fim, sendo que sua idéia fundamental é que só se pode inserir um novo elemento no final da fila e só se pode retirar o elemento de seu início. A Figura 6.3 ilustra uma fila que possui sete elementos, onde 3 é o início da fila e 4 é o final.

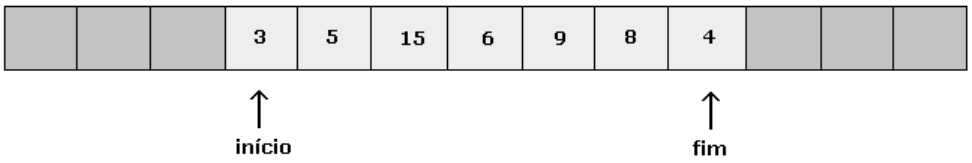


Figura 6.3: Exemplo de uma fila.

O algoritmo para implementação de uma fila pode ser encontrado em [Cormen et al., 2002].

Listas Encadeadas

Existem algumas desvantagens de se representar filas e outras estrutras de dados semelhantes usando o armazenamento sequêncial. O que ocorre é que uma grande quantidade fixa de armazenamento permanece alocada mesmo quando a estrutura estiver utilizando de fato uma quantidade menor ou possivelmente nenhum armazenamento. Além disso somente a quantidade fixa pré-estabelecida para o armazenamento poderá ser utilizada, tendo dessa forma a possibilidade de estouro da fila [Tenenbaum et al., 1995]. Uma solução para obter maior flexibilidade é a utilização de listas encadeadas.

Uma lista encadeada é uma estrutura em que os objetos estão organizados de forma linear, porém diferente dos vetores no qual a ordem linear é determinada pelos índices do vetor, nesta estrutura a ordem é determinada por um ponteiro em cada objeto [Cormen et al., 2002]. Cada item na lista é denominado de células e contém dois campos um para armazenar os dados e outro para guardar o endereço da próxima célula da lista. O campo de dados armazena o real

elemento da lista, o campo de endereço guarda onde está na memória a próxima célula da lista. Este campo é conhecido como ponteiro sendo muito importante para manter a lista organizada. A lista inteira é acessada a partir de um ponteiro externo, que guarda o endereço para sua primeira célula. O ponteiro da última célula da lista guarda seu final que é indicado por NULL, o qual não é um endereço válido [Tenenbaum et al., 1995].

A Figura 6.4 mostra uma lista encadeada com 3 elementos.

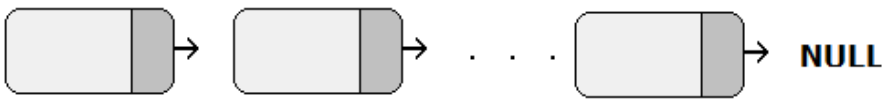


Figura 6.4: Diagrama conceitual de uma lista encadeada.

As listas encadedas fornecem uma representação simples e flexível para conjuntos dinâmicos. As vantagens de se usar listas encadeadas e não vetores é que a inserção e remoção de um elemento na lista não implica em mudança de posição dos outros elementos. Não é necessário conhecer antecipadamente o número máximo de elementos da lista, o que implica em não desperdiçar memória. Entretanto as filas também possuem suas desvantagens, a manipulação dos dados torna-se mais perigosa pois se a ligação entre as células da lista for mal realizada toda lista pode ser perdida. Para fazer o acesso do n-ézimo elemento da lista os $n-1$ elementos devem ser percorridos.

Os algoritmos das operações para manipulação de listas encadeadas são apresentados em [Cormen et al., 2002].

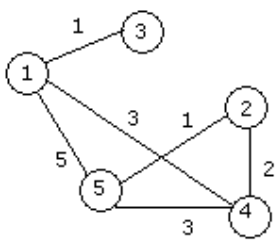
Estruturas Implementadas

Como mencionado na Seção 3.3 o problema de reagrupamento capacitado hierárquico foi representado como um grafo formado por um conjunto de n nós e m arestas, onde cada aresta do grafo representa um trecho de rua de uma planta urbana e cada nó a encruzilhada entre duas ruas. O problema traz informações que precisam ser armazenadas de forma adequada para que o algoritmo não encontre grandes dificuldades de trabalhar com esses dados.

Nesta seção são mostradas e discutidas as principais estruturas utilizadas para armazenar e tratar os dados do problema deste trabalho.

Matriz de Distância

Foi utilizada uma matriz bidimensional de tamanho $n \times n$ que guarda a distância euclidiana entre cada par de nós do grafo. A Figura 6.5 mostra uma idéia de como é essa matriz para um grafo de quatro nós.



Dist =

	1	2	3	4	5	
	0	∞	1	3	5	1
	∞	0	∞	2	1	2
	1	∞	0	∞	∞	3
	3	2	∞	0	3	4
	5	1	∞	3	0	5

Figura 6.5: Matriz de distância.

O cálculo de cada posição dessa matriz é realizado utilizando os valores das coordenadas de cada nó, sendo que a equação da distância foi mostrada na Equação (3.1).

Listas de nós

Foram utlizadas três listas encadeadas V_q, $N(V_q)$ e RCL. V_q é uma lista que guarda todos os nós pertencentes ao território q. $N(V_q)$ guarda todos os nós vizinhos ao território V_q. Esta lista é utilizada na heurística construtiva com o objetivo de encontrar os vizinhos do território que está sendo construído, o que a torna muito importante, pois dessa forma é garantida a conectividade dos territórios criados pela heurística construtiva. RCL é a lista de nós candidatos a entrar no território corrente, esta lista também é utilizada pelo algoritmo construtivo e é criada selecionando candidatos a partir da $N(V_q)$, avaliando o valor da função $\phi(v)$ para cada nó pertencente a $N(V_q)$. Esta função é mostrada na Equação (4.1).

A estrutura dessas três listas são semelhantes, sendo que V_q guarda apenas os nós pertencentes ao território q e as listas $N(V_q)$ e RCL guardam os nós vizinhos e os nós candidatos respectivamente, como também o grau de cada um e seu valor de $\phi(v)$.

A Figura 6.6 ilustra a estrutura dessas listas.

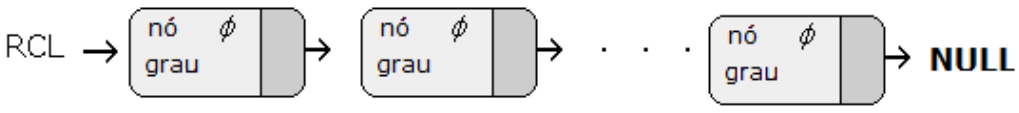


Figura 6.6: Lista de Candidatos Restrita.

Na heurística construtiva estas listas iniciam-se vazias. A medida que os nó são inserido no território corrente V_q, a lista $N(V_q)$ é criada/atualizada guardando os nós vizinhos ao território q, depois que a lista de vizinhos está criada é calculado o ϕ para cada vizinho e a RCL é criada com os vizinhos que estiverem dentro do intervalo mostrado no algoritmo 2.

A justificativa para utilização de listas encadeadas na maior parte das estruturas é devido a facilidade de mover um elemento de uma lista para outra, operação que ocorre com muita

freqüência na heurística construtiva e entre os territórios, durante a BL.

Estrutura Grafo

Foi criada uma estrutura para guardar as informações originais do grafo, as quais são obtidas através dos dados de entrada do problema. Essa estrutura é um vetor de nós de tamanho n, onde cada posição i desse vetor guarda informações referentes ao nó i. A Figura 6.7 ilustra essa estrutura.

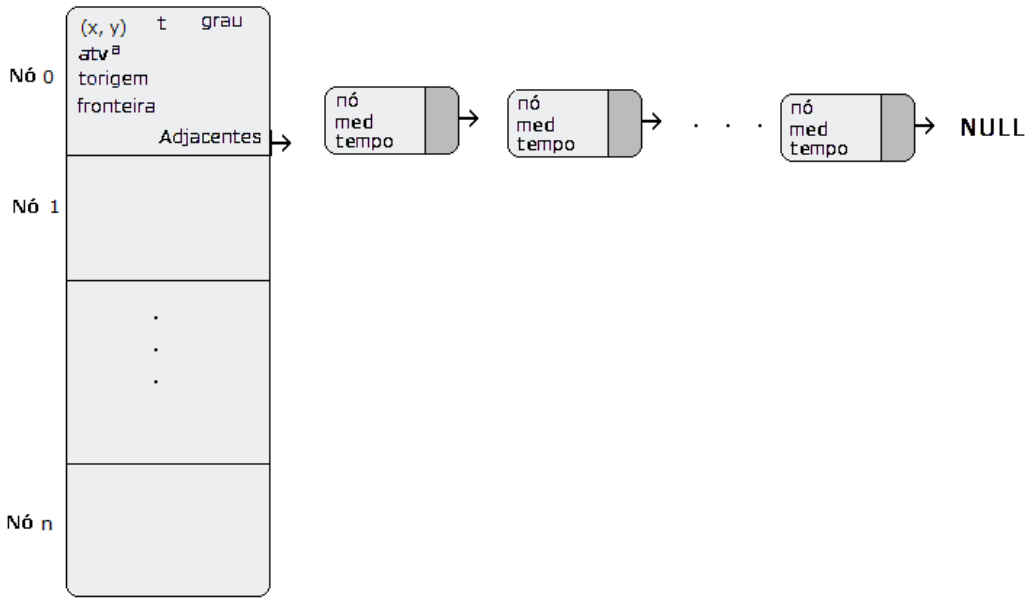


Figura 6.7: Estrutura de dados usada para armazenar as informações do grafo.

Cada posição desse vetor guada informações referentes a cada nó como: suas coordenadas (x, y), o grau do nó, o território que o nó estava alocado na configuração inicial, o território que está alocado após o reagrupamento, o valor da atividade a referente ao nó, informação se o nó é fronteira ou não e uma lista de adjacência. Nesta lista, em cada posição, são guardados o nó adjacente, o número de medidores e o tempo de leitura da aresta.

A maior parte das informações guardadas nesta estrutura são constantes, pois não se alteram no decorrer do programa (coordenadas, território original, grau, atividades e a lista de adjacência). As informações de fronteira e o território final são modificados durante a busca local que é discutida na Seção 4.4.4.

Estrutura Territórios

O objetivo do problema proposto é encontrar um reagrupamento, com as informações que foram guardadas na estrutura apresentada na seção anterior, tal que esse reagrupamento seja

o melhor possível de acordo com os critérios apresentados na Seção 3.3. Para guardar as informações pertinentes a cada agrupamento foi criada a estrutura exibida na Figura 6.8

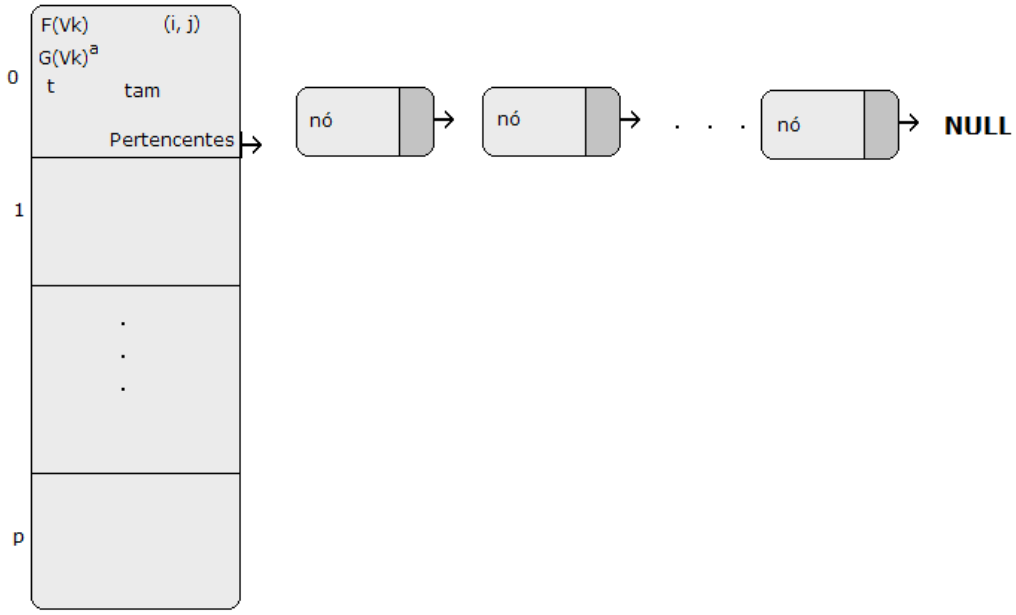


Figura 6.8: Estrutura de dados usada para armazenar as informações dos territórios.

A idéia geral da estrutura dos territórios é muito semelhante a do grafo. Porém esta é um vetor de tamanho p (com excessão da fase construtiva que pode ser um pouco maior), onde cada posição desse vetor guarda informações de um território. Em cada posição é armazenado o valor da atividade a para o território, sua medida de dispersão, juntamente com os dois nós que deram origem a ela, o tamanho do território, seu rótulo atual e uma lista encadeada de nós pertencentes ao território (Seção 6).

Os dados guardados nesta estrutura são calculados no decorrer do programa e sofrem constantes mudanças até alcançar a solução final.

Bibliografia

[Ahmadi and Osman, 2005] Ahmadi, S. and Osman, I. H. (2005). Greedy random adaptive memory programming search for the capacitated clustering problem. *European Journal of Operational Research*, 162(1):30–44.

[Chaves, 2003] Chaves, A. A. (2003). Modelagens exata e heurística para resolução do problema do caixeiro viajante com coleta de prêmios. Dissertação de mestrado, Universidade Federal de Ouro Preto.

[Chinchuluun and Pardalos, 2007] Chinchuluun, A. and Pardalos, P. M. (2007). A survey of recent developments in multiobjective optimization. *Annals of Operations Research*, 154(1):29–50.

[Christian and Andrea, 2003] Christian, B. and Andrea, R. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308.

[Cormen et al., 2002] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2002). *Algoritmos: Teoria e Prática*. Elsevier.

[França et al., 2007] França, P. M., Garcia, V. J., Morelato, A., and Usberti, F. L. (2007). Enfoque multicritério para o problema de redistritamento capacitado. *XXXIX Simpósio Brasileiro de Pesquisa Operacional - SBPO*, pages 1–12.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of Np-Completeness*. W H Freeman & Co, Gordonsville, Virginia.

[Kuhn, 1995] Kuhn, H. W. (1995). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(3):83–97.

[Laporte and Erkut, 2003] Laporte, G. andBozkaya, B. and Erkut, E. (2003). A tabu search heuristic and adaptive memory procedure for political districting. *European Journal of Operational Research*, 144(1):12–26.

[Lawler, 2001] Lawler, E. (2001). *Combinatorial Optimization: networks and matroids*. Dover Publications, Mineola, New York.

[Lobianco and Meza, 2007] Lobianco, A. T. M. and Meza, L. A. (2007). Uma comparação de métodos de solução para problemas de programação linear multiobjetivo. pages 1–14.

[Osman and Christofides, 1994] Osman, I. and Christofides, N. (1994). Capacitated clustering problems by hybrid simulated annealing and tabu search. *International Transactions in Operations Research*, 1(3):317–336.

[Pereira et al., 2007] Pereira, F. T., Figueira, J. R., Mousseau, V., and Roy, B. (2007). Multiple criteria districting problems - the public transportation network pricing system of the Paris region. *Ann Operations Research*, 154(1):69–92.

[Resende and Ribeiro, 2008] Resende, M. G. C. and Ribeiro, C. C. (2008). Greedy randomized adaptive search procedures: Advances and applications. *AT & T Labs Research Technical Report.*, pages 1–35.

[Ricca and Simeone, 2008] Ricca, F. and Simeone, B. (2008). Local search algorithms for political districting. *European Journal of Operational Research*, 189(3):1409–1426.

[Ríos-Mercado and Fernández, 2007] Ríos-Mercado, R. Z. and Fernández, E. (2007). A reactive grasp for a commercial territory design problem with multiple balancing requirements. *Computer & Operations Research*, pages 1–43.

[Sosa, 1996] Sosa, N. G. M. (1996). Heurísticas e metaheurísticas para o problema de agrupamento capacitado. Dissertação de mestrado, Universidade Estadual de Campinas.

[Tenenbaum et al., 1995] Tenenbaum, A. M., Langsam, Y., and J., A. M. (1995). *Estruturas de dados usando C*. Makron Books.

[Usberti, 2007] Usberti, F. L. (2007). Algoritmos para o Problema de Roteamento de Leituristas. Dissertação de mestrado, Universidade Estadual de Campinas.

[Vickrey, 1961] Vickrey, W. (1961). On the prevention of gerrymandering Political Science Quarterly. 76:105–110.

[Wolsey, 1998] Wolsey, L. A. (1998). *Integer Programming*. A Wiley-Interscience Publication.

[Zadeh, 1963] Zadeh, L. (1963). Optimality and non-scalar-valued performance criteria. *IEEE Transactions on Automatic Control*, 8(1):59–60.