Universidade Federal da Paraíba

Centro de Tecnologia

Departamento de Engenharia de Produção

Programa de Pós-Graduação em Engenharia de Produção

Juliana Holanda Correia

UMA FERRAMENTA WEB INTEGRADA A MÉTODOS HÍBRIDOS APLICADOS A PROBLEMAS DE LOCALIZAÇÃO

João Pessoa

2011

Juliana Holanda Correia

UMA FERRAMENTA WEB INTEGRADA A MÉTODOS HÍBRIDOS APLICADOS A PROBLEMAS DE LOCALIZAÇÃO

Dissertação apresentada ao Departamento de Engenharia de Produção da Universidade Federal da Paraíba, como parte dos requisitos para obtenção do título de Mestre em Engenharia de Produção.

Orientador: Dr. Roberto Quirino do Nascimento

João Pessoa

2011

À minha família.

AGRADECIMENTOS

Ao professor Ms. Gilberto Farias de Sousa Filho, pela grande ajuda neste projeto.

À Roberto Vieira pela ideia inicial deste projeto, pelo grande apoio e muitas leituras deste texto.

À Nádia Pinheiro, que desde o início foi meu suporte técnico.

Aos colegas do Laporte, em especial, a Gustavo, pela paciência de me ensinar um pouco de programação.

À Elisson Silvino, por me incentivar a continuar nos momentos que pensei em desistir.

Ao Tribunal Regional Eleitoral da Paraíba pelo apoio e cooperação.

À Capes pelo apoio financeiro.

Juliana Holanda Correia

RESUMO

Este trabalho apresenta um sistema computacional que, integrado com um sistema WebGIS, tem a função de otimizar os problemas de localização de medianas e cobertura. O acesso ao sistema se dá através de um navegador web e conexão com a internet e, se propõe a gerar a matriz de distâncias reais entre clientes e facilitadores. O Sistema foi aplicado ao problema de localização de pontos de coleta e transmissão, enfrentado pelo sistema eleitoral brasileiro, a fim de auxiliá-lo na tomada de decisão acerca dos melhores locais para instalação de tais pontos. O intuito do tribunal é minimizar o somatório das distâncias totais percorridas, bem como também ter a opção de minimizar a máxima distância percorrida. Para ilustrar a utilização do Sistema foi feita uma aplicação do mesmo no Tribunal Regional Eleitoral da Paraíba onde o mesmo conseguiu diminuir em, no mínimo, 23% o somatório da distância total percorrida dos locais de votação até os pontos de coleta e transmissão de votos e diminuir em 70% a distância máxima percorrida entre o local de votação e seu respectivo PCT. Neste exemplo de aplicabilidade do sistema foi tratado o problema P-mediana com a metaheurística GRASP que também foi testada em instâncias da biblioteca OR-Library e atingiu a solução ótima em mais de 62% dos casos.

Palavras-chave: WebGIS, problema de localização, geoprocessamento

ABSTRACT

This work presents a computational system that integrated with a WebGIS system, has the function to optimize the problems of facility location. System access is via a web browser and Internet connection, and aims to generate the array of actual distances between clients and facilitators. The system was applied to the problem of finding points of collection and transmission, faced by the Brazilian electoral system in order to assist in decision making about the best locations for installation of such points. The order of the court is to minimize the sum of the total distances traveled, and also have the option to minimize the maximum distance. In this example of applicability of the treaty system was the P-median problem with GRASP.

Keywords: WebGIS, facility location, GIS

Sumário

Lista de Figuras

Capítulo 1

Introdução

Para que uma empresa se mantenha competitiva no mercado é essencial que a administração das suas operações seja eficiente. A utilização de tecnologias que melhoram a sua organização e o seu gerenciamento constitui um diferencial estratégico. Dentro deste contexto, a logística tem passado por uma crescente valorização.

A logística refere-se ao processo de planejar, implementar e controlar, de maneira eficiente, o fluxo e a armazenagem de produtos, bem como os serviços de informação associados, cobrindo desde o ponto de origem até o ponto de consumo, com o objetivo de atender aos requisitos do consumidor (NOVAES, 2001).

De acordo com Ballou (1993), um dos objetivos da logística é melhorar o nível de serviço oferecido ao cliente. A logística, portanto, é um fator que pode ser utilizado como estratégia para uma organização. Sua aplicação se dá através da escolha adequada de fornecedores, passando pela organização e chegando ao cliente.

Para Neto et al. (2001), ao ser corretamente entendida e aplicada, a logística permite desenvolver estratégias para a redução de custos e o aumento do nível de serviço ofertado ao cliente. Como estas duas condições, isoladamente ou em conjunto, possibilitam o estabelecimento de diferenciais competitivos, justifica-se que este seja o caminho escolhido por um número crescente de empresas para buscar vantagens sobre a concorrência. Dentre o escopo dos problemas de logística temos os problemas de localização de facilidades, onde o termo facilidade pode representar depósitos, postos policiais, escolas, entre outros.

Modelos que buscam minimizar a distância de todos os clientes às suas respectivas facilidades são apropriados para descrever problemas do setor privado, no qual

as medidas de custo estão diretamente ligadas às distâncias envolvidas no atendimento das demandas. Também encontram-se aplicações deste modelo em problemas do setor público. Desta forma, a resolução de questões com estas características apresentam-se como uma boa base para auxiliar a tomada de decisões dentro de uma organização, seja ela do setor público ou privado.

Neste trabalho apresentamos métodos para tomada de decisão baseados na otimização de um problema de Cobertura ou de localização, tendo tais problemas as distâncias entre facilidades como dados de entrada. Surge então a necessidade de utilizar um sistema de geoprocessamento e a tecnologia WebGIS, principalmente nos casos em que não há necessidade de uma grande precisão, para obter esses dados com maior rapidez.

Para que os problemas de localização sejam resolvidos com pouca interferência do usuário, foi desenvolvido um Sistema Otimizador que, integrado com um sistema WebGIS, pretende resolver modelos genéricos dos problemas de localização de medianas ou de máxima cobertura. Como forma de exemplificar o uso do Sistema, foi feita uma aplicação do mesmo ao problema enfrentado pelo Tribunal Regional Eleitoral da Paraíba - TRE-PB. A dificuldade enfrentada pelo TRE-PB consiste em tentar minimizar o somátorio da distância total percorrida dos locais de votação ao pontos de coleta e transmissão de votos, o que iria garantir que o resultado da eleição fosse divulgado mais rápido. De acordo com o levantamento bibliográfico feito, tal problema se assemelha ao das P-medianas e foi resolvido pela metaheurística GRASP implementada no Sistema.

Este trabalho encontra-se organizado da seguinte maneira: no capítulo 2 é apresentado um breve histórico dos problemas de localização de facilidades e alguns modelos clássicos; no capítulo 3 são descritos os métodos de resolução para problemas de localização de facilidades; no capítulo 4 é apresentada a definição de um sistema de geoprocessamento e uma ferramenta WebGIS e suas principais utilizações e, também, é feita a descrição do sistema otimizador proposto; no capítulo 5 é mostrado uma aplicação do Sistema Otimizador no sistema eleitoral brasileiro, bem como os resultados computacionais apresentados pela metaheurística utilizada; no capítulo 6 é feita uma descrição detalhada da utilização do Sistema Otimizador; por fim, temos o capítulo 7 as conclusões.

Capítulo 2

Modelos de Cobertura e Localização

Grande parte dos problemas enfrentados no cotidiano das empresas pode ser caracterizado como problemas de localização de facilidades. À medida que se consegue, através das ferramentas de otimização, reduzir os custos logísticos de uma empresa, a sua margem de lucro aumenta, tornando-a cada vez mais competititva no mercado.

Para ilustrar melhor as situações que estão dentro do escopo dos problemas de localização de facilidades, vamos apresentar o seguinte questionamento: suponha que você planeja construir uma nova cadeia de lojas numa dada cidade k, e tem que identificar o potencial da loja em um certo número de vizinhança. Assume que a demanda para as lojas em cada vizinhança da cidade é conhecida. Se você quer construir exatamente n lojas, onde elas poderiam se localizar de modo que a média das distâncias para seus clientes seja mínima? Ou, se você for construir um número de lojas com o custo de construção conhecido, onde você construiria lojas para minimizar o custo de construção e a média de distância das viagens para seus clientes?

Essas questões também são conhecidas como análise de localização e fazem parte de um ramo da pesquisa operacional que está relacionada com modelos matemáticos e soluções que geram o melhor arranjo de facilidades. O termo facilidade inclui entidades como, por exemplo, portos, fábricas, armazéns, escolas, hospitais, pontos de ônibus, estações de metrô, dentre outros.

Neste capítulo será abordado um breve histórico sobre problemas de localização de facilidades e em seguida serão apresentados alguns modelos clássicos.

2.1 Histórico

A determinação do melhor local para instalação de uma unidade produtiva, comercial ou de prestação de serviços não passou a constituir matéria de estudos apenas nos anos mais recentes. Tais tipos de problemas são estudados desde o século XII, quando o matemático Evangelista Torricelli empenhou-se em estabelecer as áreas ideais para construção de mercados e hospitais (Lima Junior, 2006). No entanto, o histórico dos problemas de localização apresenta o trabalho de Alfred Weber (1868-1958) como o pioneiro da era moderna, que publicou seu trabalho em 1909. A técnica utilizada por Weber na determinação do melhor local para instalar uma indústria, considerando a minimização dos custos de transporte da matéria-prima e os custos até o mercado consumidor, consistia em procedimentos exclusivamente geométricos. (Weber, 1929, apud Okabe, Boots, Sughiara, 1995).

A partir da década de 60, com o avanço da computação e da programação matemática, este tipo de problema passou a ser amplamente focalizado. Um importante trabalho da década de 60 é o de Teitz e Bart (1968). Esses autores investigaram as possibilidades de se encontrar a mediana de um grafo ponderado, propondo então um método que obteve bons resultados em comparação com os que até então tinham sido realizados.

De acordo com Ferreira (2008), devido ao processo de reestruturação produtiva vivida na década de 70, muitas pesquisas sobre problemas de localização foram desenvolvidas. Há trabalhos como de Wesolowsky e Love (1971), Toregas et al. (1974), Walker (1974), Wagner e Falkson (1975), Davies e Thomas (1976), McAllister (1976), e o trabalho de Bach (1980).

Problemas de grande porte eram, até então, de difícil resolução. Mas, a partir das décadas de 80 e 90, com o avanço computacional ocorrido na época, tais problemas tornaram-se passíveis de resolução. Considera-se para programação linear um problema de grande porte quando o mesmo possui 10 mil variáveis e 5 mil restrições. Destacam-se os trabalhos como o de Beasley (1987) e o de Beasley e Chu (1996), visando a resolução de problemas de grande porte. Além desses, muitos outros trabalhos foram feitos a partir dessa época, em que se destacam, dentre outros: Beguin et al. (1992); Silva e Pizzolato (1993); Bezerra (1995); Rosa (1996); Lobo (1998); Lima e Gonçalves (1999); Sampaio (1999); Corrêa (2000); Colombo (2001); Smiderle (2001); Batistus (2002); Lobo (2003) e

Formigoni (2005).

2.2 Modelagem Matemática

Informalmente, um modelo matemático é uma tentativa de representar matematicamente um situação real. Suponha a seguinte situação: a prefeitura de uma cidade deseja construir 5 novos postos de saúde na zona sul, cada um com capacidade para atender 5 mil pessoas. Onde devem ser instalados estes postos de saúde para que os mesmos possam cobrir o maior número de possíveis pacientes num intervalo máximo de, por exemplo, 15 minutos? Essa é uma situação real e a pergunta norteadora para resolver da melhor maneira possível este problema é como representar essa situação através de um modelo matemático?

Para que um modelo seja uma boa representação da realidade precisamos definir alguns elementos modeladores.

- **Espaço**

  A busca de soluções para problemas de localização pode desenvolver-se em meios distintos. Modelos contínuos são aqueles em que a montagem do problema assume como área de ação um espaço $\mathbb{R}^n$, para algum valor de n; modelos discretos são aqueles em que as unidade somente podem ser instaladas em pontos previamente especificados e, finalmente, modelos em rede nos quais os possíveis locais de instalação encontram-se conectados através de ligãçoes denominadas arcos.

- **Número de unidade a serem instaladas**

  Frequentemente, o modelo objetiva estabelecer o local ideal para instalação de $p \in \mathbb{N}$ facilidades. De acordo com Lima Junior (2006), quando $p > 1$, pode ser indispensável considerar, na montagem do modelo, análise relativa a interações entre as múltiplas unidades a serem postas em funcionamento, principalmente quando se pretende atingir a satisfação plena da demanda dos clientes, levando-se em conta a capacidade de atendimento de cada uma delas.

- **Medida de capacidade de atendimento de cada unidade**

  Na modelagem de um problema de localização de facilidades pode-se exigir a prévia determinação de um potencial de atendimento aos clientes. Quando esta limitação

existir, o modelo será dito capacitado, caso contrário, o modelo será classificado como sem limitação de capacidade.

- **Clientes**

  Para definir um modelo de um problema de localização de facilidades é importante saber se os clientes estão uniformemente distribuídos sobre a região a ser estudada ou se apresentam-se em pontos específicos do espaço. Também é interessante conhecer a demanda de cada cliente.

- **Objetivo**

  Normalmente o objetivo dos problemas de localização se refere a minimização dos custos de produção/operação ou mesmo a maximização dos lucros. Tornam-se cada vez mais comuns as situações nas quais se faz necessária a definição de áreas indesejáveis, capazes de produzir efeitos danosos às pessoas e ao meio ambiente, por exemplo, usinas termonucleares.

  Segundo Lima Junior (2006), pesquisas recentes, abrangendo esse tipo de problema de localização, levaram à utilização de múltiplos objetivos nas representações análiticas correspondentes, razão pela qual passaram a ser utilizados modelos pretendendo, simultaneamente, minimizar os custos de instalação das unidades e maximizar as distâncias entre esta e os indivíduos atingidos pelo seu funcionamento.

É óbvio que, dependendo da situação real a ser modelada, alguns desses elementos não serão utilizados na montagem do modelo, podendo ocorrer, também, que outros não citados venham integrar o processo descritivo da situação sob estudo.

2.3 Modelos Clássicos

Os problemas de localização estão divididos em problemas de máxima cobertura e problemas de localização de facilidades. Em ambos, decisões são tomadas sobre onde localizar facilidades, considerando os outros centros como clientes que devem ser servidos, de forma a otimizar um dado critério. Dentre estes problemas podemos citar o Problema de Localização de Máxima Cobertura (Church e Revelle, 1974) que tem como objetivo localizar p facilidades de modo que a máxima população possível seja coberta dentro

da distância de serviço. Uma área (ponto) de demanda é considerada coberta se está dentro da distância de serviço de pelo menos uma facilidade. Modelos de cobertura são frequentemente utilizados por órgãos públicos para a localização de serviços emergenciais ou não-emergenciais.

No serviço público não emergencial podemos citar como modelo de um problema de máxima cobertura a seguinte situação: a administração de uma cidade gostaria de ter alunos a uma curta distância de suas respectivas escolas, pois tal situação melhoraria o trânsito da cidade uma vez que diminuiria o número de pessoas percorrendo largas distâncias para chegar ao local desejado. No entanto, esses problemas de planejamento tem um número máximo de unidades que deve ser respeitado. Como dito anteriomente, o objetivo do Problema de Localização de Máxima Cobertura (PLMC) é localizar um número pré-determinado de instalações, p, de modo a maximizar a abrangência das facilidades. Assim, o PLMC assume que pode não haver instalações suficientes para cobrir todos os nós de demanda. Se nem todos os nós podem ser abrangidos, o modelo visa ao regime de localização que abrange a maior parte da demanda.

O PLMC pode ser formulado como a seguir:

Maximize:

$$\sum_{i \in U} h_i z_i \tag{2.1}$$

Sujeito a

$$\sum_{j \in N_i} y_j \geq z_i, \ \forall i \in U \tag{2.2}$$

$$\sum_{j \in F} y_j = p \tag{2.3}$$

$$y_j \in \{0, 1\}, \ \forall j \in F \tag{2.4}$$

$$z_i \in \{0, 1\}, \ \forall i \in U \tag{2.5}$$

onde,

$F = \{j; j = 1, 2, \ldots, m\}$, representa o conjunto de locais elegíveis para instalação de unidades;

$U = \{i; i = 1, 2, \ldots, n\}$, representa o conjunto dos pontos que geram demanda. Normalmente $F \subseteq U$;

D_c = distância de cobertura;

d_{ij} = distância entre a demanda do nó i e o candidato j;

$N_i = \{j | d_{ij} \leq D_c\}$;

= o conjunto de todos os locais candidatos que podem cobrir ponto de demanda i

$y_j = 1$ se o vértice j é uma facilidade e $y_j = 0$, caso contrário;

h_i = demanda do nó i;

p = número de facilidades para localizar;

$z_i = 1$ se a demanda do nó i é coberta e $z_i = 0$, caso contrário.

A função objetivo (2.1) maximiza a demanda total coberta. A restrição (2.2) assegura que a demanda do nó i só estará coberta quando o mesmo estiver dentro do raio de cobertura de uma facilidade y_j. A restrição (2.3) limita o número de facilidades a serem alocadas. O conjunto de restrições (2.4) e (2.5) reflete a natureza binária da decisão de localizar facilidades e a maneira de cobrir os nós, respectivamente.

Nos problemas de localização no setor público, de caráter emergencial, busca-se em geral prover cobertura das áreas de demanda. De acordo com (Galvão, 99), a noção de cobertura implica a definição de uma distância (tempo) de serviço, que é a distância (tempo) crítica além da qual a área de demanda é considerada não coberta. Uma área de demanda é, portanto, considerada coberta se está localizada na área de cobertura de pelo menos uma das facilidades definida pela distância crítica, independentemente de a facilidade (ou servidor) estar ou não disponível quando o serviço é solicitado.

Considere-se, por exemplo, o problema de localizar serviços de atendimento de emergência por ambulâncias ou por estações do corpo de bombeiros em uma dada região, de tal modo que toda a população da região esteja a menos de 10 quilômetros de pelo menos uma das facilidades. Neste caso, 10 quilômetros definem a distância crítica e o problema consiste na determinação do número de facilidades e de sua localização na região em consideração, de tal forma que cada área de demanda esteja a menos de 10 quilômetros de pelo menos uma das facilidades localizadas. Logo, o objetivo é localizar um número pré-especificado de facilidades, tal que a máxima população possível de uma dada região esteja coberta a menos de uma distância crítica D_m pré-definida.

Matematicamente o modelo de localização de caráter emergencial difere do

modelo de localização de caráter não emergencial pelo acréscimo da seguinte restrição mandatória:

$$\sum_{j \in M_i} y_j \geq 1, \ \forall i \in U \tag{2.6}$$

onde,

$$M_i = \{j; d_{ij} < D_m\} \tag{2.7}$$

e D_m representa a máxima distância de cobertura.

Outro problema é o Problema das P-medianas (Revelle & Swain, 1970), que pode ser capacitado ou não. Tal problema é combinatorial e NP-*hard*, ou seja, é altamente complexo do ponto de vista computacional e sua resolução por métodos exatos já foi feita por (Christofides and Beasley, 1982; Galvão, 1980; Galvão and Raggi, 1989; Hanjoul and Peeters, 1985; Mirchandani et al, 1985) entre outros, porém este método de resolução limita o tamanho das instâncias possíveis de serem resolvidas. Com o intuito de encontrar uma solução satisfatória, várias heurísticas foram aplicadas ao problema das P-medianas, veja (Maranzana, 1964; Teitz and Bart, 1968).

Informalmente, o problema das P-medianas não capacitado, assume que não existe limite superior para ser satisfeito pelos pontos de demanda, então temos apenas que minimizar a distância do cliente para o ponto de demanda mais próximo. Já no problema capacitado, existe uma demanda máxima que não pode ser ultrapassada. Todas elas tem que oferecer serviços aos clientes de forma a minimizar a distância de todos os clientes às suas respectivas facilidades enquanto a capacidade dos pontos de demanda é atendida.

O problema das P-medianas é um problema de localização-alocação que visa selecionar p vértices, em uma rede conectada por caminhos, para a instalação de facilidades de forma a minimizar o somatório das distâncias entre os vértices de demanda (clientes) e a facilidade mais próxima e, de acordo com Resende (2003), o modelo não capacitado pode ser formulado como o seguinte problema de otimização.

Dado um conjunto F de m potenciais facilidades, um conjunto U de clientes, uma função distância $d : U \times F \rightarrow R$, e uma constante $p < m$, determinar p facilidades de modo que o somatório das distâncias dos clientes para as facilidades seja mínimo. O modelo (1) do problema está descrito a seguir:

Minimizar:

$$\sum_{i \in U} \sum_{j \in F} d_{ij}\ x_{ij} \tag{2.8}$$

Sujeito a:

$$\sum_{j \in F} x_{ij} = 1,\ \forall i \in U \tag{2.9}$$

$$x_{ij} \leq y_j,\ \forall i \in U;\ \forall j \in F \tag{2.10}$$

$$\sum_{j \in F} y_j = p \tag{2.11}$$

$$x_{ij} \in \{0, 1\},\ \forall i \in U\ e\ y_j \in \{0, 1\}\ \forall j \in F \tag{2.12}$$

onde $[x_{ij}]_{n \times m}$ é uma matriz de alocação, com $x_{ij} = 1$ se o vértice i está alocado à mediana j, e $x_{ij} = 0$, caso contrário.

A função objetivo (2.8) informa a distância total correspondente a uma solução de localização-alocação. O conjunto de restrições (2.9) e (2.10) garantem que cada vértice i seja alocado a apenas um vértice j, que deve ser uma mediana. A restrição (2.11) determina o número de medianas a serem localizadas e o conjunto de restrições (2.12) impõe a condição de integralidade sobre as variáveis.

O objetivo é determinar $p < n$ vértices (medianas) de modo a minimizar a soma das distâncias dos outros vértices do grafo à mediana mais próxima. A matriz de distâncias $D = [d_{ij}]_{n \times m}$ em cada par de vértices deve ser conhecida.

Outra formulação para o problema da P-mediana é dada ao substituirmos a equação (10) pela equação

$$\sum_{i \in U} x_{ij} \leq n \cdot y_j, \forall\ j \in F \tag{2.13}$$

onde n é o número total de clientes. Com essa modificação passamos a ter n restrições e no modelo original temos $n \cdot m$ restrições. Porém, enquanto este conjunto de restrições reduz substancialmente o tamanho do problema, ao resolvê-lo usando a programação linear, sem qualquer exigência de variáveis inteiras, produziremos quase todos valores fracionários para y_j.

Por outro lado, o conjunto de restrições (2.10) torna o problema muito grande mesmo eles sejam relativamente pequenos, pois tal conjunto de restrições percorre todo o conjunto dos clientes e das facilidades. No entanto, quando resolvemos o problema de p-medianas em sua forma estendida usando relaxação de programação linear, a maioria das soluções são inteiras. Por este motivo adotaremos no modelo das P-medianas o conjunto de restrições (10).

Quando existe restrições de capacidade, a formulação matemática do problema das P-medianas sofre a seguinte alteração na função objetivo:

Minimizar:

$$\sum_{i \in U} \sum_{j \in F} a_i \; d_{ij} \; x_{ij} \tag{2.14}$$

onde a_i representa a demanda do ponto i.

De acordo com Drezner (2001), em alguns problemas de localização a distância máxima existe a priori. Por exemplo, se escolas forem localizadas de tal forma que os seus estudantes residam a no máximo 2km da escola, elas poderiam se deslocar sem gastar com trasporte. No setor privado, alguns negocios tem garantia de serviço dentro de um tempo pré-determinado (exemplo, 20 min para entregar uma pizza). No primeiro caso, a prefeitura de uma cidade pode querer encontrar locais para construir escolas de modo que minimizem o número de estudantes que devam gastar com transporte. Neste último exemplo, uma cadeia de pizza pode querer localizar seus pontos de venda para maximizar o número de potenciais clientes atendidos no prazo máximo de 20 minutos de um dos estabelecimentos.

Para resolver esses tipos de problemas, podemos adaptar o modelo da P-mediana e criar o modelo P-mediana^{max}, onde o objetivo é minimizar a máxima distância percorrida por cada cliente a sua facilidade mais próxima, dado um número de facilidade (p) fixo.

Minimizar

$$D_{max} \tag{2.15}$$

Sujeito a

$$\sum x_{ij} = 1 \tag{2.16}$$

$$x_{ij} \leq y_j \tag{2.17}$$

$$\sum y_j = p \tag{2.18}$$

$$D_{max} \geq \sum_{j=1}^{n} x_{ij} D_{ij}, \ \forall i \in \mathbb{N} \tag{2.19}$$

onde a equação (2.19) garante que D_{max} não irá assumir valor zero. Para ilustração, podemos utilizar o seguinte exemplo: dado um número fixo de escolas (p), o objetivo é minimizar a distância máxima (D_{max}) percorrida por cada professor ao seu local de trabalho.

Neste trabalho será criado um Sistema Otimizador de localização que tem o objetivo de determinar os melhores pontos de localização para as facilidades dos problemas acima mencionados. Para cada modelo matemático exposto nesta seção temos como dado de entrada a matriz de distâncias entre os clientes e os facilitadores. Essa matriz de distâncias real será obtida através da integração do Sistema Otimizador com um sistema georeferenciado.

Capítulo 3

Métodos de Resolução para Problemas de Localização de Facilidades

Os problemas citados na seção 2.3 podem ser resolvidos através de métodos exatos. Para os problemas lineares contínuos, o método Simplex, desenvolvido por Dantzig em 1947 tem sido o método de referência até hoje; uma outra possibilidade é a utilização de algoritmos de pontos interiores, ou através de metaheurísticas. Porém, as abordagens exatas devem ser utilizadas para resolver problemas de pequenas dimensões, pois para os demais (problemas maiores), elas demandariam um tempo exponencial de processamento.

Os métodos mais usados para resolver problemas de localização de facilidades são heurísticas e metaheurísticas, que são algoritmos rápidos e determinam soluções aproximadas de excelente qualidade (Resende & Werneck, 2006; Mladenovic et al., 2007).

3.1 Métodos exatos

Os métodos exatos são aqueles que possibilitam a obtenção da solução ótima para o problema, porém sua elevada complexidade permite apenas a solução de problemas pequenos.

3.1.1 *Branch and Bound*

Utilizado em problemas de programação inteira e na programação de atividades, o método *branch and bound*, proposto por (Land e Doig, 1960), consiste em percorrer uma árvore de enumeração que particiona o conjunto solução do problema, em subconjuntos disjuntos, podando-o sempre que encontrar um ramo não promissor ou inviável. Veja, Ishida (2010)

Caso a árvore de numeração fique muito grande, utilizam-se limitantes para podar ramos da árvore que não contém a solução ótima. Para a programação inteira, cada nó da árvore é ramificado enquanto houver valores fracionários em sua solução. Se existir várias variáveis não inteiras após a relaxação linear deve-se selecionar uma para definir a divisão.

Segundo Taha (2007), para a etapa de ramificação, devemos selecionar uma das variáveis inteiras x_j, cujo valor ótimo x_j^* na solução do problema seja não inteiro. Em seguida, elimina-se a região

$$[x_j^*] < x_j < [x_j^*] + 1 \tag{3.1}$$

(na qual $[v]$ define o maior inteiro $\leq v$), criando dois subproblemas de programação linear que correspondem a

$$x_j \leq [x_j^*] \; e \; x_j \geq [x_j^*] + 1 \tag{3.2}$$

Outra forma é selecionar a variável não inteira que possui maior coeficiente na função objetivo.

A seguir ilustramos o funcionamento do algoritmo

3.1.2 *Branch and Cut*

Algoritmos baseados na estratégia *branch and cut* têm se mostrado muito eficazes na resolução de instâncias reais para problemas difíceis de otimização combinatória.

No método branch-and-cut nós dividimos o problema em subproblemas, resolvendo-os de modo a encontrar a solução ótima. Tais divisões são feitas através de planos de cortes identificados ao longo da árvore de enumeração gerada pela execução do *branch and bound*.

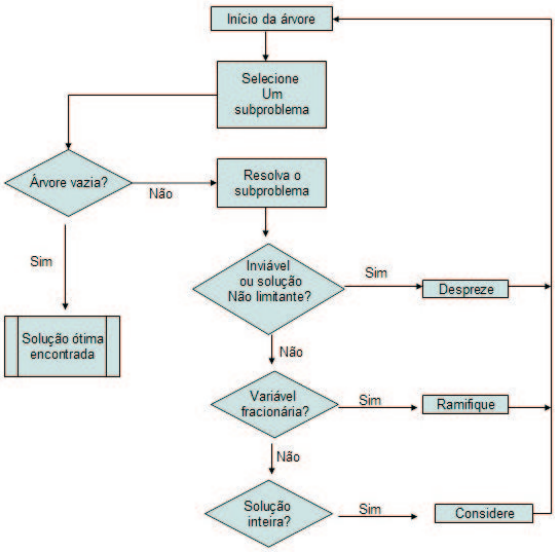


Figura 3.1: Algoritmo Branch and Bound

Esses planos de cortes são desigualdades válidas violadas por $\tilde{x}$, solução do problema proveniente da relaxação linear do problema original, a qual é obtida a cada nó da árvore. Depois de identificadas, essas desigualdades são incorporadas à formulação do problema, o que nos ajuda a obter um limitante dual melhor que o disponível. (Cavalcante, 2008)

O algoritmo *branch and cut* se diferencia do algoritmo *branch and bound*, pois o mesmo não precisa de tanto esforço computacional em cada nó da árvore, enquanto o *branch and bound* privilegia uma reotimização rápida executada a cada nó da árvore.

3.1.3 Relaxação Lagrangeana

A relaxação Lagrangeana tem sido aplicada na solução de vários problemas de otimização combinatória, incluindo o problema do caixeiro viajante, problemas de localização, de recobrimento e de particionamento de conjuntos e problemas de roteamento. Algumas utilizações deste método podem ser vistas em (Sridharan, 1995; Bazaraa, 1993; Galvão et al. (2000, 2002)).

Experiências com relaxação Lagrangeana indicam, com razoável custo computacional, bons limitantes superiores, no caso de problemas de maximização, pois a solução ótima do problema Lagrangeano é um limite superior para o valor ótimo do problema original. Tal relaxação é obtida multiplicando um conjunto de restrições de um problema inteiro (PI) por um vetor λ de multiplicadores de Lagrange, adicionando-se o

produto a função objetivo. Nesse sentido, a relaxação Lagrangeana também é utilizada, nos casos de problemas de grande porte, para auxiliar a fazer os cortes da árvore gerada pelo algoritmo *branch-and-bound*.

Considere o (PI)

Maximizar

$$v = c^t x$$

Sujeito a:

$$Dx \leq d$$

$$x \in X$$

onde, $X = \{x \in \mathbb{Z}_+^n : Ax \leq b\}$.

Nemhauser & Wolsey (1988) definem uma relaxação do problema (PI) como um problema de otimização, (PI(λ)), que possui as seguintes propriedades: (i) o conjunto de soluções viáveis de (PI) é um subconjunto das soluções viáveis de (PI(λ)), e (ii) o valor da função objetivo do problema (PI) não é maior que o correspondente valor de (PI(λ)), isto é, $v(PI(\lambda)) \geq v(PI)$ (para problemas de maximização).

Agora que já definimos o que é uma relaxação, considere o seguinte problema PI(λ).

Maximizar

$$v(\lambda) = c^t x + \lambda^t (d - Dx)$$

Sujeito a:

$$x \in X$$

Proposição 3.1 *O problema PI(λ) é uma relaxação do problema PI* $\forall \lambda \geq 0$

Prova:

I) A região viável do PI(λ) é maior ou igual a região viável do PI pois $\{x \in X : Dx \leq d\} \subseteq X$

II) O valor da função objetivo do PI(λ) é maior ou igual ao valor da função objetivo do PI para todas as soluções viáveis do PI pois $\lambda \geq 0$ e se a solução for viável (d-Dx). Logo, $c^t x \leq c^t x + \lambda(d - Dx)$

O problema PI(λ) é uma relaxação de PI com parâmetro λ. Como PI(λ) é uma relaxação de PI, $v(\lambda) \geq v$ e nós obtemos um limite superior para o valor ótimo de PI. Para encontrar o valor ótimo de PI(λ) que mais se aproxime do valor ótimo de PI, ou seja, o menor valor de PI(λ), devemos resolver o Lagrangeano dual.

$$W_{LD} = minimizar\ \{v(\lambda) : \lambda \geq 0\}$$

Resolver a relaxação lagrangeana PI(λ) pode, algumas vezes, levar a solução ótima do problema original PI.

Proposição 3.2 *Se* $\lambda \geq 0$,

I) $x(\lambda)$ *é solução ótima para PI(*λ*)*

II) $Dx(\lambda) \leq d$

III) $(Dx(\lambda))_i = d_i$ *sempre que* $\lambda_i \geq 0$, *então* $x(\lambda)$ *é ótimo para PI.*

Prova:

Por I $W_{LD} \leq v(\lambda) = c^t x(\lambda) + \lambda(d - Dx(\lambda))$. Por III $c^t x(\lambda) + \lambda(d - Dx(\lambda)) = c^t x(\lambda)$.

Por II $x(\lambda)$ é viável em PI enrão $c^t x(\lambda) \leq v$. Assim $W_{LD} \leq c^t x(\lambda) + \lambda(d - Dx) = c^t x(\lambda) \leq z$. Por outro lado, pelo Teorema da Dualidade fraca, $W_{LD} \geq v$. Portanto, $W_{LD} = v$ e $x(\lambda)$ é ótimo em PI.

Resolvendo assim o problema dual fornece um limite superior que pode ser usado no contexto do procedimento *branch-and-bound*.

Vamos mostrar uma aplicação desta abordagem no problema de localização de facilidades.

$$v = max \sum_{i \in U} \sum_{j \in F} c_{ij}\ x_{ij} - \sum_{j \in F} f_j\ y_j \tag{3.3}$$

Sujeito a

$$\sum_{j \in F} x_{ij} = 1,\ \forall i \in U \tag{3.4}$$

$$x_{ij} \leq y_j,\ \forall i \in U,\ \forall j \in F \tag{3.5}$$

$$x \in \mathbb{R}^{|U| \times |F|},\ y \in B^F \tag{3.6}$$

onde, f_j é o custo de uma unidade no local j e c_{ij} é o custo de satisfação da demanda do cliente i pela unidade instalada no local j.

Dualizando as restrições de demanda, temos $\lambda(1 - \sum_{j \in F} x_{ij})$

Maximizar:

$$v(\lambda) = \sum_{i \in U} \sum_{j \in F} (c_{ij} - \lambda_i)\ x_{ij} - \sum_{j \in F} f_j\ y_j + \sum_{i \in U} \lambda_i \tag{3.7}$$

Sujeito a:

$$x_{ij} - y_j \leq 0 \tag{3.8}$$

$$x \in \mathbb{R}^{|U| \times |F|},\ y \in B^F \tag{3.9}$$

O problema acima se divide em um subproblema para cada localidade. Assim, $v(\lambda) = \sum_{j \in F} v_j(\lambda) + \sum_{i \in U} \lambda_i$, onde

$$v_j(\lambda) = maximizar\ \sum_{i \in U} \sum_{j \in F} (c_{ij} - \lambda_i)\ x_{ij} - f_j y_j \tag{3.10}$$

Sujeito a:

$$x_{ij} - y_j \leq 0 \tag{3.11}$$

$$x_{ij} \geq 0 \tag{3.12}$$

O problema acima é facilmente resolvido por inspeção. Se $y_j = 0$, então $x_{ij} = 0$ para todo i e o valor da função objetivo é zero. Se $y_j = 1$, todos os clientes que são rentáveis são servidos, ou seja, aqueles com $c_{ij} - \lambda_i > 0$. O valor da função objetivo é então $\sum_{i \in U} max[c_{ij} - \lambda_i, 0] - f_{ij}$. Assim $v(\lambda) = max\{0, \sum_{i \in U} max[c_{ij} - \lambda_i, 0] - f_{ij}\}$

3.2 Heurísticas

As técnicas heurísticas, que são uma consequencia da intuição aplicada ao processo de resolução de problemas, são usadas, por exemplo, nos problemas em que a complexidade do algoritmo para encontrar uma solução cresce de forma exponencial.

Embora as tais técnicas não garantam uma solução ótima para o problema, elas frequentemente nos dão resultados muito próximos do ótimo e geralmente se enquadram dentro dos seguintes grupos:

- heurísticas de construção, são aquelas que selecionam sequencialmente elementos de um conjunto, eventualmente descartando alguns já selecionados, de tal forma que ao final se obtenha uma solução viável para o problema;
- busca local, são aquelas que partem de uma solução inicial viável obtida através de um método construtivo e vai melhorando sucessivamente a solução corrente através de uma busca na sua vizinhança e para assim que um ótimo local for encontrado;
- metaheurísticas é uma ferramenta algoritmica geral que pode ser aplicada a diferentes problemas de otimização com modificações relativamente pequenas, para torná-la adaptável a um problema específico.

Alguns exemplos de metaheurísticas são: *simulated annealing*, busca tabu, algoritmos genéticos e GRASP.

3.2.1 Algoritmos Genéticos

Os algoritmos genéticos são metaheuríticas baseadas na analogia entre o processo de evolução natural nos quais as variáveis são representadas como genes em um cromossomo (sequência de bits) e cada cromossomo corresponde a um ponto no espaço de soluções do problema de otimização.

Durante a evolução, as populações evoluem de acordo com os princípios da seleção natural e de "sobrevivência dos mais adaptados". Desta forma, os indivíduos que obtiveram sucesso no processo de adaptação ao seu meio ambiente terão maior chance de sobreviver e reproduzir. Assim, as espécies evoluem tornando-se cada vez mais adaptadas ao meio.

O processo de solução adotado pelos algoritmos genéticos consiste em gerar novos cromossomos a partir da população corrente e incluí-los na população enquanto outros são excluídos. Para isso, selecionamos os cromossomos pais e rea-lizamos uma operação de *crossover*, que é uma combinação simples das representações de cada cromossomo.

Uma das vantagens de um algoritmo genético é a simplificação que eles permitem na formulação e solução de problemas de otimização. Algoritmos genéticos simples normalmente trabalham com descrições de entrada formadas por cadeias de bits de tamanho fixo. Outros tipos de algoritmos genéticos podem trabalhar com cadeias de bits de tamanho variável, como por exemplo os algoritmos genéticos usados para Programação Genética. Algoritmos genéticos possuem um paralelismo implícito decorrente da avaliação independente de cada uma dessas cadeias de bits, ou seja, pode-se avaliar a viabilidade de um conjunto de parâmetros para a solução do problema de otimização em questão (veja, Lorena et al, 2006).

Uma implementação de um algoritmo genético começa com uma população aleatória de cromossomos. Essas estruturas são, então, avaliadas e associadas a uma probabilidade de reprodução de tal forma que as maiores probabilidades são associadas aos cromossomos que representam uma melhor solução para o problema de otimização do que aqueles que representam uma solução pior. Pode-se estabelecer critérios de parada através da estabilização da população, da impossibilidade de melhor solução ou pelo número de gerações.

3.2.2 Busca tabu

A metaheurística busca tabu (*tabu search*) originou-se de uma solução para problemas de programação inteira e posteriormente foi generalizada para uso em problemas de otimização combinatória. A caracteristica principal dessa metaheuristica é a existência de uma memória de curto prazo que armazena o histórico da evolução do processo de busca realizado pelo algoritmo. Tais movimentos são armazenados numa estrutura de dados (lista tabu) por um determinado número de iterações, evitando assim que uma solução seja visitada novamente. Porém, segundo Gonçalves (2005), se a solução obtida através de um certo movimento, atender um critério de aspiração previamente definido, o movimento pode ser aceito mesmo pertencendo a lista tabu. Tais critérios de aspiração podem modificar o estado tabu de uma solução, um exemplo de critério bastante utilizado é admitir soluções que são melhores que a melhor solução conhecida até o momento.

Outra importante característica dessa metaheuristica é que ela dispensa uma solução inicial elaborada, porém a solução inicial deve fazer parte do conjunto de soluções

possíveis do espaço. A cada iteração é gerado um subconjunto V de soluções vizinhas a solução corrente s. Esse conjunto é avaliado através de uma função de avaliação f. A solução que oferecer o menor custo, de acordo com f, será solucionada, tornando-se a solução corrente.

A condição de parada do método geralmente é estabelecida através de um número máximo de iterações sem melhora, ou quando o algoritmo atingir um valor melhor que o valor mínimo conhecido para função de avaliação.

3.2.3 *Simulated Annealing*

Simulated Annealing, ou Recozimento Simulado, é uma metaheurística que se fundamenta numa analogia com um processo de aquecimento de um sólido até o seu ponto de fusão, seguindo de arrefecimento progressivo até que se solidifique novamente obtendo estados mínimos de energia.

O algoritmo começa a busca a partir de uma solução inicial qualquer, s, e vai substituindo a solução inicial por uma solução (s') na vizinhança do espaço de soluções, escolhida de acordo com uma função objetivo e com uma variável T. A variação da função objetivo é testada a cada geração de um vizinho através do seguinte cálculo:

$$\Delta = f(s') - f(s)$$

- Se $\Delta < 0$, s' passa a ser a nova solução corrente;
- Se $\Delta > 0$, o vizinho será aceito com uma probabilidade $\exp^{\frac{-\Delta}{T}}$

Esse procedimento é repetido até que T seja tão pequeno que não aceite novos movimentos.

Uma das principais vantagens do *Simulated Annealing* sobre as outras metaheurísticas é a possibilidade de evitar mínimos locais pois o algoritmo emprega uma busca aleatória, que pode aceitar vizinhos com uma energia mais elevada. Porém, a probabilidade de aceitar um vizinho com energia maior decresce com o tempo, que é controlado através do parâmetro T. Mas, apesar da possibilidade teórica de convergir para solução ótima, a velocidade de diminuição da temperatura implicaria que o algoritmo terá que visitasse um número exponencial de soluções distintas.

3.2.4 GRASP

Um GRASP (Greedy Randomized Adaptive Search Procedure) é um procedimento iterativo que combina várias propriedades favoráveis de outras heurísticas (veja, Resende, 2001). Mais especificamente, cada iteração do GRASP consiste de dois estágios: uma fase de construção da solução e uma fase de busca local. Na fase de construção, uma solução de boa qualidade é gerada através de uma mistura de aleatoriedade e gulosidade. A partir dessa solução, uma busca local é aplicada com o objetivo de melhorar essa solução. Em cada iteração uma solução é encontrada, sendo a melhor delas considerada a solução final.

Na fase de construção de uma solução, iniciamos com um conjunto vazio, que iterativamente recebe um elemento, até formar uma solução viável. Nesta etapa dois aspectos são analisados a cada iteração: a aleatoriedade e a adaptação. A aleatoriedade deve-se ao fato de que o próximo elemento a ser escolhido para compor a solução partirá de uma lista, denominada Lista Restrita de Candidatos (LRC), que contém os α melhores elementos candidatos segundo o critério guloso especificado. O valor de α é um parâmetro do algoritmo. Por outro lado, o aspecto da adaptação do processo mostra-se evidente, à medida que se escolhe um elemento que irá compor a solução inicial, pois os próximos elementos a serem inseridos em LRC dependerão dos já incluídos na solução inicial.

As soluções obtidas na fase de construção do GRASP não são garantidas como ótimos locais, considerando uma dada vizinhança (diz-se que uma solução s é localmente ótima, se não existe uma solução melhor na vizinhança de s). Portanto, o emprego da segunda fase do GRASP é feito com o intuito de se melhorar a solução obtida na fase de construção.

Um algoritmo de busca local é utilizado nesta segunda fase para, sucessivamente, substituir a solução atual por uma solução melhor, encontrada na vizinhança da solução atual. O algoritmo termina quando nenhuma solução melhor é encontrada na vizinhança da solução atual ou após algum outro critério de parada ter sido atingido.

Esta metaheurística será mais detalhada na seção 6.2, quando será descrita sua implementação para resolução do problema das P-medianas.

3.3 Métodos Híbridos

Atualmente, os métodos híbridos tem recebido gradativa atenção na comunidade acadêmica pois, nos últimos anos um grande número de algoritmos reportados da literatura não seguem puramente os conceitos de uma única metaheurística clássica, mas combinam várias ideias, algumas vezes até fora do campo das metaheurísticas. A motivação por trás da hibridização de várias metaheurísticas normalmente é obter sistemas com performace melhor, que explorem e unam vantagens das estratégias puras aplicadas individualmente.

Podemos classificar tal método de acordo com:

a) seu nível de hibridização:

- **Hight-level**: mantém a identidade individual dos algoritmos originais;
- **Low-level**: algoritmos dependem fortemente um do outros, ou seja, os componentes individuais dos algoritmos são trocados.

b) a ordem de execução das metaheurísticas associadas:

- **Sequencial**: um algoritmo é executado depois o outro, e só é passada informação em uma direção;
- **Intercalado**: a cada iteração de um algoritmo, o outro é executado;
- **Paralelo**: os algoritmos são executados em paralelo e a informação pode ser passada em qualquer direção.

c) a estratégia de controle:

- **Integrada**: um algoritmo é considerado um subordinado, ou seja, componente embutido de um outro algoritmo;
- **Colaborada**: algoritmos trocam informação entre si, mas não são partes um do outro.

Para validar a arquitetura proposta para o Sistema Otimizador, foram implementados dois Componentes Otimizadores paras as classes de problema P-Mediana e P-Mediana^{max} utilizando a metaheurística GRASP como classe de algoritmo de resolução dos problemas. A escolha desta metaheurística deve-se ao seu emprego com êxito em

outros trabalhos que tratam do mesmo problema (Resende, 2002; Resende, 2003) e pelo fato da mesma gerar uma nova solução a cada fase de construção, facilitando assim a aplicação de métodos de resolução exatos.

Capítulo 4

Geoprocessamento e o Sistema Otimizador

4.1 Sistema de Geoprocessamento

O Sistema de Geoprocessamento GIS (Geographic Information System) é uma ferramenta utilizada para representar a realidade do espaço geográfico em um ambiente computacional. São sistemas para armazenar, analisar, criar e modificar dados geográficos.

As principais utilidades de um GIS estão destacadas a seguir:

1. ferramenta para produção de mapas;
2. suporte para análise espacial de fenômenos;
3. banco de dados geográficos, com função de armazenamento e recuperação de informação espacial.

Neste trabalho foi dada ênfase na função de armazenamento e recuperação espacial de dados geográficos.

Um GIS compreende diversas ferramentas integradas, dentre elas destacamos a Cartografia Digital e a Geocodificação.

Cartografia Digital

Segundo FILHO (2000, p.4) um sistema de Cartografia Digital pode ser compreendido como um conjunto de ferramentas, incluindo programas e equipamentos,

orientados para a conversão do meio digital, armazenamento e visualização de dados espaciais. Um sistema de Cartografia Digital tem como ênfase a produção final de mapas.

Geocodificação

Segundo ESRI (2007) a geocodificação é um procedimento computacional que transforma endereços ou referências a lugares em coordenadas pontuais de acordo com um sistema de referência utilizado. O resultado retornado ao usuário é uma estimativa de onde deveria estar localizado o endereço entregue para consulta.

Para tanto, a base de referência pode ser composta de diversas maneiras, suportando diversos tipos de endereçamento, desde sistemas de endereço métricos compostos por geometrias lineares, como sistemas de endereçamento regionais compostos por polígonos (ZIP codes nos Estados Unidos).

No Brasil, o sistema de endereçamento é o sistema métrico, muito utilizado no mundo todo, em que cada logradouro possui um nome e numeração (do lado esquerdo e direito), sendo esta proporcional à distância percorrida desde seu ponto de início. Este sistema de endereçamento é possível de ser geocodificado.

4.2 WebGIS

De acordo com Painho, et al (2001) um WebGIS é simplesmente a soma das funcionalidades de um GIS para web com o benefício de não necessitar de um software. WebGIS é uma tecnologia de disponibilização de dados geográficos e mapas iterativos para consulta e manipulação através de um protocolo remoto, como o HTTP (Hypertext Transfer Protocol) na internet ou intranet.

Neste trabalho abordaremos um WebGIS bastante popular o Google Maps (GOOGLE, 2010), seu serviço é provido de forma aberta a partir de uma API de programação, Google Maps API, que permite ao usuário acessar os recursos básicos de geoprocessamento.

O Google Maps API é um serviço, via internet, que fornece diversos utilitários para manipular mapas, dentre estes serviços serão utilizamos três que servirão para o propósito deste trabalho: a exibição de mapas (Cartografia Digital), a localização de pontos no mapa pelo endereço real ou pelo par latitude/longitude fornecidos; e a definição das distâncias entre quaisquer dois pontos no mapa, traçando rotas entre os mesmos

através das ruas cadastradas no mapa, levando em consideração a existência, por exemplo, de ruas com mão e contra-mão (Geocodificação).

4.3 Serviços do Google Maps API

A ferramenta Google Maps API fornece uma biblioteca de serviços acessados pela linguagem de script JavaScript, sendo a mesma interpretada por qualquer navegador de Internet moderno. As principais chamadas ao serviço de geoprocessamento que serão utilizadas neste trabalho são:

- **Gmap2.setCenter(latlng:GLatLng)**: cria uma interface que exibe o mapa representando a Cartografia Digital da região associado ao centro fornecido por parâmetro.
- **GClientGeocoder.getLocations(query:String)**: este método envia uma solicitação ao serviço de geocodicação do Google, a geocodicação se refere a conversão de endereços legíveis para valores de latitude/longitude. O serviço de geocodicação do Google também suporta a geocodicação reversa, em que um ponto geográfico fornecido é convertido em um endereço.
- **GDirections.loadFromWaypoints(waypoints:Array)**:gera uma nova consulta de rotas de tráfego, podendo exibi-los em um mapa e/ou recuperar a distância destas rotas. Os pontos de localização que se deseja traçar a rota são fornecidos como entradas.

4.4 Sistema Otimizador

Analisamos na seção 2.3 deste trabalho diversos problemas de localização. Neste capítulo propõe-se uma arquitetura que permite uma flexibilidade ao usuário para definir a classe de problema que o mesmo quer tratar e qual algoritmo o Sistema Otimizador deverá utilizar para resolvê-lo. A principal entrada para qualquer problema de localização de facilidade é a matriz de distâncias entre clientes e facilitadores, para isso, integraremos nossa arquitetura com um WebGIS, sendo o mesmo o responsável pela geração da matriz

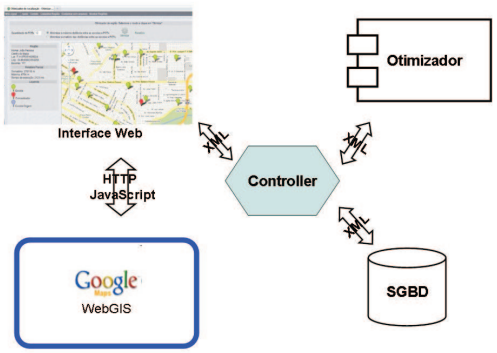


Figura 4.1: Arquitetura do Sistema

de distância real, que leva em consideração, por exemplo, a existência de mão e contra-mão do trânsito. O acesso ao sistema se dará através de um navegador web e conexão com a internet.

4.4.1 Arquitetura do sistema

A arquitetura do Sistema Otimizador (Figura 4.1) será dividida em cinco elementos: Interface de Usuário; WebGIS, responsável pelas informações geográficas e consulta das informações espaciais; Controller, controlador de configuração do Sistema; SGBD, responsável pela persistência dos dados do Sistema e o Componente Otimizador, responsável pelos algoritmos de resolução dos problemas de localização.

Nesta arquitetura os dados serão armazenados e transferidos no formato XML, por ser uma meta linguagem que pode ser usada para definir modos de descrever e capturar conteúdos de todos os tipos. Além disso, o XML é totalmente integrado com os serviços da Interface web (JavaScript) e a camada de persistência representada pelo Sistema de Gerenciamento de Banco de Dados (SGBD).

Na Interface de Usuário será utilizada a tecnologia web (Boente, 2004), especificamente as tecnologias HTML e JavaScript servem para a exibição dos componentes visuais do sistema, além da integração com o serviço Google Maps (WebGIS) para exibição de mapas interativos, que permitirão ao usuário ampliar, minimizar e mover para direita ou esquerda, para cima ou para baixo, com o intuito de mostrar áreas que podem não estar exibidas na tela.

4.4.2 Componente Otimizador

Brown e Wallnau (2006) descrevem um componente como "uma não-trivial, quase independente, e substituível parte de um sistema que cumpre uma função clara no contexto de uma arquitetura bem definida". Para Krutchen (2008), o componente é um elemento independente, que pode ser substituído, contudo, ele é significativo, pois tem uma função clara no contexto em que foi definido.

A chave para a flexibilidade da arquitetura que será proposta estará no Componente Otimizador, para isso será implementado uma coleção de componentes que variam pela classe de problema de localização abordado na seção 2.3.

Mesmo resolvendo a mesma classe de problemas, podem existir vários componentes distintos que se diferenciam pela classe de algoritmos utilizados para sua resolução, podendo variar na classe de Algoritmos exatos, mataheurística e métodos híbridos, que serão expostos no capítulo 5.

Desta forma o usuário além de escolher o tipo de problema que deseja resolver, terá a opção de escolher o método que melhor se adapte as dimensões da instância do problema e a necessidade da precisão da solução.

4.4.3 Controller e o Modelo de Componentes

O Controller será o componente central da arquitetura proposta, nele ocorrerão às configurações necessárias para o sistema se adaptar para a classe de pro-blema escolhida pelo usuário, será também responsável por carregar o otimizador específico com a classe de algoritmo definido para resolver o problema.

Para que haja esta flexibilidade do Sistema na escolha da classe de problema e da classe de algoritmo de solução, o Controller implementará o modelo de componente CCM (Corba Component Model), permitindo assim que o Componente Otimizador possa ser trocado em tempo de execução, e o Sistema seja reconfigurado pela escolha do usuário.

O modelo CCM, descrito em (Wang, 2000), define uma arquitetura para implantação de componentes constituída por um conjunto de interfaces, que são utilizadas por uma ferramenta de implantação utilizada pelo configurador do sistema (Controller) para implantar componentes. O processo de implantação de componentes proposto consiste dos seguintes passos:

1. Através de interações com um usuário, identificar a classe de problemas de localização e classe de algoritmo que será abordado, escolhendo assim o componente que será carregado;

2. Instalar a implementação dos componentes correspondentes, de acordo com as definições do passo anterior;

3. Por fim, as conexões entre as interfaces devem ser feitas e o componente deve entrar em execução.

Será de responsabilidade do Controller adaptar a Interface de Usuário para as especificidades de cada classe de problema abordado, passando a exigir do usuário os dados de entrada necessários para a resolução do problema. Este elemento também é responsável por adaptar o arquivo XML de intercomunicação entres os elementos da arquitetura para adaptarem-se as estruturas de dados específicas de cada classe de problema.

Capítulo 5

Aplicação no Sistema Eleitoral Brasileiro e Resultados Computacionais

Nesta capítulo descreveremos a implementação da metaheuríistica GRASP proposta. Detalharemos a representação das solução, sua fase de construção e busca local. Também será mostrado os resultado computacionais obtidos quando aplicamos o Sistema Otimizador ao problema de apuração dos votos enfrentado pelo Tribunal Regional Eleitoral da Paraíba como também, os resultado obtidos quando testamos a metaheurística nas instâncias da biblioteca OR-Library.

5.1 Metaheurística GRASP

Antes de descrevermos os detalhes de cada uma das fases da metaheurística GRASP implementada para o problema das P-medianas, introduziremos uma representação do problema e a forma como avaliaremos a escolha dos facilitadores que servirão cada cliente.

5.1.1 Representação da Solução

Como já mencionado, as dimensões básicas do problema são $n = |U|$, $m = |F|$, e p, o número de facilitadores a serem escolhidos. Por definição $1 \leq p \leq m$, ignora-se os

casos triviais e assume-se que $1 < p < m$ e que $p < n$. Não há relação entre n e m.

Neste trabalho, u representa um cliente genérico, e f um facilitador genérico. O custo de f servir a u é $d(u, f)$, que representa a distância entre eles. Uma solução S é qualquer subconjunto de F com p elementos. Cada usuário u deve ser atribuído ao facilitador mais próximo $f \in S$, minimizando $d(u, f)$. Este facilitador será chamado de $\phi_1(u)$; analogamente, o segundo facilitador mais próximo de u em S será chamado de $\phi_2(u)$. Para simplificar a notação, será adotado as seguintes abreviaturas $d(u, \phi_1(u))$ como $d_1(u)$, e $d(u, \phi_2(u))$ como $d_2(u)$.

Sendo assim, a função objetivo f, de cada solução S, para o problema da P-mediana é definida como sendo igual a:

$$f(S) = \sum_{u \in U} \phi_1(u) \tag{5.1}$$

Para o caso do problema P-mediana^{max} a função de avaliação será dada por

$$f(S) = \sum_{u \in U} \phi_1(u) \cdot \lambda + \Theta(1 - \lambda) \tag{5.2}$$

onde Θ é o valor máximo de $d_1(u)$, $\forall u \in U$ e $\lambda \in [0, 1]$ é a combinação convexa das duas funções. O valor de λ varia de acordo com a seguinte função:

$$\lambda = \lambda - |\frac{f_i(S) - f_f(S)}{f_f(S)}| \tag{5.3}$$

tal que, $f_i(S)$ é o valor de f para primeira iteração e $f_f(S)$ é o valor de f para segunda iteração. Tais valores são atualizados a cada nova iteração do algoritmo.

Durante a apresentação dos movimentos de vizinhança serão frequentemente referenciados os facilitadores que serão inseridos ou removidos, sendo assim, os facilitadores candidatos à inserção serão referenciados como f_i (por definição $f_i \notin S$); já os facilitadores candidatos à remoção serão referenciados por f_r ($fr \in S$).

Por fim, neste trabalho, a distância entre os clientes e qualquer facilitador pode ser encontrado em apenas um passo ($O(1)$). Para isso, há a necessidade da existência de uma matriz de distâncias. Neste modelo, todos os valores de f_1 e f_2 para uma dada solução S pode ser computado em $p \cdot n$ passos ($O(pn)$).

5.1.2 Construção da Solução

Antes de entendermos como a metaherística GRASP foi adaptada ao problema das p medianas, devemos ressaltar que durante vários pontos da nossa proposta de solução, levamos em consideração um aspecto especial encontrado em nossas instâncias, cuja confecção será apresentada na sessão 4, e nas instâncias utilizadas na literatura, como por exemplo, (Resende, 2002; Resende, 2003; Tseng, 2009), onde cada localização pode ser um cliente ou um facilitador, ou seja, os conjuntos F e U são os mesmos. A seguir será apresentado o processo de *clusterização* da matriz de distâncias de uma instância e o processo de construção da solução.

5.1.3 *Clusterização* da Matriz de Distâncias

O processo de *clusterização* (agrupamento) da matriz de distâncias tem o objetivo de indicar pontos que estejam próximos entre si, atribuindo estes pontos ao mesmo *cluster*, construindo assim um conjunto C de *clusters*. Este processo de *clusterização* se dá a partir da definição de um valor chamado $filtro$, este valor foi definido experimentalmente (obteve os melhores resultados durante a fase de construção da solução) como:

$$filtro = 2 \cdot (dist_{max}/n) \tag{5.4}$$

onde $dist_{max}$ é o maior valor encontrado na matriz de distâncias e n o número de localizações (clientes ou facilitadores). De posse do valor do $filtro$ será criada uma nova matriz, chamada matriz de *clusterização*, de mesma dimensão da matriz de distâncias, porém seus valores são preenchidos segundo a seguinte regra: para cada valor da matriz de distâncias, o mesmo será comparado com o valor do $filtro$, se o valor for menor ou igual a filtro coloca-se o valor 1 na posição correspondente na matriz de *clusterização*, caso contrário coloca-se o valor 0. Este processo pode ser observado na ilustração da Figura 6.1.

O último passo no processo de clusterização é a geração dos *clusters*, para isso iremos utilizar o exemplo apresentado na Figura 6.1, onde podemos observar que a matriz de *clusterização* resultante é binária. Este último passo se dá pela análise de cada linha da matriz de *clusterização*, todos os índices das colunas que estiverem com o valor 1 formarão

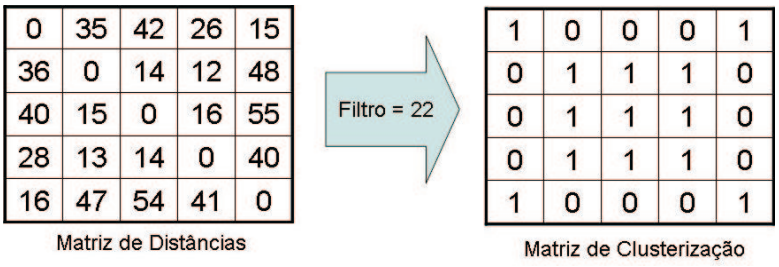


Figura 5.1: Exemplo do processo de clusterização da matriz de distâncias.

um *cluster*, indicando assim que estes pontos estão próximos em relação a suas distâncias. Sempre que for visitada uma nova linha deve-se verificar se este índice já faz parte do conjunto de C de *clusters*.

De acordo com o exemplo da figura 6.1 temos que a distância entre os pontos $\{0, 4\}$ é menor que o valor do $filtro$ (valor mostrado na primeira linha e quarta coluna da matriz) indicando assim que foi formado um *cluster*. Seguindo o mesmo raciocínio, foi criado outro *cluster* contendo os elementos $\{1, 2, 3\}$.

5.1.4 Fase de Construção GRASP

Na fase de construção, iniciamos com uma solução S vazia, ou seja, não temos a presença de facilitadores. A cada iteração é escolhido um *cluster* $c \in C$, e deste *cluster* será escolhido, de forma aleatória, um facilitador a partir da lista restrita de candidatos (LCR). A lista LCR corresponde a uma parte dos melhores facilitadores ainda não escolhidos dentro do *cluster*. Esta lista é obtida com o auxílio de uma função gulosa, denotada por $g(f_i)$, definida para cada facilitador f_i candidato. A função $g(f_i)$ corresponde à média aritmética das distâncias de todos os outros facilitadores do *cluster* ao facilitador f_i.

Os melhores candidatos formados pelos facilitadores candidatos serão aqueles que satisfizerem a condição:

$$g(f_i) \leq g_{min} + \alpha(g_{max} - g_{min}) \tag{5.5}$$

onde $g_{min} = min\{g(f_i)|(fi \in c)\ e\ (f_i \notin S)\}$, $g_{max} = max\{g(f_i)|(f_i \in c)\ e\ (f_i \notin S)\}$ e $\alpha \in (0, 1)$.

Por último, após a obtenção da LRC e a escolha aleatória de um facilitador $f_i \in LRC$, realizamos o processo de adaptação, ou seja, adicionamos o facilitador f_i à

solução S e passamos para outro *cluster* do conjunto C. Um pseudocódigo do algoritmo da fase de construção é apresentado.

Algoritmo Fase - Construção (n, m, C, $Semente$, α)

1. $S = \{\}$; \\ nenhum facilitador escolhido
2. Inicializar a função gulosa g;
3. $c = \{c|c \in C\}$; \\ primeiro cluster C
4. equanto ($|S| < p$) faça
5. {
6. $LC = \{f_i$ em ordem crescente pela função $g(f_i)|(f_i) \in C$ e $(f_i \notin S)\}$;
7. $g_{min} = min\{g(f_i)|f_i \in LC\}$;
8. $g_{max} = max\{g(f_i)|f_i \in LC\}$;
9. $LCR = \{f_i \in LCR|g(S) \leq g_{min} + \alpha(g_{max} - g_{min})\}$;
10. Selecionar de forma aleatória um elemento f de LCR;
11. $S = S \cup \{f\}$;
12. \\ pegar o próximo cluster $c \in C$;
13. $c = \{c|c \in C\}$;
14. }
15. retornar (S).

Pode-se observar pelo algoritmo da Figura 2 que a condição de parada da fase de construção é definida quando são atribuídos a solução S, p facilitadores que, por definição, é o número necessário para se obter uma solução viável do problema. Na linha 13 da Figura 2, podemos observar a atualização do cluster c, porém o número de elementos do conjunto C é frequentemente menor que p, logo em certo momento quando todos os elementos de C já foram visitados, o *cluster* c volta a receber o primeiro *cluster* de C.

5.1.5 Busca Local

Na fase de busca local, as soluções vizinhas à solução obtida na fase de construção serão geradas, utilizando-se vizinhanças definidas na literatura e variações da mesma proposta neste trabalho. Esta seção irá apresentar a vizinhança adotada na literatura e os algoritmos propostos neste trabalho.

Busca Local *Swap-based*

Introduzido por Teitz e Bart em (Teitz, 1968), o procedimento padrão de busca local para o problema das p medianas é baseado na troca de facilitadores (*swap-based*). Para cada facilitador $f_i \notin S$ o procedimento indica qual facilitador $f_r \in S$ irá melhorar a função caso f_i e f_r sejam trocados(f_i será inserido e f_r será removido da solução). Se existir um movimento de *swap* de melhora, o mesmo será realizado e o procedimento é repetido a partir de uma nova solução. Caso contrário o procedimento para, tendo assim encontrado o ótimo local.

Vizinhança Proposta

Este trabalho propõe a divisão do escopo de candidatos em três subconjuntos de movimento *swap*, estes subconjuntos são descritos a seguir, os mesmos associados a um procedimento de busca local variável, tem a função de acelerar a busca por melhores vizinhos, descendo rapidamente na vizinhança, pois inicia com um escopo menor de candidatos, e este escopo vai aumentando sempre que o atual não conseguir mais obter melhorias na solução.

Subconjuntos do *swap*

- *Swap-Assing*: Cada f_r será trocado apenas com os f_i , cujo $\phi_1(f_i) = f_r$, ou seja, serão testadas as trocas apenas entre f_r e as localizações cujo o facilitador mais próximo é o próprio f_r.

- *Swap-Cluster*: Cada f_r será trocado apenas com os f_i pertencentes ao mesmo *cluster* e cujo $\phi_1(f_i) \neq f_r$.

- *Swap-InterCluster*: Cada f_r será trocado apenas com os f_i dos outros *clusters*.

Método de Descida em Vizinhança Variável

Na fase de busca local do GRASP, foi adotado o Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND), proposto por Nenad Mladenovic e Pierre Hansen (Mladenovic, 1997). É um método de refinamento que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções que melhoram a solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada.

O pseudocódigo desse algoritmo, em que se considera o refinamento de uma solução S utilizando uma função de avaliação $f(S)$, a ser minimizada, e um conjunto N de r diferentes vizinhanças, que para este trabalho pode ser representado por $N = \{Swap\text{-}Assig, Swap\text{-}Cluster, Swap\text{-}InterCluster\}$, sendo ilustrado abaixo.

Procedimento VND($f(.)$, $N(.)$, r, s)

1. Seja r o número de estruturas diferentes de vizinhança.
2. $k \leftarrow 1$; {Tipo de estrutura de vizinhança corrente}
3. enquanto $(k \leq r)$ faça
4. Encontre o melhor vizinho $s' \in N^{(k)}(s)$;
5. se $(f(s') < f(s))$;
6. então
7. $s \leftarrow s'$;
8. $k \leftarrow 1$;
9. senão
10. $k \leftarrow k + 1$;
11. fim-se;
12. fim-enquanto;
13. Retorne s

fim VND;

O hibridismo da metaheurística proposta se dá quando é utilizado o método VND na fase de construção do GRASP.

5.2 Resultados Computacionais

Nesta seção, apresentamos os resultados computacionais obtidos com a metaheurística GRASP proposta para a resolução do problema das p medianas. Esta metaheurística foi implementada utilizando a linguagem C++. Todos os experimentos computacionais foram realizados em um computador com sistema operacional Linux Ubuntu 9.04 e com processador Intel Core 2 Quad 2.33 Ghz com 2 GB de memória RAM.

Antes de apresentarmos os resultados obtidos mencionaremos os valores adotados para os parâmetros da metaheurística GRASP. Estabeleceu-se que o número

máximo de iterações, denotado por NUM_MAX_ITER, fosse igual a 50. O valor escolhido para α é 0.9, que demonstrou melhores resultados durante os experimentos.

A instância utilizada para a obtenção dos resultados apresentados nesta seção foi gerada a partir das informações fornecidas pelo Tribunal Regional Eleitoral da Paraíba, dados estes relacionados com o pleito de 2008. Foram fornecidas as localizações de 177 locais de votação utilizados na cidade de João Pessoa, a matriz de distâncias foi calculada com o auxílio do Sistema abordado no capítulo 4, que integrado ao Google Maps, consegue obter resultados reais das distâncias, considerando a existência de mão e contra-mão das ruas, gerando assim uma instância intitulada JP_177. Essa mesma instância foi dividida por zonas para gerarmos outras instâncias, onde o número de elementos e de concentradores varia (Z01, Z64, Z70, Z76, Z77).

Nesta aplicação, a interface com o usuário permite que o mesmo defina quais os pontos da cidade que serão candidatos a pontos de coleta e transmissão. Também permite, ao registrar um ponto qualquer da cidade, optar por ele ser obrigatoriamente um PCT ou de ser apenas um receptor e, gerar de relatórios que trazem como principal informação os PCT, destacados no mapa pela cor vermelha, e os pontos que estão ligados a ele, bem como a distância entre eles.

Para efeito de comparação entre os resultados obtidos para as duas classes de problema, o P-Mediana original e sua variação P-Mediana^{max}, calculamos os valores da solução real gerada pela equipe de logística do TRE-PB no ano de 2008.

Instância	TRE-PB		P-mediana^{max}			
	Soma	Max_Dist	Soma	gap(%)	Max_Dist	gap(%)
JP_177 / p=26	277486	8821	140172	-49,485	4107	-53,441
Z01 / p=3	59252	3384	45576	-23,08	3123	-7,71
Z64 / p=10	31176	4283	9533	-69,42	753	-82,42
Z70 / p=4	53743	8394	34254	-36,26	2594	-69,10
Z76 / p=5	75990	8821	44498	-41,44	6525	-26,03
Z77 / p=4	57325	4153	42037	-26,67	3078	-25,88

Tabela 1: Comparação entre o resultado do TRE-PB e resultado do modelo da P-mediana

Instancia	TRE-PB		P-mediana^{max}			
	Soma	Max_Dist	Soma	gap(%)	Max_Dist	gap(%)
JP_177 / p=26	277486	8821	142410	-48,678	2524	-71,386
Z01 / p=3	59252	3384	51598	-12,92	2384	-29,55
Z64 / p=10	31176	4283	9591	-69,24	753	-82,42
Z70 / p=4	53743	8394	34577	-35,66	2594	-69,10
Z76 / p=5	75990	8821	48332	-36,40	3316	-62,41
Z77 / p=4	57325	4153	44165	-22,96	2840	-31,62

Tabela 2: Comparação entre o resultado do TRE-PB e o resultado do modelo P-mediana^{max}

onde, **Soma** representa o somatório da distância total percorrida dos locais de votação ao seus respectivos PCTs e **Max_Dist** representa a distância máxima percorrida entre o local de votação e seu respectivo PCT.

O modelo P-mediana (tabela 1) obteve melhorias, com relação ao somatório das distâncias totais percorridas de 49,48%. Ao fazer uma otimização por zona eleitoral as melhoras foram superiores a 23% em todas as intâncias quando comparadas ao resultado do TRE-PB. Em particular, houve um ganho de 69,42% na instância Z64.

Observa-se que os resultados obtidos pelo P-mediana^{max} (tabela 2), quando comparados a máxima distância da instância JP_177 / p=26, tiveram uma melhoria de 71,3% com relação ao valor da solução usada pelo TRE-PB. Já o modelo P-Mediana original melhorou 53,4%, que é um bom resultado, porém bem abaixo do resultado do P-Mediana^{max} proposto neste trabalho.

O modelo P-Mediana^{max} obteve esta melhora sem aumentar muito o somatório das distâncias, obtendo uma melhoria de 48,6% do resultado do TRE-PB contra 49,4% do P-mediana original. Esse fato ocorreu pois a função de avaliação do modelo P-mediana^{max} é uma combinação convexa dos dois modelos.

Para validar a implementação da nossa metaheurística GRASP com o VND fizemos testes com instâncias da biblioteca ORLibrary. Veja resultado na tabela 3.

Instancia			Ótimo	Metaheurística				GAP_{min}
nome	n	p	F	F_{min}	F_{med}	F_{max}	tempo	(%)
pmed1.txt	100	5	5819	5819	5819	5819	77	0,00
pmed2.txt	100	10	4093	4093	4098,399902	4102	109	0,00
pmed3.txt	100	10	4250	4250	4250	4250	137	0,00
pmed4.txt	100	20	3034	3034	3034	3034	162	0,00
pmed5.txt	100	33	1355	1358	1358	1358	265	0,221
pmed6.txt	200	5	7824	7824	7824	7824	636	0,00
pmed7.txt	200	10	5631	5631	5631	5631	662	0,00
pmed8.txt	200	20	4445	4445	4445	4445	682	0,00
pmed9.txt	200	40	2734	2737	2737,199951	2738	1292	0,110
pmed10.txt	200	67	1255	1256	1256	1256	2998	0,080
pmed11.txt	300	5	7696	7696	7696	7696	3375	0,00
pmed12.txt	300	10	6634	6634	6634	6634	2816	0,00
pmed13.txt	300	30	4374	4374	4374	4374	3236	0,00
pmed14.txt	300	60	2968	2968	2979,199951	2973	4815	0,00
pmed15.txt	300	100	1729	1733	1735	1737	10362	0,231
pmed16.txt	400	5	8162	8162	8162	8162	16214	0,00
pmed17.txt	400	10	6999	6999	6999	6999	8215	0,00
pmed18.txt	400	40	4809	4809	4811,399902	4815	8250	0,00
pmed19.txt	400	80	2845	2847	2847,800049	2849	15213	0,070
pmed20.txt	400	133	1789	1790	1792,800049	1795	33767	0,056
pmed21.txt	500	5	9138	9138	9138	9138	31278	0,00
pmed22.txt	500	10	8579	8579	8579	8579	20497	0,00
pmed23.txt	500	50	4619	4619	4621,600098	4628	17171	0,00
pmed24.txt	500	100	2961	2964	2967	2968	31691	0,101
pmed25.txt	500	167	1828	1830	1833,800049	1837	70070	0,109
pmed26.txt	600	5	9917	9917	9917	9917	57876	0,00
pmed27.txt	600	10	8307	8307	8307	8307	37016	0,00
pmed28.txt	600	60	4498	4503	4505,399902	4508	34265	0,111
pmed29.txt	600	120	3033	3045	3047	3049	63261	0,396
pmed30.txt	600	200	1989	2018	2018	2018	106592	1,458
pmed31.txt	700	5	10086	10086	10086	10086	71900	0,00
pmed32.txt	700	10	9297	9297	9297	9297	110461	0,00
pmed33.txt	700	70	4700	4702	4707	4712	52379	0,043
pmed34.txt	700	140	3013	3019	3021	3024	114019	0,199
pmed35.txt	800	5	10400	10400	10400	10400	95450	0,00
pmed36.txt	800	10	9934	9934	9934	9934	88697	0,00
pmed37.txt	800	80	5057	5060	5064	5068	59792	0,059
pmed38.txt	900	5	11060	11060	11060	11060	33725	0,00
pmed39.txt	900	10	9423	9423	9423	9423	319090	0,00
pmed40.txt	900	90	5128	5133	5135,399902	5139	167431	0,098

Tabela 3: Comparação entre da biblioteca OR-Library e a implementação da mataheurística GRASP proposta.

De acordo com a tabela 3, temos que das 40 intâncias testadas, a metaheurística proposta neste trabalho encontrou o ponto ótimo em 62,5% dos casos. Nas demais intâncias o resultado da metaheurística se aproximou bastante do valor ótimo, tendo um *gap* máximo de 1,458%.

Capítulo 6

Manual de utilização do Sistema Otimizador

Neste capítulo iremos descrever detalhadamente as funções existentes no Sistema Otimizador, proposto no capítulo 4, sempre tomando como exemplo a aplicação feita no Tribunal Regional Eleitoral da Paraíba. Também será feito um breve histórico sobre o sistema eleitoral brasileiro.

6.1 Histórico

A partir da independência do Brasil, o país obrigou-se a aperfeiçoar sua legislação eleitoral que durante todo império teve suas normas copiadas do modelo Francês. Surgiu então a necessidade de criar a Justiça Eleitoral, que foi instituída em 1930 com o objetivo de moralizar as eleições, e é composta pelo Tribunal Superior Eleitoral; por um Tribunal Regional em cada estado, no Distrito Federal e nos Territórios; pelos juízes e pelas juntas eleitorais.

Também em 1930 foi criado o primeiro Código Eleitoral brasileiro que estabeleceu uma série de medidas para sanar os "vícios eleitorais". E já previa o uso da máquina de votar. A Justiça Eleitoral, agora responsável por todos os trabalhos eleitorais (alistamento, organização das mesas de votação, apuração dos votos, proclamação e diplomação dos eleitos), buscava mecanismos para garantir a lisura dos pleitos.

Somente nas eleições municipais de 1996, no entanto, é que a Justiça Eleitoral deu início ao processo de informatização do voto. Usaram a "máquina de votar", nesse

ano, cerca de 33 milhões de eleitores. E a partir do ano 2000, todos os eleitores puderam utilizar as urnas eletrônicas para eleger seus candidatos (TSE, 2010).

Atualmente no Brasil, em particular, na Paraíba, seu respectivo Tribunal Regional Eleitoral - TRE - utiliza a seguinte logística na realização das eleições: inicialmente as urnas são distribuídas para seus respectivos locais de votação; depois de encerradas as eleições as mídias são levadas a pontos de coletas e transmissão, que são escolhidos empiricamente, e depois de apurados os votos o resultado é divulgado.

Para que o resultado da eleição seja divulgado mais rápido, o TRE-PB precisa minimizar o tempo de deslocamento das mídias aos pontos de coleta e transmissão. Umas das possibilidades é ter cada local de votação como um ponto de coleta e transmissão das mídias. Mas, devido à inviabilidade dessa opção (elevados custos e dificuldades no gerenciamento), escolhem-se alguns pontos de tal maneira a atingir o objetivo de minimizar este tempo de deslocamento. Na literatura, a escolha destes pontos de coleta e transmissão, é conhecido como um problema de localização de facilidades denominado problema das P-medianas. Porém o P-mediana trata a minimização do somatório das distâncias e não a minimização da distância máxima, não garantindo assim uma otimização individual da entrega da mídia de cada local de votação a seu respectivo Ponto de Coleta e Transmissão - PCT. Por este motivo, foram realizados testes utilizando o modelo clássico da P-mediana e o modelo P-mediana^{max}.

6.2 Sistema Otimizador

O acesso ao sistema se dá através do seguinte endereço eletrônico: www.minmaxsistemas.com.br/tre. A página inicial do sistema está ilustrada na figura 6.1.

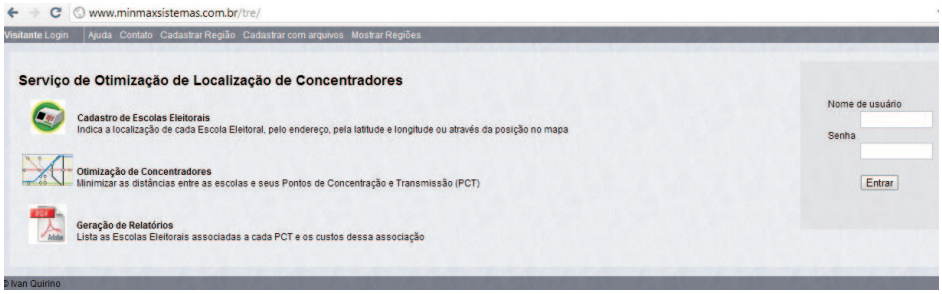


Figura 6.1: Tela de acesso.

Para utilizar o sistema o usuário deverá ser previamente cadastrado pela equipe de suporte que o forncerá um login e uma senha. Depois de digitar seu login e sua respectiva senha, o usuário irá se deparar com uma tela que já contém algumas regiões cadastradas (Figura 6.2), que para o problema P-mediana ou P-mediana^{max} será a instância, ou seja, o número de pontos que o problema irá trabalhar.

Figura 6.2: Instâncias.

Caso seja de interesse do usuário, o mesmo poderá cadastrar novas regiões através do mapa ou por arquivos, através dos ícones **cadastrar região** ou **cadastrar arquivos** que estão em vermelho na figura A.2. Ao clicar em cadastrar região o sistema o abrirá uma tela de cadastro, ilustrada pela figura 6.3.

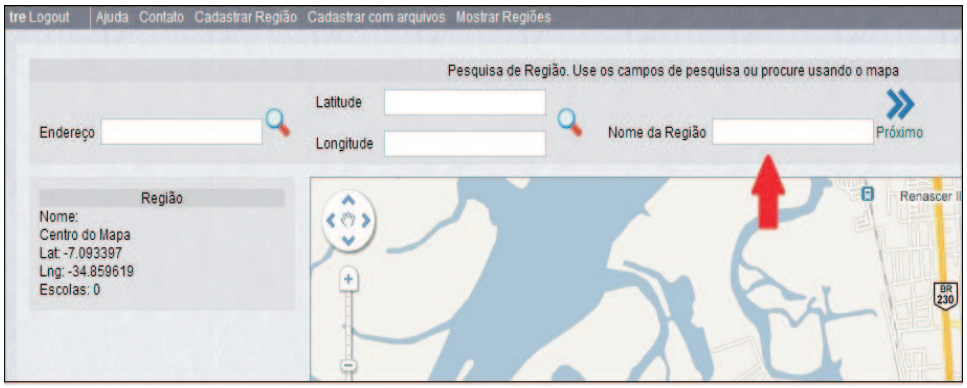


Figura 6.3: Tela de cadastro de região.

Nesta tela, localiza-se um endereço qualquer digitanto no espaço em branco ao lado de **Endereço** em seguida, cria-se um nome para região, digitando no espaço em branco indicado na figura A.3 pela seta vermelha. Para finalizar, aperta-se no botão **Próximo**. Nesse momento, o sistema irá direcionar o usuário a uma tela de cadastro dos locais de votação (Figura 6.4).

Nesta página, o usuário terá três opções para cadastrar um local de votação. Através de um endereço no formato (Rua, número, cidade), clicando em cima do mapa,

ou mesmo, se for conveniente, através da latitude e longitude de cada ponto. Ao realizar uma das três opções de cadastro, o sistema mostrará o balão da figura 6.4.

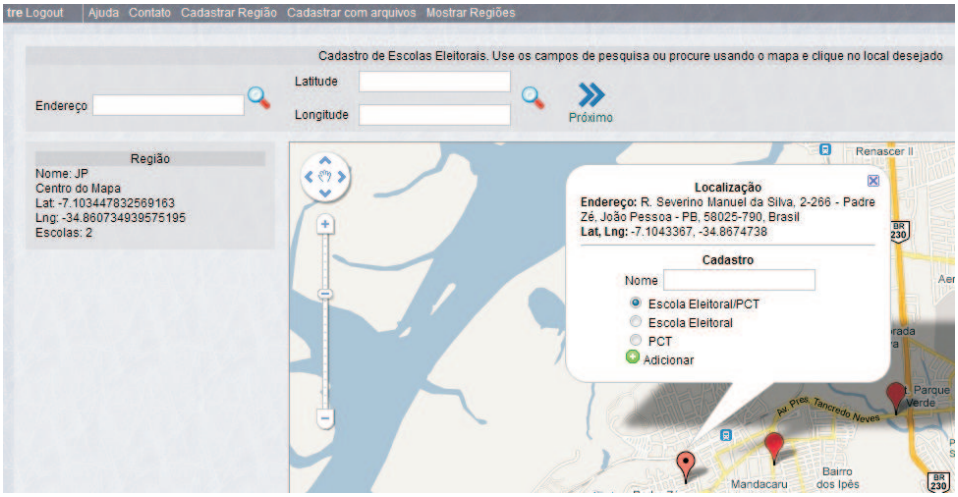


Figura 6.4: Tela de cadastro dos locais de votação.

Através do balão, o sistema oferece três opções de cadastro: **Escola Eleitoral/PCT**, clicando neta opção o usuário permitirá que o sistema defina se o ponto será uma Escola Eleitoral (local de votação), apenas um PCT, ou ambos; **Escola Eleitoral**, o local será obrigatoriamente apenas um local de votação; **PCT**, o local será um ponto de coleta e transmissão. Depois de fazer a opção aperta-se em adicionar. Ao terminar de cadastrar todos os locais de votação, clica-se no botão **Próximo**. O sistema o encaminhará para tela ilustrada pela figura 6.5

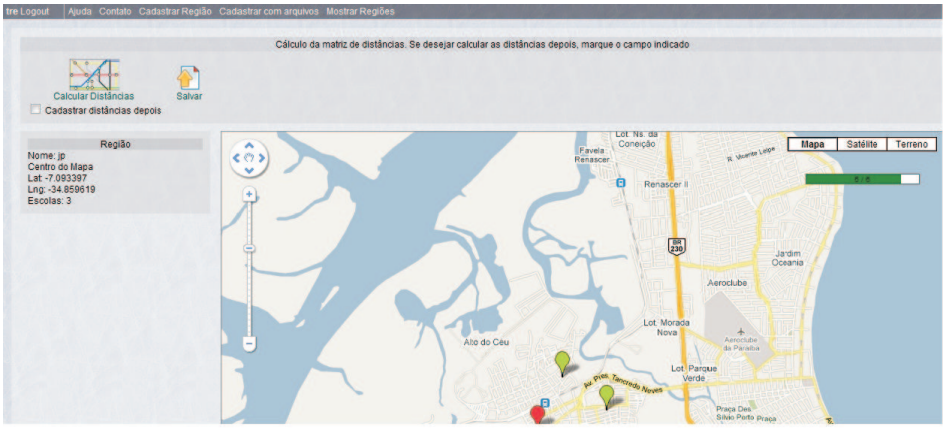


Figura 6.5: Tela de cadastro da matriz de distâncias.

Nesta tela, o usuário irá apertar em **Calcular distâncias** para que o sistema gere a matriz de distâncias. Em seguida aperta em **Salvar**. Depois de salvar a região o

usuário irá fazer a opção de otimizar a região. Para isso o sistema abrirá a tela ilustrada pela figura A.6

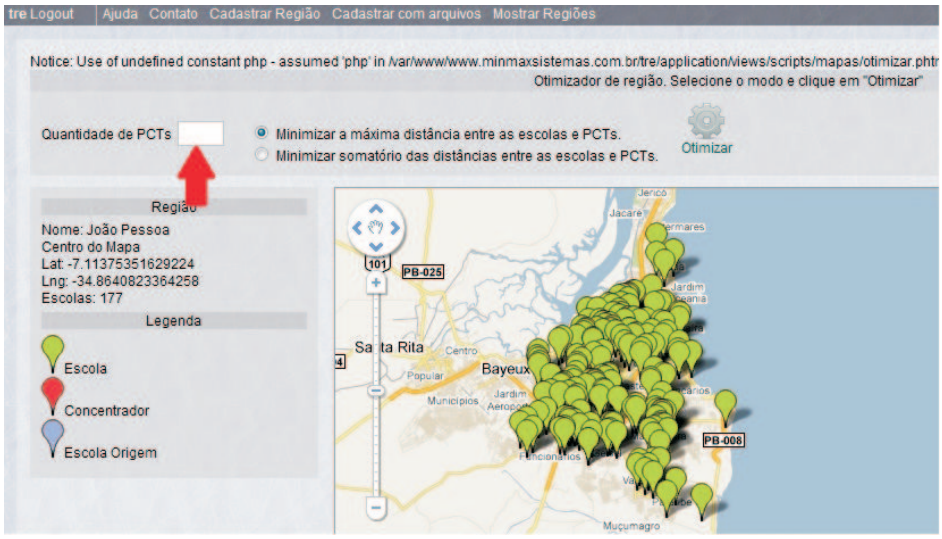


Figura 6.6: Tela para especificar o número de pontos de coleta e transmissão e o problema a ser resolvido.

Agora o usuário irá definir a quantidade que ele deseja de pontos de coleta e transmissão digitando o número no local indicado pela seta vermelha na figura 6.6. Em seguida fará a opção de minimizar a máxima distância entre as escolas e PCTs, o que se refere ao modelo do P-mediana^{max}, ou minimizar o somatório das distâncias entre as escolas e os PCTs, o que se refere ao modelo das p-medianas. Para finalizar, clica-se em **otimizar**.

Após o sistema ativar seu Componente Otimizador e o mesmo otimizar a instância específica, como resultado, teremos a indicação de quais são os locais de votação que foram definidos como PCT e quais locais de votação estão ligados a ele. Veja a figura 6.7.

Os pontos verdes, de acordo com a legenda se referem aos locais de votação; os pontos vermelhos são os PCT (concentradores). Para que o sistema exiba quais locais de votação estão diretamente ligados a cada PCT basta clicar em cima do ponto vermelho e a tela, ilustrada pela figura 6.8 aparecerá

Caso seja de interesse do usuário, o sistema gera um relatório, no formato pdf, para isto basta clicar em **Relatório** (Figura 6.9).

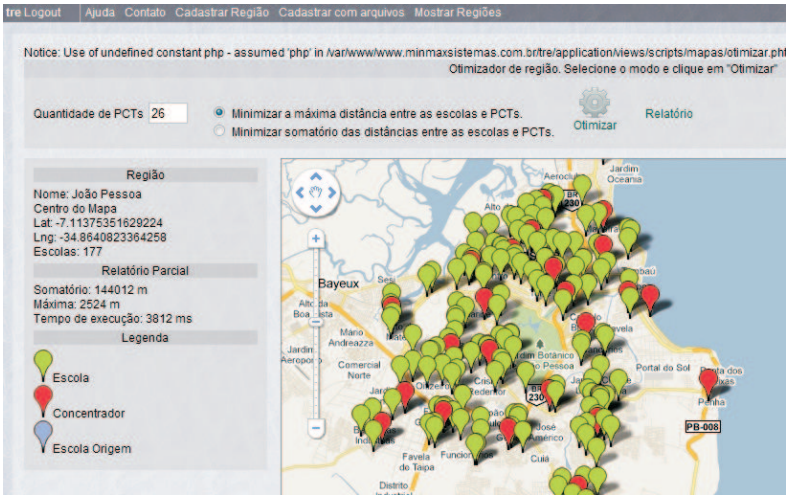


Figura 6.7: Tela de resultado.

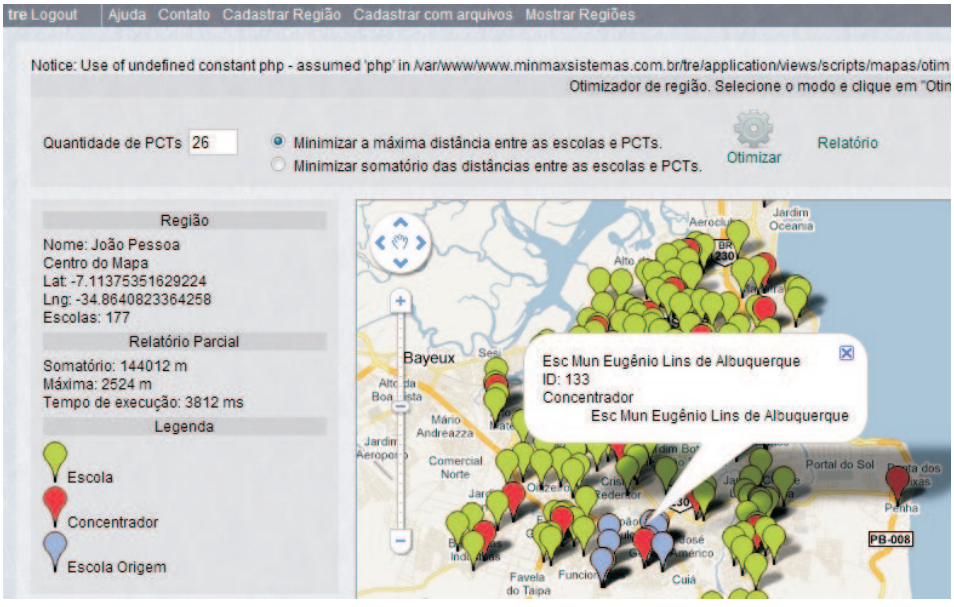
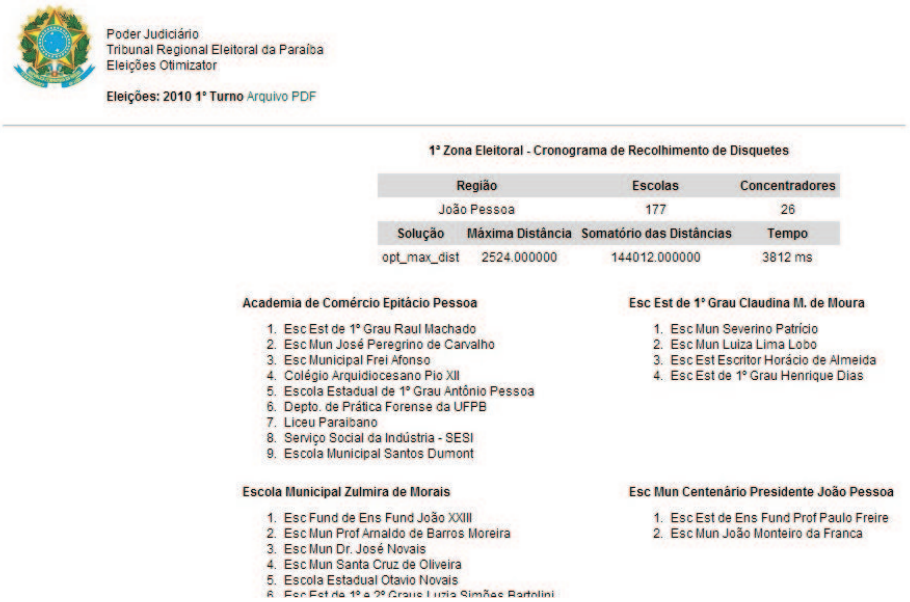


Figura 6.8: Tela de atribuição.



Poder Judiciário
Tribunal Regional Eleitoral da Paraíba
Eleições Otimizator

Eleições: 2010 1º Turno Arquivo PDF

1ª Zona Eleitoral - Cronograma de Recolhimento de Disquetes

Região	Escolas	Concentradores
João Pessoa	177	26

Solução	Máxima Distância	Somatório das Distâncias	Tempo
opt_max_dist	2524.000000	144012.000000	3812 ms

Academia de Comércio Epitácio Pessoa

1. Esc Est de 1º Grau Raul Machado
2. Esc Mun José Peregrino de Carvalho
3. Esc Municipal Frei Afonso
4. Colégio Arquidiocesano Pio XII
5. Escola Estadual de 1º Grau Antônio Pessoa
6. Depto. de Prática Forense da UFPB
7. Liceu Paraibano
8. Serviço Social da Indústria - SESI
9. Escola Municipal Santos Dumont

Esc Est de 1º Grau Claudina M. de Moura

1. Esc Mun Severino Patrício
2. Esc Mun Luiza Lima Lobo
3. Esc Est Escritor Horácio de Almeida
4. Esc Est de 1º Grau Henrique Dias

Escola Municipal Zulmira de Morais

1. Esc Fund de Ens Fund João XXIII
2. Esc Mun Prof Arnaldo de Barros Moreira
3. Esc Mun Dr. José Novais
4. Esc Mun Santa Cruz de Oliveira
5. Escola Estadual Otavio Novais
6. Esc Est de 1º e 2º Graus Luzia Simões Bartolini

Esc Mun Centenário Presidente João Pessoa

1. Esc Est de Ens Fund Prof Paulo Freire
2. Esc Mun João Monteiro da Franca

Figura 6.9: Relatório.

Capítulo 7

Conclusões

Problemas de localização de facilidade tem sido bastante valorizados devido sua grande aplicabilidade em diversas áreas do conhecimento e, inclusive, em aplicações que ajudam a melhorar problemas reais. Para que esses problemas sejam resolvidos mais rápido e facilmente foi proposto, neste trabalho, um Sistema Otimizador que integrado ao um sistema WebGIS pretende resolver modelos genéricos de problemas de localização com uma pequena interferência do usuário. Tal sistema é flexível pois além de resolver um modelo genérico de problema de localização também permite ao usuário definir o método de resolução a ser utilizado, de acordo com a classe de cada problema.

Para validar a arquitetura proposta para o sistema, foi feita uma aplicação do mesmo ao problema enfrentado pelo TRE-PB. Para minimizar o tempo gasto na apuração dos votos, depois do pleito ter sido encerrado, o TRE-PB pretende minimizar a distância total percorrida para levar as mídias dos locais de votação até os pontos de coleta e transmissão. Pois quando menor for a distância percorrida, supõe-se que menor será o tempo gasto. Então o intuito é descobrir qual o melhor local para instalar os pontos de coleta e transmissão. Na literatura essa situação foi mapeada como o problema das P-medianas.

O método de resolução utilizado foi a metahurística GRASP com uma aplicação do VND na fase de busca local. Os resultados obtidos melhoraram em torno de 48% o resultado do TRE-PB. Mas, para comprovar a eficiência da metaheurística, também foram realizados testes com as instâncias da biblioteca OR-Library e foi constatado que a metaheurística atingiu o valor ótimo em 62,5% dos casos testados.

Como proposta de trabalhos futuros é sugerido implementar outro modelo do

problema de localização de cobertura e também fazer um hibridismo da metaheurística GRASP com métodos exatos.

Apêndice A

Valores dos testes realizados com a metaheurística

Neste anexe será apresentada a justificativa da escolha do valor de $\alpha = 0,9$ para a metaheurística GRASP.

A.1 Testes

Os testes com a metaheurística foram realizados com vários valores α. A escolha da utilização do α 0,9 na metaheurística justifica-se por que é o que apresenta a menor média para o *gap.*

De acordo com a tabela 4, temos que a média dos valores do *gap* para $\alpha = 0,3$ é $med = 0,0012$

Instancia			Ótimo	Metaheurística $\alpha = 0,3$				
nome	n	p	F	F_{min}	F_{med}	F_{max}	tempo	gap (%)
pmed1.txt	100	5	5819	5819	5819	5819	77	0
pmed2.txt	100	10	4093	4093	4093	4093	90	0
pmed3.txt	100	10	4250	4250	4250	4250	120	0
pmed4.txt	100	20	3034	3038	3038	3038	163	0,0013
pmed5.txt	100	33	1355	1358	1358	1358	256	0,0022
pmed6.txt	200	5	7824	7824	7824	7824	633	0
pmed7.txt	200	10	5631	5631	5631	5631	660	0
pmed8.txt	200	20	4445	4445	4445	4445	632	0
pmed9.txt	200	40	2734	2737	2737,199951	2738	1180	0,0011
pmed10.txt	200	67	1255	1255	1255.800049	1256	2872	0,0006
pmed11.txt	300	5	7696	7696	7696	7696	3399	0
pmed12.txt	300	10	6634	6634	6634	6634	2800	0
pmed13.txt	300	30	4374	4374	4374	4374	3178	0
pmed14.txt	300	60	2968	2968	2969.600098	2972	4860	0,0005
pmed15.txt	300	100	1729	1732	1732.800049	1733	9731	0,0022
pmed16.txt	400	5	8162	8162	8162	8162	16243	0
pmed17.txt	400	10	6999	6999	6999.799805	7003	8399	0,0001
pmed18.txt	400	40	4809	4811	4811	4811	8201	0,0004
pmed19.txt	400	80	2845	2845	2847.800049	2851	14947	0,0010
pmed20.txt	400	133	1789	1789	1790.800049	1792	33721	0,0010
pmed21.txt	500	5	9138	9138	9138	9138	31195	0
pmed22.txt	500	10	8579	8579	8579	8579	21701	0
pmed23.txt	500	50	4619	4619	4623	4629	15376	0,0009
pmed24.txt	500	100	2961	2964	2967	2969	32160	0,0020
pmed25.txt	500	167	1828	1834	1837.400024	1840	73889	0,0051
pmed26.txt	600	5	9917	9917	9917	9917	58301	0
pmed27.txt	600	10	8307	8307	8307	8307	35957	0
pmed28.txt	600	60	4498	4507	4508.399902	4510	34460	0,0023
pmed29.txt	600	120	3033	3045	3046.600098	3048	60537	0,0045
pmed30.txt	600	200	1989	2018	2018	2018	106087	0,0146
pmed31.txt	700	5	10086	10086	10086	10086	71465	0
pmed32.txt	700	10	9297	9297	9297.799805	9301	109355	0,0001
pmed33.txt	700	70	4700	4704	4705.399902	4706	55790	0,0011
pmed34.txt	700	140	3013	3014	3018.399902	3021	118856	0,0017
pmed35.txt	800	5	10400	10400	10400	10400	96295	0
pmed36.txt	800	10	9934	9934	9934	9934	90142	0
pmed37.txt	800	80	5057	5058	5063.600098	5069	62862	0,0013
pmed38.txt	900	5	11060	11060	11060	11060	34132	0
pmed39.txt	900	10	9423	9423	9423	9423	317024	0,00
pmed40.txt	900	90	5128	5134	5137.799805	5142	165389	0,0019

Tabela 4: Valores da metaheurística para o parâmetro $\alpha = 0,3$.

De acordo com a tabela 5, para o valor de $\alpha = 0,5$ temos que a média do *gap* é dada por $med = 0,0012$

Instancia			Ótimo	Metaheurística $\alpha = 0,5$				
nome	n	p	F	F_{min}	F_{med}	F_{max}	tempo	gap (%)
pmed1.txt	100	5	5819	5819	5819	5819	77	0
pmed2.txt	100	10	4093	4093	4098.399902	4102	109	0,0013
pmed3.txt	100	10	4250	4250	4250	4250	137	0
pmed4.txt	100	20	3034	3034	3034	3034	162	0
pmed5.txt	100	33	1355	1358	1358	1358	265	0,0022
pmed6.txt	200	5	7824	7824	7824	7824	636	0
pmed7.txt	200	10	5631	5631	5631	5631	662	0
pmed8.txt	200	20	4445	4445	4445	4445	682	0
pmed9.txt	200	40	2734	2737	2737.199951	2738	1292	0,0012
pmed10.txt	200	67	1255	1256	1256	1256	2998	0,0008
pmed11.txt	300	5	7696	7696	7696	7696	3375	0
pmed12.txt	300	10	6634	6634	6634	6634	2816	0
pmed13.txt	300	30	4374	4374	4374	4374	3236	0
pmed14.txt	300	60	2968	2968	2970.199951	2973	4815	0,0007
pmed15.txt	300	100	1729	1733	1735	1737	10362	0,0035
pmed16.txt	400	5	8162	8162	8162	8162	16214	0
pmed17.txt	400	10	6999	6999	6999	6999	8215	0
pmed18.txt	400	40	4809	4809	4811.399902	4815	8250	0,0005
pmed19.txt	400	80	2845	2847	2847.800049	2849	15213	0,0010
pmed20.txt	400	133	1789	1790	1792.800049	1795	33767	0,0021
pmed21.txt	500	5	9138	9138	9138	9138	31278	0
pmed22.txt	500	10	8579	8579	8579	8579	20497	0
pmed23.txt	500	50	4619	4619	4621.600098	4628	17171	0,0006
pmed24.txt	500	100	2961	2964	2967	2968	31691	0,0020
pmed25.txt	500	167	1828	1830	1833.800049	1837	70070	0,0032
pmed26.txt	600	5	9917	9917	9917	9917	57876	0
pmed27.txt	600	10	8307	8307	8307	8307	37016	0
pmed28.txt	600	60	4498	4503	4505.399902	4508	34265	0,0016
pmed29.txt	600	120	3033	3045	3047	3049	63261	0,0046
pmed30.txt	600	200	1989	2018	2018	2018	106592	0,0146
pmed31.txt	700	5	10086	10086	10086	10086	71900	0
pmed32.txt	700	10	9297	9297	9297	9297	110461	0
pmed33.txt	700	70	4700	4702	4707	4712	52379	0,0015
pmed34.txt	700	140	3013	3019	3021	3024	114019	0,0027
pmed35.txt	800	5	10400	10400	10400	10400	95450	0
pmed36.txt	800	10	9934	9934	9934	9934	88697	0
pmed37.txt	800	80	5057	5060	5064	5068	59792	0,0014
pmed38.txt	900	5	11060	11060	11060	11060	3375	0
pmed39.txt	900	10	9423	9423	9423	9423	319090	0
pmed40.txt	900	90	5128	5133	5135.399902	5139	167431	0,0014

Tabela 5: Valores da metaheurística para o parâmetro $\alpha = 0,5$.

De acordo com a tabela 6, para o valor de $\alpha = 0,7$ temos que a média do *gap* é dada por $med = 0,0012$

Instancia			Ótimo	Metaheurística $\alpha = 0,7$				
nome	n	p	F	F_{min}	F_{med}	F_{max}	tempo	gap (%)
pmed1.txt	100	5	5819	5819	5819	5819	75	0
pmed2.txt	100	10	4093	4093	4100.200195	4102	101	0,0018
pmed3.txt	100	10	4250	4250	4250	4250	132	0
pmed4.txt	100	20	3034	3034	3034	3034	157	0
pmed5.txt	100	33	1355	1358	1358	1358	250	0,0022
pmed6.txt	200	5	7824	7824	7824	7824	633	0
pmed7.txt	200	10	5631	5631	5635.799805	5639	734	0,0009
pmed8.txt	200	20	4445	4445	4445	4445	643	0
pmed9.txt	200	40	2734	2737	2737.199951	2738	1193	0,0012
pmed10.txt	200	67	1255	1255	1256.400024	1259	2810	0,0011
pmed11.txt	300	5	7696	7696	7696	7696	3388	0
pmed12.txt	300	10	6634	6634	6634	6634	2812	0
pmed13.txt	300	30	4374	4374	4374	4374	3452	0
pmed14.txt	300	60	2968	2968	2968.800049	2972	5511	0,0003
pmed15.txt	300	100	1729	1732	1733.800049	1736	11010	0,0028
pmed16.txt	400	5	8162	8162	8162	8162	16533	0
pmed17.txt	400	10	6999	6999	7000.600098	7003	8130	0,0002
pmed18.txt	400	40	4809	4809	4810	4811	8011	0,0002
pmed19.txt	400	80	2845	2845	2847.399902	2850	15971	0,0008
pmed20.txt	400	133	1789	1789	1790.800049	1792	33833	0,0010
pmed21.txt	500	5	9138	9138	9138	9138	30984	0
pmed22.txt	500	10	8579	8579	8579	8579	22022	0
pmed23.txt	500	50	4619	4619	4624.600098	4629	16099	0,0012
pmed24.txt	500	100	2961	2964	2967.199951	2969	32720	0,0021
pmed25.txt	500	167	1828	1832	1834.800049	1836	74490	0,0037
pmed26.txt	600	5	9917	9917	9917	9917	58712	0
pmed27.txt	600	10	8307	8307	8307	8307	35170	0
pmed28.txt	600	60	4498	4505	4506.600098	4509	35193	0,0019
pmed29.txt	600	120	3033	3045	3048.600098	3051	64269	0,0051
pmed30.txt	600	200	1989	2018	2018	2018	104099	0,0146
pmed31.txt	700	5	10086	10086	10086	10086	70498	0
pmed32.txt	700	10	9297	9297	9297.799805	9301	109330	0,0001
pmed33.txt	700	70	4700	4704	4710	4719	49006	0,0021
pmed34.txt	700	140	3013	3016	3019.199951	3023	113171	0,0021
pmed35.txt	800	5	10400	10400	10400	10400	94031	0
pmed36.txt	800	10	9934	9934	9934	9934	86517	0
pmed37.txt	800	80	5057	5058	5061.799805	5065	64703	0,0009
pmed38.txt	900	5	11060	11060	11060	11060	34367	0
pmed39.txt	900	10	9423	9423	9423	9423	314571	0
pmed40.txt	900	90	5128	5132	5136.600098	5145	164673	0,0017

Tabela 6: Valores da metaheurística para o parâmetro $\alpha = 0,7$.

De acordo com a tabela 7, para o valor de $\alpha = 0,9$ temos que a média do *gap* é dada por $med = 0,0011$

Instancia			Ótimo	Metaheurística $\alpha = 0,9$				
nome	n	p	F	F_{min}	F_{med}	F_{max}	tempo	gap (%)
pmed1.txt	100	5	5819	5819	5819	5819	80	0
pmed2.txt	100	10	4093	4093	4093	4093	108	0
pmed3.txt	100	10	4250	4250	4250	4250	117	0
pmed4.txt	100	20	3034	3034	3037.199951	3038	170	0,0011
pmed5.txt	100	33	1355	1358	1358	1358	238	0,0022
pmed6.txt	200	5	7824	7824	7824	7824	619	0
pmed7.txt	200	10	5631	5631	5631	5631	645	0
pmed8.txt	200	20	4445	4445	4445	4445	713	0
pmed9.txt	200	40	2734	2737	2737.199951	2738	1263	0,0012
pmed10.txt	200	67	1255	1256	1256	1256	2930	0,0008
pmed11.txt	300	5	7696	7696	7696	7696	3415	0
pmed12.txt	300	10	6634	6634	6634	6634	2951	0
pmed13.txt	300	30	4374	4374	4374	4374	3252	0
pmed14.txt	300	60	2968	2968	2969	2971	5076	0,0003
pmed15.txt	300	100	1729	1733	1733.400024	1734	9611	0,0025
pmed16.txt	400	5	8162	8162	8162	8162	16222	0
pmed17.txt	400	10	6999	6999	6999.799805	7003	8315	0,0001
pmed18.txt	400	40	4809	4809	4810.399902	4812	8891	0,0003
pmed19.txt	400	80	2845	2846	2847.399902	2850	14853	0,0008
pmed20.txt	400	133	1789	1789	1791.199951	1792	32700	0,0012
pmed21.txt	500	5	9138	9138	9138	9138	30847	0
pmed22.txt	500	10	8579	8579	8579	8579	21078	0
pmed23.txt	500	50	4619	4619	4619	4619	15854	0
pmed24.txt	500	100	2961	2964	2966	2968	30314	0,0017
pmed25.txt	500	167	1828	1833	1835	1838	71374	0,0038
pmed26.txt	600	5	9917	9917	9917	9917	56937	0
pmed27.txt	600	10	8307	8307	8307	8307	35520	0
pmed28.txt	600	60	4498	4504	4506.600098	4510	37597	0,0019
pmed29.txt	600	120	3033	3043	3046.199951	3049	60370	0,0044
pmed30.txt	600	200	1989	2018	2018	2018	105649	0,0146
pmed31.txt	700	5	10086	10086	10086	10086	70822	0
pmed32.txt	700	10	9297	9297	9297	9297	109138	0
pmed33.txt	700	70	4700	4704	4704.399902	4706	51951	0,0009
pmed34.txt	700	140	3013	3015	3018.399902	3022	117070	0,0018
pmed35.txt	800	5	10400	10400	10400	10400	95306	0
pmed36.txt	800	10	9934	9934	9934	9934	87706	0
pmed37.txt	800	80	5057	5061	5063.399902	5068	63578	0,0013
pmed38.txt	900	5	11060	11060	11060	11060	34195	0
pmed39.txt	900	10	9423	9423	9423	9423	315636	0
pmed40.txt	900	90	5128	5133	5135.799805	5139	147053	0,0015

Tabela 7: Valores da metaheurística para o parâmetro $\alpha = 0,9$.