METAHEURÍSTICAS GRASP E VNS PARA PROBLEMA DE GRAFO COM RESTRIÇÕES DE CAPACIDADE E CONEXIDADE

Nádia Mendes dos Santos
Instituto Federal do Piauí
Praça da Liberdade, 1597 – Centro – Teresina (PI) - 64000-040 - Brasil
nadiaphb@gmail.com

Luiz Satoru Ochi
Instituto de Computação – Universidade Federal Fluminense
Rua Passo da Pátria, 156 – Bloco E 3º Andar – Niterói (RJ) – CEP: 24210-240, Brasil
luiz.satoru@gmail.com

José André Brito
Escola Nacional de Ciências e Estatística
Rua André Cavalcanti, 106 – Santa Teresina – Rio de Janeiro (RJ) – CEP: 20231-050, Brasil
jambrito@gmail.com

Gustavo Silva Semaan
Universidade Federal Fluminense, Instituto do Noroeste Fluminense de Educação Superior.
Av. João Jasbick, s/n – Aeroporto - 24210240 - Santo Antônio de Pádua, RJ – Brasil
gsemaan@ic.uff.br

RESUMO

São propostos novos métodos de resolução para o problema de definição de áreas de ponderação. Este trabalho traz propostas para a resolução do problema de agrupamento com restrições de capacidade e de conexidade. Mais especificamente, foram desenvolvidas versões diferentes de algoritmos heurísticos que buscavam a recuperação de informações através do uso das metaheurísticas GRASP e VNS. No que concerne ao algoritmo GRASP foi implementada uma versão utilizando o modelo clássico. O algoritmo VNS foi implementado em seu modelo tradicional e com duas variações, ou seja, um VNS (Variable Neighbourhood Search) padrão, um GVNS (General Variable Neighbourhood Search) e um RVNS (Reduced Variable Neighbourhood Search). De forma a avaliar a aplicabilidade e a eficiência desses algoritmos, são apresentados resultados de experimentos computacionais com exemplos reais do Censo Demográfico 2010 do IBGE (Instituto Brasileiro de Geografia e Estatística).

PALAVARAS CHAVE. GRAFOS, GRASP, VNS

1. Introdução

Em linhas gerais, obter uma solução para um problema de agrupamento corresponde a agrupar os objetos de uma base de dados, de maneira que os grupos formados (clusters) representem uma configuração em que cada elemento possua uma maior similaridade com elemento do mesmo cluster. Além disso deve ser maximizada a diferença entre elementos de grupos distintos. Em particular, neste trabalho será tratado um problema de agrupamento em grafos, conhecido na literatura como problema de particionamento de grafos [Semann, 2010]. Formalmente, o problema pode ser descrito como: dado um conjunto de n elemento $X=\{x_1, x_2, ..., x_n\}$ no qual cada objeto x_i possui p atributos $x_i = \{x_{i1}, x_{i2},...,x_{ip}\}$, que dimensionam as características do objeto, deve-se construir k grupos C_j $(j=1,...,k)$ a partir do conjunto X, respeitando as seguintes restrições:

$$C_j \neq \emptyset \text{ para } j = 1,...,k \quad (1)$$

$$C_l \cap C_j \neq \emptyset \text{ para l, j = 1, ..., k e l} \neq \text{ j} \quad (2)$$

$$\bigcup_j^k C_j = X \quad (3)$$

Quando o valor de k é definido a priori, temos um problema clássico de agrupamento (assim como no presente trabalho). Se k não for definido a priori, temos um problema de agrupamento automático [Cruz, 2010] e a identificação da quantidade ideal de grupos (k) faz parte da resolução do problema. Fixado o número de grupos e fornecido o número de objetos n, a quantidade de soluções viáveis $(Q(n,k))$ para o problema de agrupamento clássico pode ser obtida com a utilização da Equação 4 [Chiou and Lan, 2001] (referenciar como número de stirling de segundo tipo):

$$Q(n,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i \binom{k}{i}(k-i)^n \quad (4)$$

O presente trabalho traz uma proposta de resolução de particionamento de grafos que agrega as restrições de capacidade e de conexidade. Este trabalho está dividido em cinco seções, quais sejam: Na Seção 2 tem-se a apresentação do problema, onde será descrito detalhadamente o estudo de caso deste trabalho. Na Seção 3, são relatadas as metaheurísticas GRASP padrão, além do VNS padrão e suas variações (GVNS e RVNS). Na Seção 4 e 5 são demonstrados os procedimentos de construção e de busca local utilizados nos algoritmos propostos neste trabalho. A Seção 6 traz um conjunto de resultados computacionais obtidos a partir da implementação das metaheurísticas. E finalizando o trabalho são apresentados as conclusões gerais.

2. Descrição do Problema

No problema abordado no presente trabalho deseja-se particionar um conjunto de n objetos em k clusters de forma que cada cluster seja contíguo. Este problema de agrupamento ocorre, por exemplo, em situações nas quais a regionalização geográfica torna-se necessária, isto é, nos casos dos censos demográficos e de pesquisa socioeconômicas [Openshaw, 1977].

No contexto do censo geográfico, por exemplo, os objetos podem estar associados a domicílios, a setores censitários, às áreas de ponderação (APONDs) e a municípios. No caso do problema de definição das áreas de ponderação, para a formação dos clusters são consideradas as restrições de conexidade, no sentido de que essas áreas sejam constituídas por regiões geograficamente limítrofes, e de homogeneidade, segundo um conjunto de p atributos associados às características populacionais e de infraestrutura conhecidas, denominados indicadores de áreas de ponderação (APONDs). Uma APOND corresponde a uma unidade geográfica formada por agrupamentos mutuamente exclusivos de setores censitários, sendo esses, por sua vez, formados por conjuntos de domicílios. Essas áreas são utilizadas para se estimar informações para a população. O tamanho destas áreas, em termos de número de domicílios e/ou de população, não pode ser muito reduzido sob pena de perda de precisão das estimativas. [Censo Demográfico, 2010].

As APONDs também representam os níveis geográficos mais detalhados da base operacional, sendo definidas de forma a atender às demandas por informações em níveis

geográficos menores que os municípios [Censo Demográfico, 2000].Para a formação das APONDs são considerados ainda os critérios de contiguidade e de homogeneidade. No que se refere à contiguidade, dados dois setores quaisquer A e B dentro de uma mesma APOND, deve ser possível sair de A e chegar B (ou vice-versa) passando apenas por outros que estejam nesta mesma APOND. Quanto à homogeneidade, esta é definida em função de um conjunto de *p* variáveis associadas às características populacionais e da infra-estrutura conhecidas. Estas variáveis que representaremos por x_s, *s=1, ..., p* são chamadas indicadores de áreas de ponderação.

Para a formulação do problema tratado neste trabalho, é possível associar as informações relativas à contiguidade das regiões, bem como as informações dos totais associados a cada um dos *p* atributos e às distâncias d_{ij} (Fórmula 5), a um grafo *G= (V, E)*. Cada vértice $i \epsilon V$ do grafo correspondente a uma APOND e os seus atributos, inclusive o atributo capacidade, que são utilizados para validação da restrição de capacidade. A Equação (5) apresenta d_{ij} que representa o grau de homogeneidade entre as APONDs que serão agregadas, quando duas áreas *i* e *j* são vizinhas:

$$d_{ij} = \sqrt{\sum_{l=1}^{p} (x_{il} - x_{jl})^2} \qquad (5)$$

Com base nas informações anteriores, pode-se observar que o problema de formação de áreas de ponderação agregadas pode ser mapeado em um problema de agrupamento em grafos com as restrições de conexidade (contiguidade) e capacidade (total de determinada variável). Desta forma, é necessário associar tanto as informações relativas à contiguidade (conexidade) das APONDs, quanto as informações da capacidade dos vértices associado a um grafo *G = (V, E)*. Cada vértice $i \epsilon V$ do grafo corresponde a uma APOND e contém o valor associado à variável que determina sua capacidade. Além disso, se duas APONDs *i* e *j* são ditas vizinhas, existe uma aresta *e = (i, j)* de *E*, tal que esta possui associado o valor da distância d_{ij}. Considerando a restrição de conexidade do grafo G, uma solução natural para o problema consistirá em construir uma árvore geradora mínima $T = (V, E^* \subset E)$ obtida a partir de *G,* considerando os menores valores de d_{ij}. Fornecida a árvore *T*, e um número *k* de clusters, retira-se *(k − 1)* arestas de *T*, definindo um conjunto de *k* subárvores T_j, *j = 1,... , k*, que também são conexas.

Cada uma destas subárvores corresponderá a um possível *cluster*. A propriedade de conexidade observada para cada uma das subárvores possibilita o cumprimento imediato da restrição de contiguidade das APONDs. Desta forma, uma possível abordagem para solução do problema consistirá, então, em particionar *T* em *k* subárvores T_j, *j = 1,..., k*, associadas aos *clusters*, que satisfaçam a restrição de capacidade e que resulte no menor valor possível de uma particular função objetivo que mede o grau de similaridade nos clusters.

No caso do particionamento, de forma a avaliar a qualidade dos clusters obtidos, considerando que os critérios de capacidade e contiguidade sejam respeitados, utilizamos a função objetivo descrita pela Equação (6):

$$f = \sum_{l=1}^{p} \sum_{i=1}^{n_c} (x_{il} - \overline{x_l})^2 \qquad (6)$$

Em que, $\overline{x_j} = \frac{\sum_{i=1}^{n_c} x_{il}}{n_c}$ para *p* atributos, n_c e x_{il} *l do objeto i.*

Para a formação de cada dois novos agrupamentos (*clusters*) avalia-se o valor de *f*, considerando a remoção de cada uma das arestas associadas aos agrupamentos anteriores. Quanto o menor o valor de *f*, melhor será a solução.

As soluções propostas neste trabalho se referem à construção de uma Árvore Geradora Mínima *T* (AGM) mediante a aplicação do algoritmo de *Kruskal*. A vantagem da utilização de uma AGM é o fato da adjacência espacial (contiguidade) exigida pelo problema ser atendida de

forma imediata, uma vez que por definição uma árvore representa um grafo conexo e qualquer aresta removida da mesma produzirá duas estruturas também conexas. Tal árvore é construída a partir de um grafo *G* que contém as informações das APONDs. Em seguida, para gerar os *k* clusters é necessário remover *(k – 1)* arestas de *T*. Ou seja, em cada iteração *j (j=1,..., k – 1)*, uma aresta e_j é removida de uma subárvore T_{j-1}, produzindo duas novas subárvores (dois novos *clusters*) T^1_j e $T^{2(j=1...,k-1)}_j$. Tal procedimento corresponde a uma estratégia de divisão hierárquica na qual inicialmente todas as APONDS pertencem a um único cluster. No entanto, o processo de particionamento de uma AGM consiste em, selecionar o *cluster* T_i para divisão, buscar a aresta existente entre vértices desse que possui maior $Custo_e$ (custo de cada aresta que compõe a AGM) e removê-la da árvore. Para isto, deve-se remover cada uma das arestas de T_i e aplicar a Equação 6 em cada um dos novos clusters. Observando que o valor do custo de remoção da aresta corresponde à diferença entre a função objetivo do cluster corrente e a soma da função objetivo de cada um dos novos clusters, sendo a soma da função objetivo dos novos clusters correspondente à soma dos quadrados dos desvios das duas subárvores T^1_j e T^2_j calculados os valores médios dos m atributos. Os dois novos clusters formados serão homogêneos à medida que o custo da aresta $Custo_e$ (7) a ser removida seja o maior possível [Neves, 2003],[Semaan, 2010]

$$Custo_e = f(T_i) - (f(T_i^1) + f(T_i^2)) \qquad (7)$$

Segundo Semaan (2010), o valor do custo de remoção da aresta corresponde à diferença entre a função objetivo do *cluster* atual e a soma da função objetivo de cada um dos novos *clusters*. Quanto mais homogêneos forem os dois novos *clusters*, maior será o valor de $Custo_e$ definida pela equação (7)

A estrutura de representação adotada neste trabalho foi a group-*number,* também utilizada no trabalho de Trindade e Ochi em 2006. Nesta estrutura cada índice do vetor representa o vértice do grafo e seu conteúdo, um número inteiro positivo entre *1* e *k*, representa o *cluster* ao qual o vértice pertence [Dias, 2004]. Ainda na representação da solução, os algoritmos implementados neste trabalho permitem que este valor seja informado como parâmetro de entrada.

Além deste vetor, a estrutura de dados utilizada para representar as soluções também possui um atributo referente à capacidade do *cluster*, obtida através do somatório do atributo associado à capacidade dos vértices do *cluster*. Assim é possível verificar quais *clusters* estão penalizados, ou seja, que possuem um valor de capacidade inferior ao limite pré-estabelecido , e consequentemente, quantas penalidades a solução possui [Semaan, 2010]. Na próxima Seção tem-se a descrição das metaheurísticas utilizadas neste trabalho.

3. Metaheurísticas

As metaheurísticas ganharam muita popularidade nas últimas décadas e representam uma família de técnicas de otimização que têm sido aplicadas com êxito em diversos problemas de otimização. Assim, os algoritmos de metaheurísticas podem ser simplesmente vistos como uma repetição de dois passos simples (Ibaraki et al., 2005): geração de soluções, e aperfeiçoamento das soluções pela busca local.

Os algoritmos propostos neste trabalho utilizam como base as metaheurísticas GRASP e VNS. Primeiramente, o GRASP sem alterações estruturais foi proposto para a solução do problema de clusterização com restrição de conexidade e capacidade. Os algoritmos também foram implementados utilizando como base o Variable Neighbourhood Search (VNS), inicialmente no seu modelo tradiconal, ou seja, um Variable Neighbourhood Search (VNS) padrão e duas variações que sejam: um General Variable Neighbourhood Search (GVNS) e um Reduced Variable Neighbourhood Search (RVNS).

3.1 GRASP

O GRASP (*Greedy Randomized Adaptative Search Procedure*) é uma metaheurística que tem sido usada com sucesso na resolução de problemas de otimização combinatória nas mais diversas áreas. Sua implementação é simples, usando-se algoritmos para construção de soluções e de busca local desenvolvidos para serem usados em outras abordagens. rata-se de uma metaheurística sequencial iterativa, onde cada iteração consiste em duas etapas: a fase construtiva e a fase de busca local. Após cada iteração, um ótimo local é encontrado e a melhor solução de todas as iterações é retornada como solução final. O GRASP combina as abordagens gulosa e aleatória. A fase construtiva constrói uma solução viável através de uso de uma função de aleatoriedade e gulosa. A parte gulosa da função visa gerar uma solução factível de melhor custo (baixo ou alto custo, dependendo da aplicação). O componente aleatório é incluído para explorar regiões diversas do espaço de soluções e é uma das chaves da efetividade do GRASP. O melhor ótimo local dentre todas as buscas locais é retornado como solução da metaheurística [Feo & Resende, 1995].

3.2 VNS PADRÃO

Em 1997, Hansen e Mladenović propuseram uma metaheurística que se baseava em uma troca sistemática de vizinhanças, associada a um algoritmo aleatório na determinação de pontos iniciais da busca local, chamada de Busca em Vizinhança Variável, conhecida na literatura em inglês como *Variable Neighborhood Search* (VNS). Contrariamente a outras metaheurísticas baseadas em métodos de busca local, VNS não segue uma trajetória, mas explora incrementalmente vizinhanças distantesa solução corrente, partindo da solução atual para a nova se, e somente se, uma melhora ocorrer.

Diferentemente de outras metaheurísticas, a metaheurística VNS sico e suas extensões são simples e requerem poucos, e às vezes nenhum parâmetro [MLADENOVIC 1995] [MLADENOVIC AND HANSEN, 1997]. Ou seja, a metaheurística de busca em vizinhança variável é uma extensão de um algoritmo de busca local que utiliza a estratégia de mudança de vizinhança para sair de soluções ótimas locais.

O algoritmo VNS trabalha com várias vizinhanças. Portanto, seja Nk o conjunto finito de estruturas de vizinhanças pré-selecionadas, com ($k = 1, ..., k_{max}$) e $N_k(s)$ o conjunto de soluções na *k-é*sima vizinhança de *s*. Além do conjunto N_k de vizinhanças, usa-se a função de avaliação *f*, a ser minimizada. Uma solução ótima s (ótimo global) é uma solução viável de tal maneira que para cada solução viável s' $\in$ S, tem-se que $f(s) < f(s')$.

O algoritmo VNS proposto neste trabalho é composto por duas fases, a saber: Na primeira fase é construída uma Árvore Geradora Mínima (AGM) e, em seguida, realizado o seu particionamento em *n* clusters de forma que satisfaça às restrições de conexidade e capacidade tratadas neste trabalho. A segunda fase é caracterizada pela aplicação de um procedimento VNS na solução com menor valor de função objetivo (Equação 6) resultante da etapa anterior.

3.3 GVNS

A estratégia GVNS (*General Variable Neighborhood Search*) ou Busca Geral em Vizinhança Variável é inspirada na ideia de buscar boas soluções por meio de estruturas de vizinhanças diferentes. Uma das motivações desta abordagem é que um ótimo global é, consequentemente, um ótimo local para todas as possíveis vizinhanças de uma solução. Um algoritmo similar ao desenvolvido neste trabalho já existe na literatura com bons resultados.

A metaheurística GVNS, pode ser divida em duas fases, a saber: perturbação e busca local. Na fase de perturbação, o algoritmo gera uma solução aleatória em uma dada vizinhança (exploração estocástica das estruturas de vizinhança) e na fase de busca local, é aplicado o método VND. Uma vez que o procedimento VND é método de refinamento que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhanças, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada.

O algoritmo GVNS baseia-se nos critérios de busca local em torno de uma solução corrente com intuito de explorar novas regiões, o que é análogo ao algoritmo VNS. No entanto, a mudança fundamental está na fase de melhoria da busca local usando o VND. A GVNS tem sido um dos métodos que mais obteve êxito recentemente, por exemplo.

3.4 RVNS

Uma redução da busca em vizinhança variável original foi também proposta por Mladenovic 1997, na qual não se tem um método de busca local para melhorar a solução gerada *s'* a cada iteração [MLADENOVIC AND HANSEN 1997]. Tal fato pode melhorar o tempo do algoritmo VNS em casos em que um método de busca local tem um custo computacional muito elevado. Essa redução é chamada *Reduced Variable Neighborhood Search* (RVNS).

Esse tipo de algoritmo VNS chamado de RVNS [MARTINS 2009]**,** é inspirado em dois aspectos fundamentais no processo de busca relacionados com a intensificação e a diversificação.

No algoritmo RVNS um conjunto de estruturas de vizinhanças N_k ($k=1, ..., k_{max}$) será considerado ao redor da solução atual *s* (o qual pode ser ou não um ótimo local). Geralmente essas vizinhanças estão aninhadas, isto é, os elementos da vizinhança N_1 também são elementos na vizinhança N_2 e assim sucessivamente.

Observa-se que o algoritmo RVNS aplicado ao problema tratado neste trabalho segue os mesmos passos do algoritmo VNS, utilizando até as mesmas estruturas de vizinhanças. Porém, possibilita uma escolha de vizinhos mais dinâmica selecionando vizinhos de todas as estruturas de vizinhança (diversificação) e priorizando a primeira estrutura de vizinhança (intensificação) nas fases iniciais da busca. O algoritmo de RVNS também é capaz de identificar regiões novas promissoras a partir de um ponto de ótimo local.

4. Procedimento de Construção

Com o objetivo de realizar a construção de soluções iniciais foram considerados os dois procedimentos propostos por Semaan 2010 e utilizados também em [Santos, 2012] [Santos, 2013], além de um procedimento Construtor 3 proposto por Santos 2014.

Os procedimentos de construção de soluções iniciais desenvolvidos neste trabalho geram *clusters* através do particionamento de uma AGM T . Esta árvore foi construída a partir do grafo *G* associado ao problema de clusterização com restrição de capacidade e conexidade, em *k clusters* (subárvore). Portanto, os *k clusters* são definidos através da remoção de $k-1$ arestas de *T*, sendo que a remoção de cada aresta produz duas novas subárvores, ou seja, dois novos *clusters* definidos como T_a e T_b.

Em todos os procedimentos de construção de soluções propostos neste trabalho, a restrição de conexidade é garantida. Porém, apesar dos procedimentos terem como objetivo principal a busca de soluções válidas que atendam concomitantemente às restrições de conexidade e de capacidade mínima por *cluster*, esta última restrição pode não ser atendida. Nesse caso, em algumas situações, pode ser impossível a formação de soluções que atendam ambas as restrições. ssim, utilizando a ideia de particionamento da AGM, foram implementados três estratégias de construtores de soluções iniciais que estão descritas a seguir:

4.1 Construtor 1

A primeira estratégia do procedimento construtor de soluções iniciais realiza o particionamento da AGM respeitando somente a restrição de conexidade, sem considerar a restrição de capacidade, objetivando, portanto, somente a minimização da função objetivo. O procedimento construtor é aplicado em duas etapas que são executadas ($k-1$) vezes para a obtenção de *k clusters*:

A primeira etapa consiste na **seleção do cluster a ser particionado** . Nessa etapa, seleciona-se o *cluster* com o menor grau de homogeneidade (considerando a Equação 6), excetuando-se a primeira iteração do procedimento construtor**,** uma vez que só existe 1 (um) *cluster* composto pela AGM. A seguir, define-se qual aresta será **removida** de forma a produzir dois novos *clusters* a partir desta etapa. O valor do custo da remoção da aresta é obtido através

da Equação 11. Nesse caso, quanto maior o valor de $Custo_a$ mais homogêneos são os dois novos *clusters formados.*

Nesse procedimento construtor foi considerada uma ideia similar à apresentada no procedimento de construção metaheurística GRASP. Ou seja, cria-se uma Lista Restrita de Candidatos (LRC) em que serão incluídas as α arestas que possuem os maiores valores de $Custo_{a.}$ Particiona-se, então, a AGM através da remoção aleatória de uma das arestas inseridas na LRC. E, enquanto a quantidade de clusters pré-estabelecida não for alcançada deve-se repetir os seguintes passos:

- Selecionar o cluster com maior valor da função objetivo f (Equação 6);
- Calcular o $Custo_a$ para todas as arestas do *cluster* selecionado;
- Definir a LRC com as α arestas que possuem os maiores valores de $Custo_a$
- Particionar o *cluster* selecionado através da remoção aleatória de uma aresta de LRC.

A adoção desse procedimento possibilita uma diversificação das soluções, ou seja, encontrar soluções potencialmente válidas, evitando que o algoritmo tenha um comportamento guloso. Embora exista a restrição de capacidade mínima, ou seja, um limite inferior para o somatório do atributo associado à capacidade dos vértices do *cluster*, o Construtor 1 foca apenas na restrição de conexidade. O procedimento Construtor 2, que será visto a seguir, trata da restrição de conexidade, além de garantir também a capacidade mínima.

4.2 Construtor 2

Neste segundo procedimento, é realizado o particionamento da AGM respeitando as restrições de conexidade e de capacidade mínima. No que se refere à restrição de capacidade mínima, o construtor 2 (dois) trabalha na tentativa de obter clusters que atendam essa restrição, ação esta que nem sempre é bem sucedida. O procedimento construtor 2 (dois) consiste de duas etapas descritas a seguir:

Na primeira etapa é gerada uma subárvore principal contida na AGM. A concepção de subárvore principal está associada à existência, na AGM, de um caminho simples sem ciclos, entre dois vértices dessa árvore, ou seja, toma-se um vértice V_i como a origem desse caminho e um vértice Vj como o término. A seleção dos vértices V_i e V_j é feita aleatoriamente entre todos os vértices pertencentes a AGM, para que se possa a partir dessa escolha calcular a distância entre os V8 e V9 (Figura 1), devido a essa arbitrariedade na escolha dos vértices fica impossível garantir que esse seja o menor caminho entre eles.

A partir da AGM, são selecionados de forma aleatória dois vértices, ilustrados em destaque na Figura 1 abaixo:

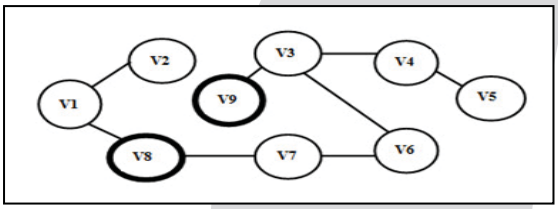


Fig. 1. Seleção da subarvore principal da AGM

A partir da AGM, são selecionados de forma aleatória dois vértices, ilustrados em destaque na Figura 1 abaixo:

Os vértices selecionados de forma aleatória (V_8 e V_9) irão formar a subárvore principal da AGM juntamente com os outros vértices que se encontram no caminho existente entre eles. Para a identificação dos vértices que se encontram no caminho existente entre os vértices A e B, utilizou-se a Equação 8.

$$Di(A,B) - (Di(A, verticeX) + Di(A, verticeX)) = 0 \quad (8)$$

- A o vértice inicial a subárvore principal da AGM e- B o vértice final da subárvore principal da AGM;

- *verticeX* corresponde ao vértice sob consulta. Caso o lado esquerdo da Equação 8 seja igual a 0 (zero), confirma-se que o vértice *X* pertence ao caminho existente entre os vértices *A* e *B*. Esta distância que está sendo referenciada corresponde ao número de arestas em que cada aresta corresponde ao comprimento um.

Caso o lado esquerdo da *Equação 8* tenha um resultado diferente de 0 (zero), então o vértice *X* não pertence à subárvore principal da AGM entre os vértices A e B. Desta forma, a distância entre os vértices V_8 e V_9 , conforme o caminho existente na AGM, é a quantidade de arestas existentes entre esses dois vértices.

Assim, ao se definir quais vértices pertencem a subárvore principal da AGM através do cálculo da Equação 8, agrupam-se os vértices que não pertencem à subárvore principal serão agrupados ao vértice mais próximo pertencente a esta subárvore. Desta forma, o grafo deve ficar com um formato de uma "barbante" e os vértices pertencentes a subárvore principal possuirão o somatório das capacidades dos vértices a eles agrupados. Então, a subárvore principal toma a forma do grafo apresentado na Figura 2:

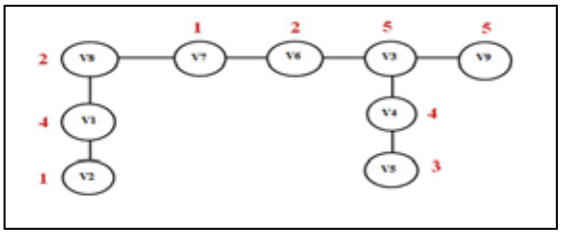


Fig. 2. Geração da subarvore principal da AGM

E a Figura 3.3 apresenta os vértices pertencentes à subárvore principal com o somatório dos valores das capacidades dos vértices a eles agrupados, os valores definidos na Tabela 3.1, a ilustração a seguir:

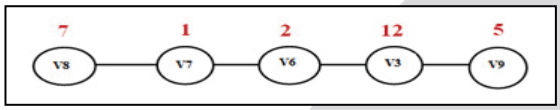


Fig. 3. Subárvore principal da AGM com os vértices capacitados

Na segunda etapa do procedimento construtor 2 (dois), é efetuado o particionamento da subárvore principal considerando a restrição da capacidade mínima até a obtenção de *k* clusters, considerando o exemplo da Figura 3.1, a partir dos vértices (V_8e V_9) que constituem a subárvore principal da AGM, são adicionados os grupos de vértices adjacentes, conforme a restrição de capacidade mínima. Portanto, os clusters são formados por vértices ou grupo de vértices, de acordo com a verificação do somatório, ou seja, se a sua capacidade for superior ao limite inferior pré-estabelecido. Assim, a restrição de capacidade mínima é garantida, o cluster é formado e um novo cluster é obtido a partir da junção dos vértices restantes da subárvore principal da AGM, até que sejam gerados os *k* clusters. Todavia, o particionamento continua sendo executado até que o número (*k*) de cluster(s) seja obtido ou, caso ainda exista algum vértice ou grupos de vértices não adicionados a nenhum cluster na subárvore principal da AGM.

4.3 Construtor 3

O procedimento Construtor 3 (três) trabalha com as restrições de conexidade e de capacidade mínima por *cluster* visando a geração de soluções válidas. O Construtor 3 (três) é executado em duas etapas, assim com o construtor 2, sendo gerada na primeira etapa uma subárvore principal contida na AGM.

Na segunda etapa deste procedimento construtor é particionada a subárvore principal**,** considerando a restrição de capacidade mínima. Tal particionamento é aplicado até a maximizar a

capacidade de um cluster considerando a restrição de conexidade existente não só na subárvore principal da AGM**,** mas também no Grafo Original.

A seguir, são apresentados os seis procedimentos de Busca Local que são propostos neste trabalho.

5. Procedimento de Busca Local

Neste trabalho foram implementadas seis versões de Algoritmos de Busca Local, que atuam na tentativa de eliminação de penalidades, referentes ao não atendimento da restrição de capacidade mínima por *cluster*, e na melhoria da solução no que concerne à minimização da função objetivo apresentada neste trabalho, propostos por Semaan 2010 e utilizados também em [Santos, 2012] [Santos, 2013]. Uma vez que esses procedimentos foram apresentados de maneira detalhada na literatura, seguem comentários de cada uma das versões:

- Procedimento de Busca Local 1: tem como objetivo eliminar penalidades. Assim, ele utiliza uma lista de arestas removidas da AGM e uma lista de clusters penalizados. O procedimento verifica a possibilidade de realizar migrações de vértices entre os clusters utilizando tanto a lista de arestas removidas quanto a lista de clusters penalizados. Para cada aresta da lista, são visualizados os clusters aos quais os seus vértices pertencem. Em seguida, são verificados os requisitos comuns a todas as buscas locais para que seja possível realizar a migração de vértices entre cluster.

- Procedimento de Busca Local 2: também tem como objetivo eliminar as penalidades. Esse procedimento verifica a possibilidade de realizar migrações de vértices entre os clusters considerando o Grafo Original, e não apenas a AGM. Tendo em vista que esse procedimento trabalha com o Grafo Original, há a necessidade de que seja verificado se as restrições de conexidade entre os vértices internos aos clusters estão sendo satisfeitas, ou seja, se os vértices internos aos clusters estão conexos.

- Procedimento de Busca Local 3: também verifica a possibilidade de realizar migrações de vértices entre o clusters considerando o Grafo Original e a AGM construída. E assim como o Busca Local 2, verifica se as restrições de conexidade entre os vértices internos aos clusters estão sendo satisfeitas, ou seja, se os vértices internos aos clusters são conexos.

- Procedimento de Busca Local 4: utiliza uma lista de arestas removidas da AGM para geração de clusters. Dessa forma, esse procedimento busca melhorar a qualidade da solução, ou seja, minimizar a função objetivo através da união e da divisão de clusters já construídos. São selecionados, aleatoriamente, os clusters aos quais os vértices das arestas removidas pertencem, podendo se unir em um cluster para, em seguida, este cluster ser dividido através da aplicação do procedimento construtor 1 (um).

- Procedimento de Busca Local 5: tem a finalidade de obter novas soluções não penalizadas (que atendam a restrição de capacidade mínima), considerando a AGM. Inicialmente utilizando uma lista de arestas removidas, escolhidas de forma aleatória, é realizada a união de dois clusters. Posteriormente, os clusters são divididos em dois novos clusters a partir de uma Lista Restrita de Candidatos (LRC) composta pela α arestas, que, se removidas, definem novos clusters. Estes clusters devem atender a restrição de capacidade mínima e, simultaneamente, devem possuir a maior diferença, em valor absoluto, considerando o somatório das capacidades dos clusters obtidos.

- Procedimento Busca Local 6: é aplicado com o objetivo de minimizar a função objetivo da solução. Dessa forma, a migração dos vértices entre clusters é realizada tendo o Grafo Original como base somente se houver a melhoria da qualidade da solução.

6. Resultados Computacionais

A presente seção apresenta os resultados computacionais obtidos a partir da aplicação dos algoritmos propostos neste trabalho, sendo todas as seis versões implementadas em linguagem C (no ambiente de desenvolvimento *NetBeans* 7.0.1). Todos os algoritmos foram executados em um computador dotado de 2GB de memória *RAM*, processador *Intel Core 2 Duo* com 1.67GHz, com sistema operacional *Windows* 7 *Ultimate de 64 bits*.

Com a finalidade de efetuar uma primeira avaliação dos algoritmos propostos neste trabalho, utilizou-se um conjunto de exemplos, geradas a partir dos dados do Censo Demográfico de 2010 (dados de uso público), descritas na Tabela 1. No total foram utilizados dois arquivos do tipo texto: o primeiro contendo as informações de vizinhança entre os objetos APONDs e o segundo arquivo contendo a identificação da área de ponderação e as variáveis utilizadas no cálculo das distâncias que são: o número do setor, o total de domicílios do setor, variável de proporção de domicílios com banheiro no setor e variável rendimento nominal total no setor.

Tabela 1 – Relação de instâncias utilizadas

Instâncias	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Nº Vértices	88	157	20	102	34	28	22	15
Nº Arestas	438	720	224	586	298	322	326	106

Foram implementados procedimentos e técnicas que permitiram utilizar o GRASP e VNS Padrão, GVNS e RVNS, considerando apenas a AGM construída e considerando, também, o grafo original conforme a descrição das siglas dos algoritmos a seguir:

- GRASP considerando AGM construída com *Kruskal* Padrão utilizando busca local 1, 4 e 5: (*i*) AGM1 com o Construtor 1(*ii*) AGM2 com o Construtor 2 e (*iii)* AGM3 com o Construtor 3;
- VNS considerando AGM construída com *Kruskal* Padrão utilizando busca local 1, 4 e 5: (*i*) VNS_AGM1 com o Construtor 1(*ii)* VNS_AGM2 com o Construtor 2 e (*iii*) VNS_AGM3 com o Construtor 3;
- GVNS considerando AGM construída com *Kruskal* Padrão utilizando busca local 1, 4 e 5: (*i*) GVNS_AGM1 com o Construtor 1(*ii*) GVNS_AGM2 com o Construtor 2 e (*iii*) GVNS_AGM3 com o Construtor 3;
- RVNS considerando AGM construída com *Kruskal* Padrão utilizando busca local 1, 4 e 5: (i) RVNS_AGM1 com o Construtor 1(ii) RVNS_AGM2 com o Construtor 2 e (iii) RVNS_AGM3 com o Construtor 3;
- GRASP considerando Grafo Original utilizando busca local 2, 3 e 6: (*i*) G1 com o Construtor 1(*ii*) G2 com o Construtor 2 e (*iii)* G3 com o Construtor 3;
- VNS considerando Grafo Original utilizando busca local 2, 3 e 6: (*i*) VNS1 com o Construtor 1(*ii)* VNS2 com o Construtor 2 e (*iii*) VNS3 com o Construtor 3;
- GVNS considerando Grafo Original utilizando busca local 2, 3 e 6: (*i*) GVNS1 com o Construtor 1(*ii*) GVNS2 com o Construtor 2 e (*iii*) GVNS3 com o Construtor 3;
- RVNS considerando Grafo Original utilizando busca local 2, 3 e 6: (i) RVNS1 com o Construtor 1(ii) RVNS2 com o Construtor 2 e (iii) RVNS3 com o Construtor 3;

Os algoritmos (seis versões) foram executados num total de 100 (cem) iterações cada um e o parâmetro alfa igual a 10 (dez). O parâmetro alfa é utilizado com o objetivo de atribuir aos procedimentos construtores um comportamento semi-guloso, mediante a manipulação do tamanho da LRC (Lista Restrita de Candidatos) do GRASP, ou seja, esse parâmetro representa o número de arestas que irão compor a LRC. Além dessas configurações, foi necessário informar aos algoritmos qual seria a quantidade *k* de clusters, que neste trabalho foi definida como 3. Somente soluções válidas foram consideradas, ou seja, aquelas em que as restrições de conexidade e de capacidade mínima por cluster foram atendidas.

As Figuras 4, 5, 6 e 7 apresentam gráficos com os quantitativos das soluções obtidas por algoritmo, classificados conforme a sua qualidade: Melhores (com *Gap* igual a 0%) representam as soluções que apresentaram menor valor da função objetivo, Razoáveis (com *Gap* maior do que 0% e menor ou igual a 10%) e Péssimos (*Gap* acima de 10%) representam as soluções que apresentaram o maior valor da função objetivo, a Equação (*Gap*) 9:

$$Gap = 100 * \frac{solução - solução_{best}}{solução_{best}} \quad (9)$$

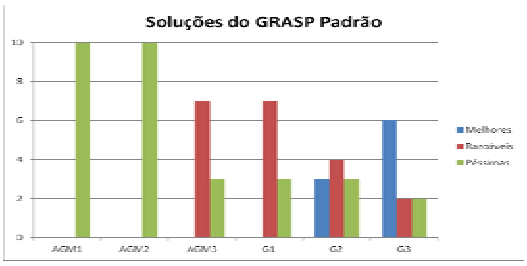


Fig. 4. Soluções dos algoritmos GRASP Padrão

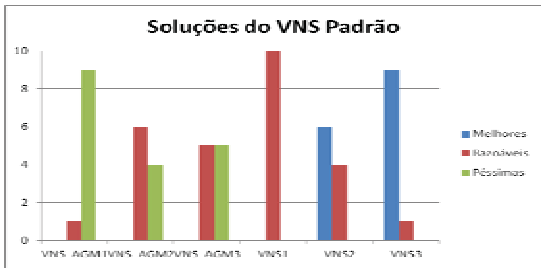


Fig. 5. Soluções dos algoritmos VNS Padrão

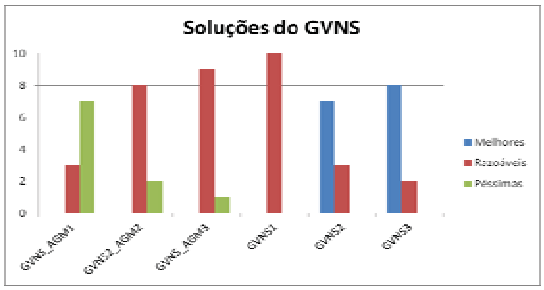


Fig. 6. Soluções dos algoritmos GVNS

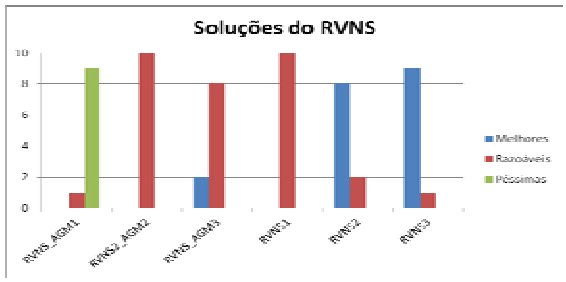
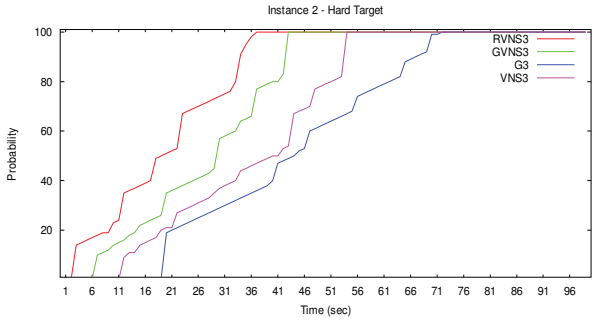


Fig. 7. Soluções dos algoritmos RVNS

O GRASP Padrão (Figura 4) utilizando a AGM apresentou os piores resultados durante os experimentos, sem nenhuma solução considerada Melhor e o com a maioria de suas soluções consideradas Péssimas. Por outro lado, a versão G3 usando o Grafo Original com o construtor 3 obteve o maior número de soluções Melhores e o menor número de soluções Péssimas. O VNS Padrão (Figura 5) utilizando a AGM também apresentou resultados ruins. Já o VNS3 que utilizou o Grafo Original e o construtor 3 apresentou o melhor resultado entre as versões apresentadas nos experimentos. O GVNS (Figura 6) utilizando a AGM só apresentou resultados ruins. Já o GVNS2 e GVNS3 que utilizou o Grafo Original com o construtor 2 e 3 apresentaram os melhores resultados entre as versões apresentadas nos experimentos. O RVNS (Figura 7) utilizando a AGM só apresentou resultados ruins. Já o RVNS2 e RVNS3 que utilizou o Grafo Original com o construtor 2 e 3 apresentaram os melhores resultados entre as versões apresentadas nos experimentos.

Uma análise de probabilidade empírica [Aiex et al. 2007] na instância 2 (grande número de de vértices/APONDs e arestas) foi realizada. Inicialmente cada algoritmo proposto foi aplicado 100 vezes em cada instância para se estabelecer dois alvos, quais sejam: um alvo difícil (com a menor função objetivo) e alvo fácil (com a maior função objetivo). Em seguida, cada versão de algoritmo foi aplicada 100 vezes na instância 2, tendo como critério de parada o alcance do alvo fácil ou tempo máximo (3600 segundos) de execução. Ou critério de parada o alcance do alvo difícil ou tempo máximo (3600 segundos) de execução.

Os gráficos da Figura 8 e 9 mostram o desempenho das versões do GRASP Padrão, VNS Padrão, GVNS e RVNS para a instância 2 que conseguiram atingir o alvo fácil e o difícil no limite de tempo estabelecido (3600 segundos).



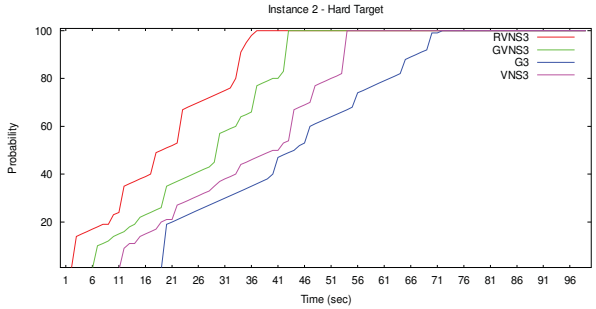


Fig. 8. Instância 2 – Alvo Fácil **Fig. 9.** Instância 2 – Alvo Difícil

Na instância 2 (de grande porte) todas as versões que utilizaram o Construtor 3 alcançaram o alvo, confirmando a influência do procedimento de construção na obtenção de solução inicial ótima. O RVNS utilizando o construtor 3 (RVNS3) alcançou o alvo fácil e díficil mais rapidamente do que a versão que utiliza o GVNS, VNS Padrão e o GRASP Padrão.

7. Conclusão

Os resultados apresentados demonstram que na análise empírica onde o critério de parada é o número de iterações, a versão RVNS3 apresenta os melhores resultados, mas exige tempos computacionais maiores que as versões de algoritmos propostos. E no segundo experimento, a mesma versão RVNS3 necessita de menos iterações para atingir bons resultados, seja o valor da função objetivo maior (alvo fácil) ou menor (alvo difícil). A partir de uma revisão da literatura, verificou-se que esta é primeira vez que o RVNS é utilizado para a solução do problema real de agrupamento com restrições de capacidade e de conexidade. E ainda que o construtor 3 apresentou os melhores resultados em relação as demais versões, o que comprova que a construção de soluções iniciais melhores o desempenho do algoritmo. Em trabalhos futuros, pretende-se implementar outras metaheurísticas (*Skewed* VNS e ILS) para desta forma tentar diminuir o tempo computacional do particionamento da AGM.

Referências

Aiex, R. M., Resende, M. G. C., e Ribeiro, C. C. TTT plots: a perl program to create time-to-target plots. *Optimization Letters*, 1:355_366, 2007.

Censo Demografico 2010- Resultados Gerais da Amostra – IBGE – ISSN 0104-3145.

Cruz, Marcelo D., O problema de Clusterização Automática. Dissertação de Mestrado de Ciência em Computação. *Universidade Federal do Rio de Janeiro (UFRJ). Engenharia de Sistemas e Computação (COPPE)*, 2010.

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

Hansen, P.; Mladenovic, n. e Moreno Pérez, J.: Variable neighborhood search, methods and applications. *Annals of Operations Research*, 175(1): 367-407, 2010.

Ibaraki, T.; Nonob e, K. & Yagiura, M. Metaheristics: Progress as real problem solvers. *Springer Verlag*, 2005.

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

Mladenovic, N.; Hansen, P. Variable Neighborhood Search. Computers Operations Research 24,1997.

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

Penna, P. H.V.; Subramanian, A e Ochi, L.S., An iterated local serach heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 2011.

Santos, Nádia M.; Brito, José A. M; Semaan Gustavo S.; Ochi, Luiz S. "Metaheuristic GRASP with path-relinking to the solution of the graph partitioning problem with capacity and connexity constraints." *Intelligent Data Engineering and Automated Learning-IDEAL 2012*. Springer Berlin Heidelberg, 2012. 630-641, 2012.

Santos, Nádia M.; Brito, José A. M; Semaan Gustavo S.; Ochi, Luiz S. Metaheurística híbrida para a solução de problema de particionamento de grafos com restrições de capacidade e de conexidade. *Universidad de Concepción. X OPTIMA, VI RED-M,* 2013.

Semaan, G.S.: Algoritmos Heurísticos para o Problema de Particionamento de Grafos com Restrições de Capacidade e Conexidade. Dissertação de Mestrado, *Universidade Federal Fluminense (UFF)*, Niterói – RJ, 2010.

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