DECOMPOSIÇÃO LAGRANGEANA DESBALANCEADA PARA O PROBLEMA DE ROTULAÇÃO CARTOGRÁFICA DE PONTOS

Sóstenes Pereira Gomes[1]
Luiz Antonio Nogueira Lorena[2]
Glaydston Mattos Ribeiro[3]
Geraldo Regis Mauri[4]
[1,2]Instituto Nacional de Pesquisas Espaciais
Av. dos Astronautas, 1758, São José dos Campos – SP, Brasil
[1]sostenes.gomes@gmail.com, [2]lorena@lac.inpe.br

[3]Universidade Federal do Rio de Janeiro
Av. Horácio Macedo, 2030, Rio de Janeiro – RJ, Brasil
glaydston@pet.coppe.ufrj.br

[4]Universidade Federal do Espírito Santo
Alto Universitário, s/n, Guararema, Alegre – ES, Brasil
mauri@cca.ufes.br

Resumo

Este trabalho aborda o Problema de Rotulação Cartográfica de Pontos (PRCP), o qual é um problema de otimização combinatória já demonstrado na literatura ser NP-difícil. É abordado o problema de obter bons resultados de particionamento, no processo de aplicação da Decomposição Lagrangeana para o PRCP, que um tema relevante, pois o número de arestas inter-partições (corte) geradas pode influenciar o número de restrições relaxadas, e consequentemente, a convergência do método de Subgradientes. Como a redução do número de partições, para obter um menor valor de arestas cortadas, pode não ser uma boa opção, por causa do aumento no tamanho dos subproblemas, é proposto o uso de tamanhos de partições desbalanceados. Isto resulta em melhores partições, com um número menor de cortes. Os resultados computacionais apresentam a solução ótima para todas as instâncias de teste utilizadas, com melhores tempos computacionais, comparados aos resultados da decomposição com partições balanceadas.

PALAVRAS CHAVE. Decomposição Lagrangeana, Programação Inteira, Rotulação Cartográfica

Área principal: Otimização Combinatória

Abstract

This paper concerns to the Point Feature Cartographic Label Placement Problem (PFCLP), which is a NP-hard combinatorial problem. We address the problem of obtain good results of partitioning in the process of applying the Lagrangean Decomposition for the PFCLP, which is a pertinent issue, since the generated number of inter-partitions edges (edge cut) can affect the number of relaxed constraints, and therefore, the convergence of the Subgradient method. Because reducing the number of partitions, to obtain a small number of edge cut, may be not a good option, due to the resulting increase in the size of generated subproblems, we propose the use of inhomogeneous partition sizes. That resulted in better partitions with a reduced number of edge cut. The computational results found the optimal solutions to all instances of test, obtained from literature, with small computer times when compared to CPLEX and the Lagrangean decomposition with balanced partitions.

Keywords: Lagrangean Decomposition, Integer programming, Cartographic label placement

Main area: Combinatorial Optimization

1. Introdução

O problema de automaticamente posicionar rótulos em mapas, diagramas, grafos ou qualquer objeto gráfico, conhecido como Problema de Rotulação Cartográfica (PRC), é um problema comum na área de Sistemas de Informações Geográficas (SIG) e é geralmente referenciado em três características diferentes: linhas (rios, estradas, etc.), polígonos (lagos, distritos, construções, etc.) e pontos (cidades, montanhas, etc.). No PRC, é necessário considerar que a sobreposição de rótulos deve ser evitada para uma melhor visibilidade do objeto gráfico.

O problema de rotular características de pontos, evitando a sobreposição de rótulos, é conhecido na literatura como o Problema de Rotulação Cartográfica de Pontos (PRCP). Christensen et al. (1995) propuseram uma padronização para a modelagem discreta das possíveis posições de rótulos, definidas como posições candidatas, em características de pontos. Esta padronização utiliza um valor de *rank* de prioridade indicando as melhores posições, onde o menor número representa a maior prioridade. A Figura 1 ilustra esta padronização, considerando quatro e oito posições candidatas.

Figura 1. Padronização cartográfica proposta por Christensen et al. (1995).

As possíveis sobreposições de rótulos, considerando a padronização de Christensen et al. (1995), podem ser modeladas como um grafo de conflitos $G = (V, A)$, onde o vértice $v_{i,j} \in V$, representa a posição candidata j do ponto i, $\forall\ i = \{1, ..., N\}$, $j = \{1, ..., P_i\}$, onde N é a quantidade de pontos no grafo e P_i é a quantidade de posições candidatas do ponto i. Adicionalmente, definem se as arestas $a(v_{i,j}, v_{k,t}) \in A$, representando uma sobreposição (conflito) entre as posições candidatas (i, j) e (k, t), caso estas posições sejam escolhidas para rotulação. Um exemplo de grafo de conflitos para um problema contendo quatro pontos, com quatro posições candidatas para cada ponto, seguindo a padronização de Christensen et al. (1995) é apresentado a seguir.

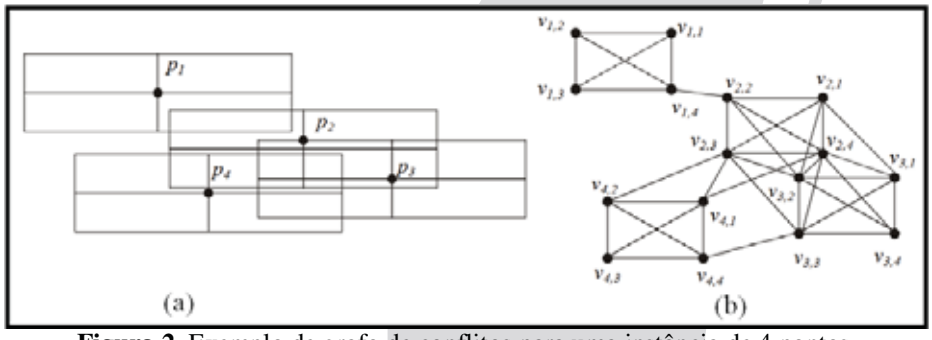

Figura 2. Exemplo de grafo de conflitos para uma instância de 4 pontos.

Existem atualmente três principais abordagens do PRCP que utilizam a estrutura do grafo de conflitos: Problema do Máximo Conjunto Independente de Vértices (PMCIV), Problema do Máximo Número de Rótulos Sem Conflitos (PMNRSC) e o Problema da Minimização do Número de Conflitos (PMNC) (Ribeiro e Lorena, 2006).

Como um PMCIV, o objetivo é rotular a maior quantidade de pontos possível, e como problemas reais são em geral muito complexos, alguns pontos podem não ser rotulados com esta abordagem. No PMNRSC, o objetivo é rotular todos os pontos do mapa de forma que a quantidade de rótulos posicionados sem conflitos seja a máxima possível. Já o PMNC visa minimizar a quantidade de conflitos gerados no processo de rotular todos os pontos do mapa.

Mauri et al. (2010) modelaram o PMNRSC como um problema de programação linear inteira binária, e propuseram uma Decomposição Lagrangeana (Chardaire e Sutter, 1995), para dividir o problema original em diversos subproblemas através do particionamento do grafo de conflitos, e resolvê-los de maneira independent. Ao particionar um grafo, é possível que vértices adjacentes fiquem em partições diferentes, gerando arestas entre partições. A quantidade destas arestas é denominada na literatura de *corte* de arestas.

Como as arestas entre vértices do grafo do PMNRSC indicam um conflito entre vértices, elas devem ser consideradas também na decomposição, e para isto um vértice que possua adjacência externa é copiado para a partição do vértice adjacente, de maneira que sua restrição de conflito seja considerada em um dos subproblemas. Para que as características do problema original sejam mantidas, são necessárias restrições adicionais, que assegurem que os valores das variáveis copiadas sejam iguais aos valores das variáveis originais. Estas restrições são então relaxadas no sentido Lagrangeano, permitindo que os subproblemas sejam resolvidos de maneira independente, com um problema dual Lagrangeano correspondente, a ser otimizado utilizando o método de Subgradientes (Narciso e Lorena, 1999).

A Decomposição Lagrangeana apresentou bons resultados e soluções ótimas para instâncias de até 1000 pontos. Porém, o método não conseguiu comprovar a otimalidade de diversas instâncias do conjunto de 1000 pontos, pois para estas instâncias, o Subgradientes teve dificuldade em obter bons limitantes para as soluções.

Este trabalho considera a sensibilidade do método Subgradientes com relação à quantidade de restrições relaxadas, e leva em conta que permitir que o tamanho dos subproblemas sejam inomogêneos, através de um particionamento desbalanceado, possibilita obter uma quantidade menor de restrições relaxadas, facilitando a convergência do método. Os resultados da Decomposição Lagrangeana desbalanceada é comparada com os resultados da decomposição proposta por Mauri et al. (2010).

O restante deste trabalho está organizado com se segue. Uma revisão bibliográfica sobre o PRCP é apresentada na próxima seção. A Seção 3 descreve a Decomposição Lagrangeana para o PMNRSC e o algoritmo para particionamento desbalanceado proposto neste trabalho. A Seção 4 apresenta os resultados computacionais obtidos. Seção 5 apresenta as considerações finais sobre o trabalho.

2. Revisão bibliográfica

Existem diversos trabalhos na literatura sobre as três principais abordagens do PRCP. Considerando como PMCIV, Zoraster (1990) apresentou uma formulação de programação linear binária e propôs restrições com posições candidatas fictícias que são penalizadas na função objetivo. Strijk et al. (2000) propuseram formulações matemáticas utilizando restrições de corte com inequações para todas as cliques máximas no grafo de conflitos. Para instâncias grandes eles aplicaram diversas heurísticas: Simulated Annealing, Diversified Neighborhood Search, *k*-opt e Busca Tabu. Agarwal et al. (1998) propuseram algoritmos de aproximação para rotular a maior quantidade possível de pontos através da obtenção do subconjunto máximo de retângulos de dimensões variadas que não se intersectam. Verweij e Aardal (1999) apresentaram um algoritmo *branch-and-cut* e resultados para instâncias com 950 pontos. Adicionalmente, os autores apresentaram uma técnica baseada em decomposição de caminhos. Ribeiro et al. (2011) apresentaram uma decomposição Lagrangeana para o problema e soluções ótimas para quase todas as instâncias de grande escala propostas na literatura.

Considerando a abordagem do PMNRSC, Christensen et al. (1995) propuseram dois algoritmos baseados em uma discretização da descida de gradiente e Simulated Annealing. Yamamoto et al. (2002) propuseram um algoritmo de Busca Tabu que obteve bons resultados em dados reais. Yamamoto e Lorena (2005) apresentaram um algoritmo exato e um Algoritmo Genético Construtivo. O algoritmo exato não conseguiu bons resultados em instâncias maiores que 25 pontos enquanto o Algoritmo Genético obteve bons resultados para instâncias de até 1000 pontos. Alvim e Taillard (2009) utilizaram as instâncias propostas por Yamamoto et al. (2002) para testes da metaheuristica POPMUSIC. A POPMUSIC constrói inicialmente subproblemas do

problema principal. Estes subproblemas são resolvidos separadamente e são integrados como uma nova solução do problema principal. O método é repetido para obter soluções melhoradas até um dado critério de parada. Mauri et al. (2010) propuseram o primeiro modelo matemático para o PMNRSC e uma decomposição Lagrangeana para resolver o problema em instâncias de até 1000 pontos. Seus resultados computacionais apresentam soluções ótimas para diversas instâncias propostas na literatura.

O PMNC foi introduzido por Ribeiro e Lorena (2006) como uma nova abordagem para melhorar a legibilidade de soluções, quando todos os pontos devem ser rotulados e sobreposições são inevitáveis. Eles apresentaram uma formulação matemática e algumas heurísticas de relaxação Lagrangeanas. Cravo et al. (2008) propuseram um algoritmo GRASP e obtiveram melhores resultados que outras técnicas da literatura. Ribeiro e Lorena (2008) apresentaram duas formulações matemáticas para o PMNC e propuseram relaxações Lagrangeanas com clusters, que obtiveram melhores resultados que os apresentados na literatura. A principal diferença entre as formulações é como os grafos de conflitos são construídos: uma formulação é baseada apenas nas posições candidatas enquanto a outra é baseada em posições candidatas e pontos.

Recentemente Gomes et al. (2013) apresentaram duas formulações de dispersão discreta para o PRCP, utilizando distâncias entre posições candidatas nos grafos de conflitos, que obtiveram bons resultados.

3. Decomposição Lagrangeana para o PMNRSC

O modelo proposto por Mauri et al. (2010) leva em conta a abordagem do PMNRSC, com os rótulos tendo dimensões fixas e utilizando a padronização cartográfica de Christensen et al. (1995). Utiliza-se a variável de decisão binária $x_{i,j}$, para identificar se a posição candidata (i, j) é utilizada para rotular um ponto ou não, a variável binária z_i indicando se um ponto i, está rotulado com conflitos e um peso $w_{i,j}$ indicando a recompensa por escolher a posição candidata (i, j).

Inicialmente o grafo G é particionado em k partições, em que T_m é o subconjunto de vértices de G, representando a partição $m = \{1, ..., k\}$. Em seguida, são selecionados vértices que possuem adjacências externas, para serem copiados para uma das partições de seus vértices adjacentes. Desta forma, são definidos os conjuntos C_m, de vértices copiados para a partição m e $X_m = V - T_m$, o conjunto de todos os vértices que não estão em T_m. Para obter as partições T_m foi utilizada a heurística Metis (Karypis e Kumar, 1998), resolvendo o problema de k-Particionamento, onde k é a quantidade de partições desejada.

Para obter os conjuntos C_m, utiliza-se a estratégia proposta por Sachdeva (2004), que iterativamente seleciona o vértice com a maior quantidade de arestas inter-partição e o copia para o cluster com a maior incidência de arestas inter-partição. Ao fazer isto, estas arestas de conflito entre partições passam a não existir mais, e as restrições de conflito referentes a elas são usadas no subproblema referente à partição onde o vértice foi copiado. Em seguida este número de arestas é redefinido e o procedimento é repetido.

Dados o grafo particionado e os conjuntos C_m e X_m, o PMNRSC é modelado, como apresentado por Mauri et al. (2010):

$$\text{Maximizar} \sum_{m=1}^{k}\left(\sum_{(i,j)\in T_m} w_{i,j}x_{i,j} - \sum_{(i,j)\in T_m} z_i\right) \tag{1}$$

Sujeito a:

$$\sum_{j\in P_i} x_{i,j} = 1 \qquad \forall i = 1,..., N \tag{2}$$

$$x_{i,j} + x_{t,u} - z_i \le 1 \qquad \forall (i,j)\in T_m; \forall (t,u)\in S_{i,j}; i \ne t; m = 1,...,k; \tag{3}$$

$$x_{i,j} + x^m_{t,u} - z_i \le 1 \qquad \forall (i,j)\in T_m; \forall (t,u)\in C_m \cap S_{i,j}; m = 1,...,k; \tag{4}$$

$$x_{i,j} + x^m_{t,u} - z^m_t \le 1 \qquad \forall (i,j)\in T_m; \forall (t,u)\in C_m \cap S_{i,j}; m = 1,...,k; \tag{5}$$

$$x^m_{t,u} = x_{t,u} \qquad \forall (t,u) \in C_m \cap X_m; m = 1,...,k; \qquad (6)$$

$$z^m_t = z_t \qquad \forall (t,u) \in C_m \cap X_m; m = 1,...,k; \qquad (7)$$

$$x_{i,j}, x_{t,u}, x^m_{t,u}, z_i, z_t, z^m_t \in \{0,1\} \qquad \forall (i,j) \in T_m; \forall (t,u) \in T_m \cup C_m; m = 1,...,k; \qquad (8)$$

onde a variável $x^m_{t,u}$ representa a posição candidata (*t, u*) copiada na partição *m* e z^m_t é a variável indicadora da rotulação com conflitos, de um ponto *t* que teve suas posições candidatas copiadas em *m*. A restrição (2) assegura que cada ponto possua apenas um rótulo. A restrição (3) atribui o valor 1 a z_i em caso de conflitos. A restrição (4) considera os conflitos inter-partições, e determina se um ponto externo à partição *m* é rotulado com conflitos. A restrição (5) considera os conflitos inter-partições determinando se um ponto da partição *m* é rotulado com conflitos. As restrições (6) e (7) asseguram a igualdade entre as variáveis de vértices copiados e as variáveis dos vértices originais.

Ao relaxar as restrições (6) e (7) no sentido Lagrangeano, cada partição pode ser modelada em um subproblema v_m como se segue, $\forall\ m = \{1,..., k\}$:

v_m = Maximizar

$$\sum_{(i,j)\in T_m} \left(w_{i,j} - \sum_{d \neq m} \alpha^d_{i,j} \right) x_{i,j} + \sum_{(t,u)\in C_m} \left(\alpha^m_{t,u} \right) x^m_{t,u} - \sum_{(i,j)\in T_m} \left(z_i - \sum_{d \neq m} \beta^d_i \right) z_i + \sum_{(t,u)\in C_m} \left(\beta^m_t \right) z^m_t \qquad (9)$$

Sujeito a:

$$\sum_{j \in P_i} x_{i,j} = 1 \qquad \forall i \in T_m \qquad (10)$$

$$x_{i,j} + x_{t,u} - z_i \leq 1 \qquad \forall (i,j) \in T_m; \forall (t,u) \in S_{i,j}; i \neq t; m = 1,...,k; \qquad (11)$$

$$x_{i,j} + x^m_{t,u} - z_i \leq 1 \qquad \forall (i,j) \in T_m; \forall (t,u) \in C_m \cap S_{i,j}; m = 1,...,k; \qquad (12)$$

$$x_{i,j} + x^m_{t,u} - z^m_t \leq 1 \qquad \forall (i,j) \in T_m; \forall (t,u) \in C_m \cap S_{i,j}; m = 1,...,k; \qquad (13)$$

$$x_{i,j}, x_{t,u}, x^m_{t,u}, z_i, z_t, z^m_t \in \{0,1\} \qquad \forall (i,j) \in T_m; \forall (t,u) \in T_m \cup C_m; m = 1,...,k; \qquad (14)$$

onde α e β são os multiplicadores Lagrangeanos, variáveis do seguinte problema dual a ser otimizado:

Minimizar $\sum_{m=1}^{k} v_m$ (15)

Sujeito a:

$$\alpha, \beta \geq 0 \qquad (16)$$

Como mencionado anteriormente, quanto maior a quantidade de restrições (6) e (7) relaxadas, mais complicado é encontrar bons limitantes duais para o problema decomposto, de forma que obter bons resultados de particionamento podem facilitar a obtenção da solução ótima para uma dada instância do problema.

Considerando um particionamento balanceado, quanto maior o número *k* de partições, maior tende ser o valor do corte, com um pior caso de *k* = *N*, onde o corte é a quantidade de arestas no problema. Por outro lado, um número pequeno de partições, que reduza o valor do corte não necessariamente é uma boa opção, pois o tamanho dos subproblemas pode ser grande demais para ser resolvido por um método exato como o Branch & Bound.

Uma alternativa é permitir que o tamanho das partições seja maior que *L*, de maneira que a inclusão de vértices em uma partição, que reduza a quantidade de arestas entre partições, não seja limitada pelo fator de balanceamento. Isto pode ser usado para obter uma maior

quantidade de subproblemas, e consequentemente subproblemas menores, evitando o acréscimo no valor do corte. O método proposto para obter as partições T_m de maneira desbalanceada é apresentado na seção a seguir.

3. Particionamento desbalanceado

O problema de particionamento geral, conhecido como particionamento em *k*-vias, ou *k*-Particionamento, é o problema de obter para um determinado grafo G= (V, A), k subconjuntos disjuntos de vértices, de maneira que o peso total das arestas entre partições – definido como *corte* – seja minimizado e o número de vértices em cada subconjunto seja o mesmo (ou aproximadamente o mesmo).

Do ponto de vista da otimização combinatória, a minimização do corte é considerada a função objetivo do problema e o balanceamento entre as partições é uma restrição. No entanto existem algumas variações desta função objetivo, que consideram, por exemplo, minimizar especificamente o número de partições interconectadas. Uma variação do problema de particionamento, utilizada neste trabalho é a de permitir o desbalanceamento dos clusters para permitir uma redução da quantidade de arestas entre partições e se possível, o número de partições interconectadas.

Um resultado de particionamento é dito ser balanceado se, $\forall m = \{1,...,k\} : |T_m| \leq L = \lceil |V| / k \rceil$, onde $|T_m|$ é a cardinalidade da partição T_m. Uma solução em que $|T_m| = L$, $\forall\ m = \{1,..., k\}$, é considerada *perfeitamente balanceada*. O algoritmo para particionamento desbalanceado apresentado nesta seção considera um dado fator máximo de balanceamento F_{bal}, de maneira que a inequação $|T_m| / L \leq F_{bal}, \forall m = \{1,...,k\}$ seja a nova restrição de balanceamento utilizada, com $F_{bal} > 1$. O algoritmo utilizado para os testes neste trabalho é apresentado na próxima Seção.

3.1 Algoritmo multinível

Nesta Seção apresenta-se o algoritmo para particionamento desbalanceado, utilizado para obter os subproblemas da Decomposição Lagrangeana. O algoritmo tem como base o trabalho de Hendrickson e Leland (1995), que apresentou uma heurística multinível para o *k*-Particionamento e é atualmente o arcabouço mais utilizado para a resolução de problemas de particionamento em geral.

O algoritmo tem como passos principais o colapsamento de vértices, coalescendo pares de nós adjacentes, gerando um novo grafo. O método é aplicado recursivamente, para formar grupos de vértices, que são utilizados para formar uma partição inicial do problema. Após o particionamento inicial, o método retorna iterativamente ao grafo original, separando os vértices colapsados e aplicando um algoritmo de refinamento da solução de particionamento em cada iteração.

Para o método apresentado nesta seção, são definidos os conjuntos V_l, de vértices de G no nível l, com $V_0 = V$. É interessante, no particionamento para a DecLag do PMNRSC, que todas as posições candidatas (i, j) de um mesmo ponto sejam coalescidas em um mesmo subconjunto C_i no nível um (l = 1), para cada ponto i. Isto porque estes vértices formam cliques, cujas arestas já estarão inclusas neste coalescimento. Então $V_1 = \{C_1, C_2,...C_N\}$ e $C_i = \{(i, 1), (i, 2),...(i, P_i)\}$, $\forall\ i = \{1,...,N\}$.

Além disto, definem se as arestas $a(C_i, C_s) \in A_l$, indicando adjacência entre os subconjuntos C_i e C_s, se existir adjacência entre ao menos um par de vértices (i, j) e (s, t); e os pesos $w_{a(C_i,C_s)}$ associados a estas arestas $\forall$ (i, j) $\in C_i$ e $\forall$ (s, t) $\in C_s$. O algoritmo da etapa de coalescimento é apresentado no pseudocódigo a seguir.

Coalesce Vértices
$V_0 = V$;
Adicionar (i, j) em C_i, $\forall i = \{1,...,N\}$ *e* $j = \{1,..., P_i\}$;
$l = 1$;
$V_l = \{C_1,..., C_N\}$;
enquanto $|V_l| < |V_{l-1}|$ *ou* $|V_l| > k$
    $V_{l+1} = V_l$;
    enquanto *(houver* C_i *não selecionado em* V_{l+1}*)*
        encontrar aleatoriamente C_i *em* V_{l+1}*, que não tenha sido selecionado;*
        marcar C_i *como selecionado;*
        encontrar C_s *em* V_{l+1} *tal que* $|C_s|$ seja o menor, $\forall\ C_s \in V_{l+1}$ *e* $\exists\ a(C_i, C_s) \in A$ *e* C_s *não tenha sido selecionado;*
        se $\exists\ \{C_i, C_s\}$ ***então***
            marcar C_s *como selecionado;*
            $C_i = C_i \cup C_s$;
            $V_{l+1} = V_{l+1} - C_s$;
        fim_se
    fim_enquanto
    $l = l + 1$;
fim_enquanto

Figura 3. Pseudocódigo do algoritmo Coalesce Vértices.

A partir de $l = 1$, dadas as arestas entre cada elemento de V_l, seleciona-se aleatoriamente um elemento C_i. Seleciona-se também o elemento adjacente C_s, que possui a menor cardinalidade. Como inicialmente $|C_s| = P_s$, isto é, cada C_s possui a mesma quantidade de vértices, que são as posições candidatas do ponto s, o primeiro elemento visitado será o escolhido.

Dados os elementos selecionados no passo anterior, para o próximo nível do grafo, todos os vértices de C_s são adicionados a C_i, e C_s é removido. Desta maneira, a cardinalidade de V_{l+1} é reduzida de um, mas a cardinalidade de C_i é incrementada de $|C_s|$. Os elementos C_i e C_s são marcados para que não sejam selecionados novamente nesta iteração e o procedimento é reiniciado. Este processo é repetido enquanto houverem elementos a serem selecionados e ao finalizar, l é incrementado.

Se V_l possuir uma cardinalidade menor que V_{l-1}, o algoritmo fará mais uma iteração dos passos anteriores na tentativa de reduzir o grafo novamente. Caso contrário, o algoritmo finaliza. Outro critério de parada é a cardinalidade de V_l ser igual à quantidade k de partições. A Figura 4 ilustra o funcionamento da etapa de coalescimento dos vértices com $k = 3$.

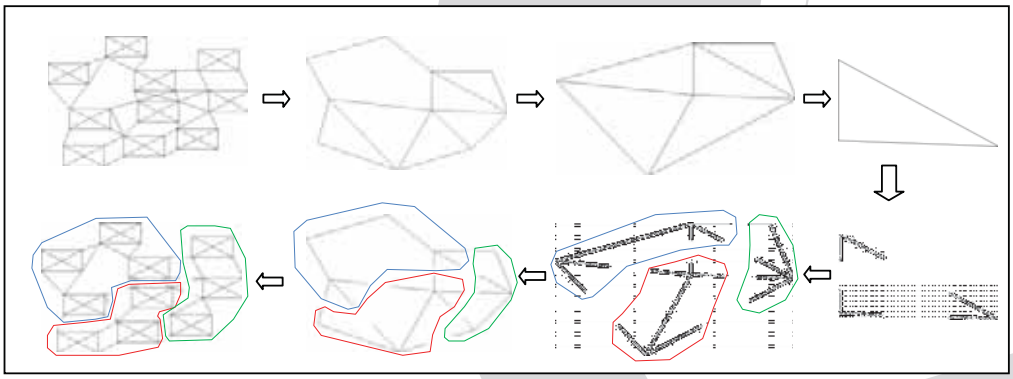

Figura 4. Exemplo da etapa de coalescimento de vértices com $k = 3$.

Como resultado da execução do algoritmo Coalesce Vértices, o grafo estará reduzido a um conjunto menor de vértices, facilitando assim, determinar uma partição inicial de *G,* obtida pelo algoritmo Obtém Partição Inicial, apresentado na Figura 5 a seguir.

Obtém Partição Inicial

repetir *para todo* $C_i \in V_l$
  encontrar C_i *que possui o menor* $|C_i|$, $\forall\ C_i \in V_l$;
  encontrar a partição T_m *tal que* $T_m = T_m \cup C_i$ *produza a menor violação de* F_{bal};
  se $|T_m \cup C_i|/L \leq F_{bal}$ ***então***
    $T_m = T_m \cup C_i$;
  fim_se
  senão
    executar ***Partição Inicial*** *em* V_{l-1};
  fim_senão
fim_repetir

Figura 5. Pseudocódigo do algoritmo obtém partição inicial.

No algoritmo da Figura 5, l é iniciado do último incremento de Coalesce Vértices, ou seja, inicia-se do último nível de coalescimento obtido para G, e sendo assim, V_l possui poucos subconjuntos C_i, que por sua vez contêm grandes quantidades de vértices do grafo original. Nesta etapa, o objetivo é obter uma partição inicial para G, inserindo os elementos de V_l nas partições T_m, $\forall\ m \in \{1,..., k\}$, levando em conta o fator de balanceamento F_{bal} desejado. Se eventualmente não é possível selecionar um elemento C_i que não viole F_{bal}, o algoritmo é aplicado recursivamente em V_{l-1}.

Os elementos de C_i, coalescidos em V_l, estão separados em V_{l-1}, e consequentemente possuem uma cardinalidade menor. O algoritmo efetua a busca novamente por um conjunto C_i que não viole F_{bal} e se encontrado, ele é inserido na partição T_m tal que $|T_m \cup C_i|/L \leq F_{bal}$, $\forall\ m \in \{1, ..., k\}$.

Apesar de o algoritmo Obtém Partição Inicial determinar uma partição para G, ele não considera o problema de minimizar a quantidade de arestas entre as partições. Para isto, neste trabalho, é utilizado um algoritmo baseado na técnica proposta por Kernighan e Lin (1970). Esta técnica utiliza o conceito de "matriz de ganho", que contém para cada vértice do grafo, valores indicando a contribuição para a minimização do corte ao "mover" este vértice da partição atual para outras partições. Para isto, utiliza-se o valor de ganho $g^m(C_i)$ ao mover o elemento C_i de sua partição atual p para a partição m. Dados os pesos $w_{a(C_i,C_s)}$ para cada aresta $a(C_i, C_s) \in A_l$, onde A_l é o conjunto de arestas de G no nível l. A matriz de ganho, baseada no modelo Kernighan – Lin pode ser definido como

$$g^m(C_i) = \sum_{a(C_i,C_s)\in A_l} \begin{cases} w_{a(C_i,C_s)} \ se\ \ C_s \in T_m \\ -w_{a(C_i,C_s)} \ se\ \ C_s \in T_p \\ 0 \quad se\ \ C_s \in T_r, r \neq p, r \neq m \end{cases} \qquad (17)$$

No somatório em (17), o ganho em mover C_i para T_m aumenta para cada elemento adjacente C_j que estiver na nova partição. Se C_s estiver na partição atual T_p, uma penalidade é imposta no valor de $g^m(C_i)$, e se C_s não pertencer nem a T_m e nem a T_p, não existe alteração no ganho. O valor de $g^m(C_i)$ é utilizado no pseudocódigo a seguir.

Refina Particionamento

enquanto *(não alcança a condição de parada)*
calcular $g^m(C_i)$ *para todo* $C_i \in V_0$ *que não tenha sido movido,*$\forall\ m = \{1, ..., k\}$;
obter o maior valor de $g^m(C_i)$, tal que $|T_m \cup C_i|/L \leq F_{bal}$;
dado $g^m(C_i)$, *mover o conjunto* C_i *para a partição m;*
marcar i como "movido";
fim_enquanto

Figura 6. Pseudocódigo do algoritmo Refina Particionamento.

No refinamento da solução, o algoritmo busca pelos melhores movimentos de vértices utilizando os maiores valores de ganho $g^m(C_i)$. O procedimento reitera enquanto houverem vértices não selecionados que tenham um movimento possível que melhore o resultado do corte, e não viole F_{bal}.

4. Resultados computacionais

Nesta seção são apresentados resultados computacionais da decomposição desbalanceada do PMNRSC para as instâncias 1000 pontos, que foram utilizadas para os testes de Mauri et al. (2010) e não tiveram as soluções ótimas comprovadas, considerando quatro posições candidatas para cada ponto, de acordo com a padronização de Christensen et al. (1995). Estas instâncias, geradas no trabalho de Yamamoto et al. (2002), estão disponíveis no site http://www.lac.inpe.br/_lorena/instancias.html.

Primeiro, são apresentados na Tabela 1, os resultados de testes com as mesmas quantidades de partições, utilizadas no trabalho de Mauri et al. (2010). A quantidade de partições é apresentada na coluna com o cabeçalho *k*, seguido das melhores soluções obtidas na coluna LI, e dos limitantes duais na coluna LS. Em seguida é apresentado o *Gap* relativo entre LI e LS, e os tempos de execução do método. Os testes foram executados utilizando o solver comercial CPLEX 12.6, utilizando o fator $F_{bal} = 2$. Ou seja, $\forall m = \{1,...,k\} : |T_m|/L \leq 2$.

Como esperado, a Tabela 1 mostra melhores resultados para o limite dual obtido pela versão desbalanceada da Decomposição Lagrangeana, possibilitando provar a otimalidade de 13 das 19 instâncias teste. Além disto, foram obtidos novos resultados para a função objetivo do problema em quatro instâncias. Foi possível também obter melhores resultados de tempos de execução do método para 12 instâncias. Em todas as instâncias, o *Gap* da decomposição desbalanceada foi melhor.

Além dos testes com as quantidades de partições utilizadas por Mauri et al. (2010), foram executados testes com outros valores de *k*. Para a escolha destas partições, foi adotado o critério do menor valor de corte gerado no particionamento.

Foi observado que para outros valores de F_{bal}, é possível obter quantidades de variáveis copiadas e restrições relaxadas similares ao valor do corte, implicando uma menor quantidade de arestas inter-partições incidentes nos vértices a serem copiados. Isto por sua vez significa que uma dada variável copiada é utilizada em poucas restrições de conflitos dos tipos (12) e (13), por precisarem eliminar poucas arestas inter-partições no processo de cópia. Como variáveis de restrições relaxadas participam de menos restrições dos subproblemas, o Subgradientes apresentou uma melhora no tempo de convergência e nos limites obtidos.

Para analisar a relação entre a quantidade de variáveis relaxadas e o valor do corte, é utilizada a proporção H = *nº de variáveis copiadas/corte*. Os dados com informações sobre os valores de *k*, quantidade de corte, restrições relaxadas e os valores de F_{bal} utilizados para obter a proporção H do particionamento desbalanceado, para todas as instâncias são apesentados na Tabela 2. Os resultados obtidos com estes particionamentos são apresentados na Tabela 3.

Tabela 1. Resultados utilizando o particionamento de Mauri et al. (2010).

		Desbalanceado				Balanceado			
Instância	***k***	**LI**	**LS**	***Gap***	**Tempo de execução (*s*)**	**LI**	**LS**	**Gap**	**Tempo de execução (*s*)**
1000_2	25	934	935.5	**0.16**	47189.06	934	962.59	2.97	**9372.64**
1000_4	40	933	935.52	**0.26**	14703.38	933	962.43	3.05	**9234.03**
1000_5	15	961	962	**0.1**	12194.26	961	976.20	1.55	**8834.52**
1000_7	25	**929**	929.94	**0.1**	**1927.41**	928	931.99	0.42	14419.30
1000_8	20	940	940	**0**	**1561.10**	940	942.21	0.23	14419.89
1000_9	20	**927**	927.92	**0.09**	**4728.74**	925	959.52	3.59	8894.97
1000_10	20	944	944.89	**0.09**	**5483.18**	944	971.54	2.83	8861.16
1000_12	25	**935**	935.99	**0.1**	**1443.80**	934	936.44	0.26	14562.00
1000_13	25	954	955.20	**0.12**	18582.80	**955**	972.99	1.84	**8964.73**
1000_14	25	933	934.17	**0.12**	19250.06	933	960.52	2.86	**9293.41**
1000_15	25	934	934.31	**0.03**	75274.17	934	963.99	3.11	**9248.70**
1000_16	25	**932**	932.97	**0.1**	**3030.37**	931	959.58	2.97	9647.72
1000_17	25	937	937	**0**	**1269.61**	937	964.61	2.86	11243.55
1000_18	25	946	946.97	**0.1**	**1132.63**	946	947.27	0.13	14509.09
1000_19	25	950	950.98	**0.1**	**2969.85**	950	953.25	0.34	15264.32
1000_21	25	930	930	**0**	**177.14**	930	955.38	2.65	8707.73
1000_22	25	952	952.99	**0.1**	**2377.17**	952	955.41	0.35	14406.28
1000_23	25	934	934.97	**0.1**	**3454.19**	934	960.13	2.72	9103.09
1000_24	25	931	933.01	**0.21**	12721.37	**932**	964.74	3.39	**9178.91**

Tabela 2. Dados dos testes utilizando diferentes valores de k e F_{bal}.

		Desbalanceado				Balanceado		
Instância	***k***	F_{bal}	**Corte**	**Variáveis Copiadas**	***H***	**Corte**	**Variáveis Copiadas**	***H***
1000_2	20	2	102	99	0.97	270	168	0.7
1000_4	26	1.7	161	142	0.88	448	246	0.54
1000_5	13	2	75	73	0.97	132	109	0.82
1000_7	20	1.7	135	105	0.77	176	159	0.9
1000_8	20	2	97	98	1.01	282	171	0.6
1000_9	20	2	123	127	1.03	201	162	0.8
1000_10	20	2	110	105	0.95	244	166	0.68
1000_12	16	3	65	73	1.12	195	148	0.7
1000_13	14	2	88	89	1.01	162	140	0.86
1000_14	10	3	44	43	0.97	89	77	0.86
1000_15	26	1.7	164	141	0.85	251	188	0.74
1000_16	25	2	194	181	0.93	312	234	0.75
1000_17	30	1.7	164	150	0.91	802	376	0.46
1000_18	25	2	170	154	0.9	277	212	0.76
1000_19	18	2	166	136	0.81	256	182	0.71
1000_21	24	2	131	119	0.9	377	234	0.62
1000_22	28	2	224	186	0.83	410	262	0.63
1000_23	26	2	176	167	0.94	397	273	0.68
1000_24	12	3	60	56	0.93	350	165	0.47

Tabela 3. Resultados com solução ótima obtidos com os particionamentos da Tabela 2.

	Desbalanceado					Balanceado			
Instância	***k***	**LI**	**LS**	***Gap***	**Tempo de execução**	**LI**	**LS**	***Gap***	**Tempo de execução**
1000_2	20	934	934	0	**462.78**	934	936.89	0.3	11750.23
1000_4	26	**934**	934.79	0.08	**805.10**	934	934.96	0.1	4260.88
1000_5	13	961	961.99	0.1	**1220.65**	961	961.22	0.02	6880.11
1000_7	20	929	929.98	0.1	**663.49**	929	929.97	0.1	3638.07
1000_8	20	940	940	0	**1561.10**	940	942.21	0.2	14419.89
1000_9	20	927	927.92	0.09	**4728.74**	925	959.52	3.73	8894.97
1000_10	20	944	944.89	0.09	**5483.18**	944	971.54	2.91	8861.16
1000_12	16	935	935	0	**196.13**	935	935.57	0.06	2660.74
1000_13	14	**955**	955	0	**136.88**	955	955.03	0.003	4623.14
1000_14	10	933	933	0	**4104.70**	933	934.87	0.2	10395.89
1000_15	26	934	934.98	0.1	**1246.76**	933	934.03	0.11	12216.92
1000_16	25	932	932.97	0.1	**3030.37**	931	959.58	3.06	9647.72
1000_17	30	937	937	0	**169.36**	937	940.45	0.36	14566.11
1000_18	25	946	946.97	0.1	**1132.63**	946	947.27	0.13	14509.09
1000_19	18	950	950	0	**244.11**	940	950.49	1.11	20842.81
1000_21	24	930	930	0	**174.97**	930	931.96	0.21	6595.36
1000_22	28	952	952.98	0.1	**5729.44**	952	953.57	0.16	18134.88
1000_23	26	934	934	0	**264.17**	934	935	0.1	8965.35
1000_24	12	**932**	932	0	**15172.38**	931	938.28	0.78	21354.97

Os valores de *k* da Tabela 3 apresentaram melhores limitantes para o problema. A otimalidade dos resultados foi comprovada para todas as instâncias e a versão desbalanceada da decomposição Lagrangeana obteve melhores resultados de tempo computacional. Além disto, foi possível obter novos valores para a função objetivo de três instâncias. Com os resultados apresentados, todas as instâncias geradas no trabalho de Yamamoto et al. (2002), e utilizadas por Mauri et al. (2010) tiveram suas soluções ótimas obtidas. Estes resultados obtidos pela decomposição desbalanceada são possíveis principalmente por possibilitar obter menor quantidade de arestas entre partições e consequentemente, melhores limitantes e melhor valor do *gap*.

5. Conclusão

Este trabalho apresentou uma abordagem de Decomposição Lagrangeana desbalanceada para o Problema de Rotulação Cartográfica de Pontos. A abordagem foi comparada com a Decomposição Lagrangeana proposta por Mauri et al. (2010), que trata-se uma formulação matemática para o Problema do Maior Número de Rótulos sem conflitos, decomposta homogeneamente no sentido Lagrangeano, que apresentou bons resultados na literatura.

Para a abordagem inomogênea, foi implementado um algoritmo de particionamento multinível, que permite o desbalanceamento de partições, para que o corte de arestas inter-partição seja menor, com o objetivo de obter menos restrições relaxadas na Decomposição Lagrangeana.

Através de testes computacionais, foi observado que a versão desbalanceada obteve melhores resultados de limitante, permitindo encontrar a solução ótima de todas as instâncias de testes, indicando que o desbalanceamento de partições melhora a convergência do método de Subgradientes, por reduzir a quantidade de restrições relaxadas, mesmo aumentando o tamanho de alguns dos subproblemas. Observa-se porém que obter particionamento ótimo, não é o escopo deste trabalho, e portanto, é possível que resultados melhores de corte destas instâncias sejam obtidos.

Trabalhos futuros incluem a implementação de métodos para determinar

automaticamente bons valores de k e de F_{bal}, que permitam obter bons resultados de corte de arestas interpartições e da proporção H.

Referências

Agarwal, P. K., Kreveld, M. V., Suri, S., (1998). Label placement by maximum independent set in rectangles. Computational Geometry: Theory and Applications 11, 209 - 218.

Alvim, A. C. F., Taillard, E. D., (2009). POPMUSIC for the point feature label placement problem. European Journal of Operational Research 192(2), 396–413.

Chardaire, P., Sutter, A., (1995) A decomposition method for quadratic zero-one programming. Management Science, v. 41, n. 4, p. 704-712.

Christensen, J., Marks, J., Shieber, S., (1995). An empirical study of algorithms for point-features label placement. ACM Transactions on Graphics 14(3), 203–232.

Gomes, S. P., Ribeiro, G. M., Lorena, L. A. N., (2013). Dispersion for the point feature cartographic label placement problem. Expert Systems with Applications, 5878-5883.

Hendrickson, B., Leland, R. A., (1995). Multilevel algorithm for partitioning graphs**.** Supercomputing '95. New York: ACM Press.

Karypis, G., Kumar, V., (1998). Multilevel k-way partitioning scheme for irregular graphs. Journal of Parallel and Distributed Computing, v. 48, n. 1, p. 96-129.

Kernighan, B. W., Lin, S., (1970) An Efficient Heuristic for Partitioning Graphs. Bell Systems Technical Journal, v. 49, p. 291-308.

Mauri, G. R., Ribeiro, G. M., Lorena, L. A. N., (2010). A new mathematical model and a lagrangean decomposition for the point-feature cartographic label placement problem. Computers & Operations Research 37(12), 2164-2172.

Narciso, M. G., Lorena LAN., (1999). Lagrangean/surrogate relaxation for generalized assignment problems. European Journal of Operational Research 114(1), 165–77.

Ribeiro, G. M., Lorena, L. A. N., (2006). Heuristics for cartographic label placement problems. Computers & Geosciences 32(6), 739-748.

Ribeiro, G. M., Lorena, L. A. N., (2008). Lagrangean relaxation with clusters for point-feature cartographic label placement problems. Computers & Operations Research 35(7), p. 2129-2140.

Ribeiro, G. M., Mauri, G. R., Lorena, L. A. N., (2011). A lagrangean decomposition for the maximum independent set problem applied to map labeling. Operational Research 11(3), 229-243.

Sachdeva, S., (2004). Development of a branch and price approach involving vertex cloning to solve the maximum weghted independent set problem**.** A&M University. [S.l.].

Strijk, T., Verweij, B., Aardal, K., (2000). Algorithms for maximum independent set applied to map labeling, 42pp. Available at www.cs.uu.nl/research/techreps/repo/CS-2000/2000-22.ps.gz, [Accessed June 20, 2012].

Verweij, A. M., Aardal, K. I., (1999). An optimization algorithm for maximum independent set with aplications in map labelling. In: Proceedings 7th Annual European Symposium on Algorithms, Prague, Czech Republic, pp. 426-437.

Yamamoto, M., Camara, G., Lorena, L. A. N., (2002). Tabu search heuristic for point-feature cartographic label placement. Geoinformatica 6(1), 77–90.

Yamamoto, M.; Lorena, L. A. N., (2005). A constructive genetic approach to point-feature cartographic label placement. In: Ibaraki, T., Nonobe, K., Yagiura, M. (Eds.), Metaheuristics: Progress as Real Problem Solvers, Kluwer Academic Publishers, pp. 285–300.

Zoraster, S., (1990). The solution of large 0-1 integer programming problems encountered in automated cartography. Operations Research 38(5), 752–759.