Novas Abordagens Exatas para o Problema de Partição de um Grafo em Árvores k-Capacitadas

Tibérius O. Bonates[§] Cláudio P. Santiago[¶] Jorge B. C. Silva[‖]

Resumo

Dados um grafo não-direcionado G, um inteiro positivo k e uma função de custo, o Problema de Partição em Árvores k-Capacitadas (PAkC) consiste em encontrar uma partição, de custo mínimo, de G em árvores contendo k ou mais vértices. Fazemos uma breve revisão da heurística HEF para o PAkC, que constrói uma solução baseada em uma Árvore Geradora Mínima (AGM) e descrevemos um algoritmo *branch-and-bound* para o PAkC, que nos permite calcular a melhor solução que é um subconjunto de uma AGM. Apresentamos, também, um modelo de programação linear inteira mista para o PAkC. Mostramos como utilizar o algoritmo *branch-and-bound* e o modelo para: (i) avaliar a qualidade da heurística HEF com relação à melhor solução baseada em AGM; e (ii) mostrar que, para grafos densos e pequenos, o custo desta melhor solução baseada em AGM é próximo do custo ótimo. Por fim, discutimos resultados computacionais e possíveis extensões do estudo.

Abstract

Given an undirected graph $G = (V, E)$, a positive integer k and a cost function, the k-Capacitated Tree Partition Problem (PAkC) asks for a minimum-cost partition of G into components, each of which is a tree spanning at least k vertices. We briefly review the HEF heuristic, which computes a solution based on a minimum spanning tree (MST). Next, we present a branch-and-bound algorithm for the PAkC, which computes the best solution that is a subset of a MST. We also present a mixed integer linear programming model for the PAkC. Then, we show how to use the branch-and-bound algorithm and the model to: (i) assess the quality of the HEF heuristic as compared to the best MST-based solution; and (ii) show that for small, dense graphs the cost of the best MST-based solution is close to the optimum. Finally, we present computational results and discuss extensions of our work.

1 Introdução

Sejam $G = (V, E)$ um grafo simples não-orientado, $c : E \to \mathbb{R}$ uma função real sobre as arestas de G, e k um inteiro positivo menor ou igual a $|V|$. O problema de Particionamento de G em Árvores k-Capacitadas (PAkC) (ver Crescenzi (1998)) consiste em encontrar um subconjunto $S \subseteq E$ de arestas, com as seguintes propriedades:

(i) o subgrafo de G induzido por S seja uma coleção $\mathcal{T}$ de árvores;

[§]Universidade Federal do Semiárido (UFERSA), Mossoró, RN, Brasil (tbonates@ufersa.edu.br).
[¶]OM Partners USA, Atlanta, GA, USA (pratas@gmail.com).
[‖]Serviço Federal de Processamento de Dados / Faculdade Integrada do Ceará, Fortaleza, CE, Brasil (jorgebergson@gmail.com).

(ii) cada árvore de $\mathcal{T}$ contenha pelo menos k vértices de V;

(iii) cada vértice de V pertença a uma única árvore em $\mathcal{T}$; e

(iv) o conjunto de arestas S tenha custo total mínimo.

Seja $F \subseteq E$ um subconjunto de arestas de G. Iremos denotar por $c(F)$ o custo total das arestas em F, ou seja $c(F) = \sum_{e \in F} c(e)$. Para um inteiro positivo $k \leq n$, uma solução para uma instância do PAkC consiste em uma coleção $\mathcal{T}$ de árvores disjuntas com relação a vértices, cada uma das quais contém, no mínimo, k vértices. O conjunto de árvores definido pelo conjunto F é chamado de uma partição de G, já que cada vértice de G pertence a exatamente uma árvore de $\mathcal{T}$. A restrição que requer um mínimo de k vértices em cada árvore é chamada de *restrição de capacidade*. Dentre os resultados teóricos acerca do PAkC, é interessante mencionar que Imielińska (1993) mostrou que o PAkC é NP-Árduo, para $k \geq 4$, e que Goemans (1995) demonstrou que o PAkC pode ser aproximado com um fator de $2 - \frac{1}{|V|}$.

O PAkC tem sido descrito na literatura sob alguns nomes diferentes. Crescenzi (1998) documenta o problema sob o nome "Minimum k-Capacitated Tree Partition Problem", enquanto Imielińska (1993), Goemans (1995) e Laszlo (2005a) tratam do mesmo problema sob a denominação genérica de "Constrained Forest Problem". Mais recentemente, o problema foi estudado por Ji (2004) sob o nome de "Minimum Weight Constrained Forest Problem". Laszlo (2005b) apresenta uma aplicação do problema no contexto de micro-agregação, uma técnica de divulgação limitada de dados, na qual registros similares são agregados em grupos de k ou mais registros, anteriormente à divulgação dos dados.

No restante do artigo iremos assumir que o grafo $G = (V, E)$ é um grafo não-orientado simples (isto é, sem arestas ligando um nó a si mesmo e sem arestas paralelas), onde $V = \{v_1, \ldots, v_n\}$ é o conjunto de vértices do grafo e $E \subseteq \{\{v_i, v_j\} : v_i, v_j \in V,\ v_i \neq v_j\}$ é seu conjunto de arestas, com $m = |E|$. Utilizaremos a notação $V(H)$ para nos referirmos ao conjunto de vértices de um subgrafo H de G. Vértices de G serão representados por letras minúsculas, tais como u, v, and w (possivelmente com índices), enquanto que arestas serão tipicamente representadas pelas letras e, f, and g, ou por pares não-ordenados de vértices $\{u, v\}$, e assim por diante.

A estrutura deste artigo está organizada da seguinte maneira. A Seção 2 descreve a heurística HEF, que é atualmente a heurística mais bem-sucedida, em termos de tempo de execução e qualidade de solução, para o PAkC. Na Seção 3, descrevemos um novo algoritmo de *branch-and-bound*, que nos permite partir de uma AGM $\mathcal{T}$ do grafo G e encontrar a melhor solução que pode ser obtida por meio da remoção de arestas de $\mathcal{T}$. A Seção 4 descreve um modelo de programação linear inteira inédito para o PAkC, além de apresentar duas famílias de restrições que podem ser utilizadas para reduzir ou eliminar a simetria naturalmente existente no modelo proposto. Uma dessas famílias de restrições pode ser manipulada para reduzir o espaço de soluções de maneira drástica, permitindo apenas soluções com determinado grau máximo. A Seção 5 apresenta resultados computacionais e descreve o primeiro conjunto de instâncias publicamente acessível. Por fim, na Seção 6 discutimos as contribuições do artigo e listamos possíveis trabalhos futuros.

2 Heurística *Heaviest Edge First*

Neste seção, descrevemos brevemente a heurística *Heaviest Edge First* (HEF), originalmente proposta por Laszlo (2005a), e discutida por Laszlo (2006). A heurística consiste em uma

estratégia gulosa, que, partindo de uma Árvore Geradora Mínima (AGM) $\mathcal{T}$ do grafo G (ver Tarjan (1983)), remove sucessivamente as arestas mais caras (ou mais *pesadas*) de $\mathcal{T}$ com o devido cuidado para evitar que, durante este processo, surjam componentes com menos de k vértices.

Na descrição da heurística, o cálculo de uma AGM é considerado como uma etapa de pre-processamento do algoritmo e pode ser implementado por meio de um dos algoritmos clássicos para o problema, como os algoritmos de Borůvka (1926), Kruskal (1956) e Prim (1957), ou, ainda, o algoritmo recente de Chazelle (2000).

Seja $\mathcal{T}$ uma árvore geradora mínima do grafo G. O algoritmo HEF prossegue de maneira gulosa, removendo as arestas mais caras de $\mathcal{T}$, desde que isso não acarrete a criação de componentes com menos de k vértices. Para tanto, é necessário, primeiramente, ordenar as arestas de $\mathcal{T}$. Seja S a lista ordenada da arestas de $\mathcal{T}$. As arestas de S são inspecionadas seqüencialmente, em ordem decrescente de custo. Para cada aresta, verifica-se se sua remoção de S resulta na existência de uma componente conexa com um número de vértices menor do que k. Se este for o caso, a aresta é mantida em S. Caso contrário, a aresta é removida.

É fácil ver que, ao final deste procedimento, obtém-se um conjunto de arestas que não somente constitui uma solução viável para o PAkC, mas também é uma solução minimal, não admitindo a remoção de nenhuma aresta adicional.

3 Um Algoritmo *Branch-and-Bound*

Neste seção, propomos um algoritmo do tipo *branch-and-bound*, que baseia-se no fato de que, dependendo da estrutura do grafo G, pode ser possível garantir que certas arestas farão, necessariamente, parte de qualquer solução viável do PAkC. Em particular, para $k \geq 2$, se o grafo contiver algum vértice v que possua exatamente um vizinho, então esta única aresta adjacente a v deverá fazer parte de qualquer solução viável do problema; caso contrário, v não faria parte de uma componente de tamanho maior ou igual a k.

Assuma que o grafo foi preprocessado da maneira descrita acima, e que uma solução parcial foi construída, consistindo apenas daquelas arestas adjacentes a vértices que possuiam um único vizinho em G. Note que, neste cenário, algumas componentes com 2 ou mais vértices começaram a ser construídas e irão, na medida em que o algoritmo progredir, constituir partes de soluções completas para o PAkC.

Diremos que um dado vértice v é um *vértice de ligação* se, e somente se: (i) v pertence a uma componente de tamanho estritamente menor que k; e (ii) dentre todas as arestas adjacentes a v, existe exatamente uma aresta que não pertence à solução parcial atual. A Figura 3 ilustra este conceito.

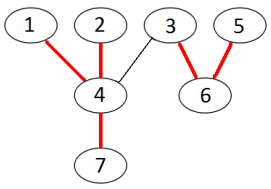

Figura 1: Exemplo de um vértice de ligação: nó 3 é um vértice de ligação, para $k = 4$.

Seja v um vértice de ligação em um grafo G que foi submetido ao passo de preprocessamento descrito acima, e seja e a única aresta adjacente a v que não pertence à solucao parcial atual.

Como e é a única ligação entre a componente inviável à qual v pertence na solução parcial em questão e o resto do grafo G, então podemos afirmar que e deve, necessariamente, pertencer a qualquer solução viável do problema. O mesmo raciocínio pode ser aplicado sucessivamente, enquanto houver alguma componente inviável com uma única aresta conectando-a ao restante do grafo. Após sucessivas aplicações deste raciocínio, é possível que nos deparemos com uma solução completa para o problema, ou com uma solução parcial que não contém vértices de ligação. Para ilustrar este último cenário, considere o caso de um caminho contendo 7 vértices (isto é, um P_7), com $k = 3$. Após a aplicação do preprocessamento e do raciocínio envolvendo vértices de ligação descrito acima, obtemos duas componentes (cada uma com tamanho 3), um único vértice que não pertence a nenhuma componente (ou pertence a uma componente de tamanho 1), e nenhum vértice de ligação.

A partir de tal situação, o algoritmo escolhe uma aresta e para realizar a operação de ramificação, criando dois novos nós da árvore de *branch-and-bound*:

- um nó que incluirá a aresta de ramificação em sua solução parcial. Nesta situação, tenta-se novamente aplicar o raciocínio de identificar vértices de ligação e incorporar as arestas associadas a eles na solução parcial atual;
- um nó onde tal aresta será descartada, o que significa que e não será considerada para inclusão em nenhuma solução futuramente gerada a partir daquele nó da árvore. A partir deste ponto, pode ser possível aplicar tanto o passo de preprocessamento que identifica vértices que possuem um único vizinho, quanto o raciocínio de identificar vértices de ligação.

Na implementação discutida neste artigo, o algoritmo seleciona preferencialmente, como arestas de ramificação, aquelas que são adjacentes a vértices que possuem apenas uma aresta que ainda não pertence à solução e que não foi ainda descartada durante a busca de *branch-and-bound*.

A seguir, discutimos um processo de limitação, que permite a poda prematura da árvore de *branch-and-bound* e que constitui um aspecto fundamental para o bom desempenho do algoritmo.

3.1 Limitação

A seguir, mostramos os detalhes do procedimento de limitação (*bounding*) do algoritmo de *branch-and-bound* aqui proposto. A idéia principal consiste em obter uma estimativa otimista do valor da função objetivo de uma solução viável construída a partir da solução parcial correspondente a um dado nó da árvore de *branch-and-bound*. O seguinte lema estabelece um limite inferior para o número de arestas necessárias para se obter uma solução viável a partir da solução parcial existente em um determinado nó da árvore de *branch-and-bound*.

Lema 3.1 *Sejam $G = (V, A)$ um grafo, com $|V| = n \geq k$ vértices, e S uma solução parcial para uma instância do PAkC sobre G. Seja $I = \{C_1, C_2, \ldots, C_t\}$ o conjunto das t componentes inviáveis de S, isto é, as componentes de S que possuem menos de k vértices. A seguinte expressão fornece um limite inferior para o número mínimo de arestas necessárias para se construir uma solução viável S^* a partir de S:*

$$N(I) = |I| - P(I), \tag{1}$$

$$onde\ P(I) = \left\lceil \frac{\sum_{i=1}^{t} |C_i|}{k} \right\rceil.$$

Prova Iremos assumir que o grafo G é completo. Tal hipotese é condizente com o fato de estarmos lidando com um limite inferior para o número de arestas necessárias para se construir uma solução viável a partir de S. Caso G não seja completo, é possível que este limite seja válido de maneira estrita.

Considere o caso em que $P(I) = 0$. O menor número de arestas necessárias para se criar uma solução viável a partir de S é t, satisfazendo (1) na igualdade.

Assumindo que (1) vale para $|I| \leq t - 1$, iremos mostrar que a expressao tambem é válida para $|I| = t$. Considere $F \subseteq I$ um subconjunto minimal (com relação a inclusão) de componentes inviáves de S, satisfazendo $\sum_{C_j \in F} |C_j| \geq k$. São necessárias pelo menos $|F| - 1$ arestas para se transformar as componentes em F em uma ou mais componentes viáveis. Assumindo que as componentes em F serão ligadas com este número mínimo de arestas, podemos escrever

$$\begin{aligned} N(I) &= |I| - P(I) \\ &= |I \setminus F| + |F| - (P(I \setminus F) + P(F)) \\ &= |I \setminus F| + |F| - P(I \setminus F) - 1 \\ &= (|F| - 1) + |I \setminus F| - P(I \setminus F) \\ &= (|F| - 1) + N(I \setminus F), \end{aligned}$$

completando a prova. ■

Para um dado nó da árvore de *branch-and-bound*, o valor obtido no cálculo de (1) pode ser utilizado para se calcular um limite inferior para o custo da função objetivo da melhor solução completa que se pode obter a partir da solução parcial associada àquele nó. Se ordenarmos todas as arestas do grafo G antes de iniciar o algoritmo, podemos calcular a soma dos custos das $N(i)$ arestas mais baratas de G, obtendo uma estimativa otimista do custo adicional para se construir uma solução viável a partir de S. É este o método de estimativa utilizado para limitação (poda) da árvore de *branch-and-bound* nos experimentos reportados aqui.

3.2 Aplicação a Subgrafos Esparsos

O algoritmo *branch-and-bound* descrito nesta seção pode ser aplicado a grafos gerais. No entanto, seu sucesso depende fortemente de sua capacidade de fixar múltiplas variáveis, em decorrência da aplicação das idéias de vértices com um único vizinho e de vértices de ligação. Em um grafo denso, tais oportunidades costumam ser reduzidas, na medida em que cada vértice tende a possuir vários vizinhos. Assim, a idéia de se aplicar o algoritmo a subgrafos esparsos de G surge como alternativa natural de obtenção de uma solução viável para o problema original.

Como o algoritmo *branch-and-bound* aqui descrito é capaz de encontrar a solução ótima para o grafo dado, a escolha cuidadosa de um subgrafo esparso de G pode representar uma alternativa interessante para a construção de uma solução de boa qualidade para o PAkC em um grafo geral. De fato, esta é a abordagem que adotamos em nossos testes computacionais descritos na Seção 5, nos quais o subgrafo esparso escolhido consiste de uma AGM de G.

4 Modelo de Programação Matemática

Nesta seção, apresentamos uma formulação em programação matemática para o problema PAkC, que consiste em um modelo de fluxo em redes sobre um grafo direcionado. Considere o grafo direcionado $D = (V, A)$, onde A é o conjunto de arcos contendo todos os arcos (v_i, v_j) e (v_j, v_i), para todas as arestas $\{v_i, v_j\} \in E$. Seja v_0 um vértice artificial, e considere os arcos artificiais (v_0, v_i), para todos $i = 1, \ldots, n$. Mostraremos abaixo como formular o PAkC como o problema de enviar n unidades de fluxo a partir do vértice v_0 até os demais vértices da rede D utilizando apenas um subconjunto de arcos corresponde a um subgrafo de custo mínimo do grafo G.

Uma importante suposição que faremos a respeito dos custos é de que todos os ciclos possuem custo positivo. Conforme veremos, esta suposição é crucial para que possamos estabelecer a viabilidade das soluções do modelo aqui apresentado.

A solução do seguinte modelo de programação inteira mista fornece um projeto de rede que induz uma solução para o problema PAkC. A solução do modelo seleciona um subconjunto das arestas do grafo original D, determinando uma partição daquele grafo. As restrições de fluxo ao longo da rede D asseguram que as restrições de capacidade de cada uma das componentes conectadas resultantes são satisfeitas.

$$
\begin{aligned}
\text{(ND)} \quad \text{minimizar} \quad & \sum_{e \in E} c(e) y_e & \\
\text{sujeito a:} \quad & \sum_{j \in V} x_{0,j} = n & (2)\\
& x_{0,i} + \sum_{(j,i) \in A} x_{j,i} - \sum_{(i,j) \in A} x_{i,j} = 1, \quad \forall\, v_i \in V & (3)\\
& k z_i \leq x_{0,i} \leq n z_i, \quad \forall\, v_i \in V & (4)\\
& \sum_{i \in V} z_i \leq \left\lfloor \frac{n}{k} \right\rfloor & (5)\\
& \sum_{e \in E} y_e + \sum_{i \in V} z_i = n & (6)\\
& \left.\begin{array}{l} x_{i,j} \leq (n-1) y_e \\ x_{j,i} \leq (n-1) y_e \end{array}\right\} \quad \forall\, e = \{v_i, v_j\} \in E & (7)\\
& x_{i,j} \geq 0, \quad \forall\, (v_i, v_j) \in A & \\
& y_e \in \{0,1\}, \quad e \in E & \\
& z_i \in \{0,1\}, \quad v_i \in V. &
\end{aligned}
$$

A variável binária de decisão y_e, para $e \in E$, corresponde à inclusão ou não do arco e no subgrafo de G. Para cada $i \in V$, temos uma variável binária de decisão z_i, cujo valor corresponde à inclusão ou não do arco artificial (v_0, v_i) no subgrafo de G. Para cada arco (v_i, v_j), definimos a variável contínua $x_{i,j}$ cujo valor corresponde ao fluxo que passa por esse arco. Para cada vértice $j \in V$, definimos a variável contínua $x_{0,j}$ cujo valor corresponde ao fluxo que passa pelo arco (v_0, v_j). Claramente, o fluxo em um arco (v_i, v_j), $v_i, v_j \in V$ deve ser zero se este não não estiver contido no subgrafo da solução.

As restrições (2) e (3) definem a quantidade de fluxo a ser enviada aos vértices em V de tal forma que cada um destes recebe exatamente uma unidade de fluxo. As restrições (4) e

(5) fazem com que o fluxo seja enviado de tal forma que as arestas correspondentes aos arcos utilizados definam componentes conexas com pelo menos k vértices.

Finalmente, o grupo de restrições (7) força que, para cada aresta não selecionada (i.e., as arestas e com $y_e = 0$), o fluxo através do arco correspodente seja zero; caso contrário, o fluxo é limitado por $n-1$, que é o fluxo máximo possível no arco (v_i, v_j), $v_i, v_j \in V$, devido às restrições (2) e (3).

O seguinte lema caracteriza as soluções obtidas pelo modelo.

Lema 4.1 *As componentes conexas obtidas em uma solução ótima do modelo (ND) são árvores.*

Prova Seja (x, y, z) uma solução ótima para o modelo (ND). Suponhamos que haja uma componente conexa F induzida pela solução (x, y, z) que não seja uma árvore. Então, F contém um ciclo H. Seja $E(H)$ o conjunto de arestas no ciclo H. Sabemos que $c(H) > 0$, de acordo com a suposição feita no início desta seção. Considere a seguinte quantidade:

$$\epsilon = \min_{(v_i,v_j)\in E(H)} \max\{x_{i,j}, x_{j,i}\}.$$

Agora, considere o conjunto $H_\epsilon = \{(v_i, v_j) \in E(H) | x_{i,j} = \epsilon\}$ e a seguinte solução (x^*, y^*, z^*):

$$\begin{aligned}
x^*_{i,j} &= x_{i,j} - \epsilon \qquad \forall (v_i, v_j) \in E(H) \\
x^*_{i,j} &= x_{i,j} \qquad \forall (v_i, v_j) \in G - E(H) \\
y^*_e &= 0 \qquad \forall (v_i, v_j) \in H_\epsilon \\
y^*_e &= y_e \qquad \forall (v_i, v_j) \in G - H_\epsilon \\
z^*_i &= z_i \qquad \forall i \in V.
\end{aligned}$$

Pode-se perceber que a nova solução (x^*, y^*, z^*) é viável para o modelo (ND). Além disso, como $c(H) > 0$ e $\epsilon > 0$, a solução (x^*, y^*, z^*) tem custo menor do que o custo da solução corrente, contradizendo sua otimalidade. ■

4.1 Restrições de Quebra de Simetria

Dependendo do valor do parâmetro k, várias soluções diferentes para o modelo (ND) podem representar a mesma solução viável do PAkC. De fato, por meio de uma escolha adequada de valores das variáveis z_i é tipicamente possível se descrever exatamente a mesma solução do PAkC de várias formas diferentes. Este fenômeno é comumente chamado de *simetria no modelo* e advém do fato de que cada componente $C \subseteq V$ em uma solução viável de (ND) deve ter exatamente uma variável z_i igual a 1, para algum $i \in C$, embora não haja restrição em relação a qual das variáveis z_i ($i \in C$) deve assumir o valor 1. Portanto, em uma solução viável onde C é uma componente, existem $|C|$ maneiras de se descrever exatamente a mesma componente C em termos do conjunto das variáveis de decisão envolvidas, uma para escolha de $z_i = 1$ ($i \in C$).

Claramente, esta simetria pode ter um efeito bastante negativo durante um procedimento de enumeração implícita, tal como um algoritmo de *branch-and-bound* baseado em programação linear (ver, por exemplo, Sherali (2001)). A seguir, propomos dois conjuntos de restrições, os quais podem ser utilizados, separadamente, com o propósito de reduzir o efeito de simetria durante a resolução do modelo (ND).

No que se segue, faremos referência ao vértice associado à variável não-zero z_i como o *vértice de referência* da componente C. O vértice de referência da componente C é o único vértice de C que é ligado ao vértice artificial de origem do fluxo no modelo (ND).

4.1.1 Folhas como Vértices de Referência

Este conjunto de restrições força que o vértice de referência de uma componente C seja uma folha, *i.e.*, um vértice de C com um único vizinho. Já que C corresponde a uma árvore (pelo Lema 4.1), podemos garantir que C contém, no mínimo, um vértice com um único vizinho , desde que $k > 1$ (o caso $k = 1$ é trivial). Portanto, este conjunto de restrições não elimina nenhuma solução ótima do PAkC, ao mesmo tempo em que elimina certas soluções viáveis de (ND): aquelas que utilizam um vértice não-raiz como o vértice de referência de uma componente. Perceba que podemos ainda ter vários vértices que satisfazem as condições para serem candidatos a ser o vértice de referência de uma dada componente. O número de tais vértices candidatos, contudo, é menor do que o tamanho da componente em questão. Desta forma, o benefício resultante do uso deste tipo de conjunto de restrições fica evidente na redução do espaço de busca.

O conjunto de restrições pode ser expresso da seguinte maneira:

$$\sum_{e=\{v_i,v_j\}\in E} y_e \leq 1 + (n-2)(1-z_i),\ \forall i \in V. \tag{8}$$

É interessante notar que (8) também pode ser utilizado com o intuito de se obter soluções viáveis de grau limitado: substituindo-se o fator $(n-2)$ no lado direito da desigualdade por um inteiro positivo $d < n-2$, podemos forçar cada vértice não-referência a ter grau menor ou igual a $d+1$. Isto pode nos permitir obter rapidamente soluções viáveis de qualidade razoável, já que o espaço de busca pode ser reduzido drasticamente com uma escolha de d pequeno.

4.1.2 Vértices de Índice Mínimo como Vértices de Referência

Este conjunto de restrições força que o vértice de referência de uma componente C seja o vértice de C com o menor *índice* em C. Em outras palavras, usamos os índices $1, 2, 3, \ldots, n$ dos vértices $v_1, v_2, v_3, \ldots, v_n$ (respectivamente) para forçar o algoritmo de solução do modelo a escolher, como o vértice de referência da componente C, aquele de menor índice dentre os vértices que pertencem a C. Assim como no caso do conjunto de restrições (8), também fica claro que esta condição não remove do espaço de busca nenhuma solução ótima do problema, ao mesmo tempo em que identifica um único vértice de cada componente C como candidato a vértice de referência de C. O fato de termos um vértice de referência identificado unicamente remove o impacto potencial de se explorar um grande número de soluções de (ND), muitas das quais poderiam, na verdade, representar precisamente a mesma solução do PAkC.

O conjunto de restrições pode ser expresso da seguinte maneira:

$$y_e + z_j \leq 1,\ \forall e = \{v_i, v_j\} \in E,\ \text{com } i < j. \tag{9}$$

Como subproduto da abordagem por índice mínimo, podemos também fixar em zero os valores de algumas variáveis z_i, desde que cada vértice v_i associado a uma delas jamais seja o vértice de menor índice dentre os vértices pertencentes à mesma componente de v_i. Obviamente, decidir quais conjuntos de variáveis poderão ser fixados desta maneira é uma tarefa que depende da topologia do grafo. Podemos, no entanto, garantir que:

$$\sum_{i=n-k+2}^{n} z_i = 0. \tag{10}$$

Por fim, vale a pena destacar que, embora os conjuntos de restrições (8) e (9)-(10) possam ser utilizados conjuntamente, tal uso simultâneo pode resultar em um problema inviável, já que estaríamos exigindo que o vértice de menor índice em cada componente fosse uma folha.

Nos testes computacionais reportados na Seção 5, utilizamos o conjunto de restrições de quebra de simetria dados pelas desigualdades (9) e (10) acima.

5 Resultados Computacionais

Com o intuito de conduzir uma avaliação empírica dos algoritmos discutidos neste artigo, utilizamos um conjunto de instâncias do PAkC geradas aleatoriamente, conforme descrito a seguir. Cada instância consiste de um grafo completo, com cada vértice associado a um ponto no plano, cujas coordenadas são geradas aleatoriamente. O custo $c(e)$ de cada aresta $e = \{v, w\}$ é dado pela distância Euclidiana entre os pontos associados a v e w. O conjunto de instâncias utilizado aqui pode ser obtido juntamente aos autores deste trabalho.

Para efeito de apresentação dos resultados, dividimos as instâncias em dois grupos: um dito de "pequeno porte", denominado *Grupo 1*; e um grupo de instâncias de "médio porte", denominado *Grupo 2*. As instâncias do Grupo 1 foram utilizadas como base para avaliarmos (i) a qualidade das soluções produzidas pela heurística HEF em relação às soluções produzidas pelo algoritmo *branch-and-bound* (B&B); e (ii) a qualidade das soluções do algoritmo B&B em relação às melhores soluções (muitas delas comprovadamente ótimas) obtidas por meio do uso do resolvedor linear inteiro Xpress-MP (ver FICO (2010)). Para as instâncias do Grupo 2, o modelo (ND) descrito na Seção 4 consome uma quantidade excessiva de tempo. Por esta razão, optamos por utilizar tais instâncias apenas para comparação da qualidade das soluções obtidas pela heurística HEF em relação àquelas produzidas pelo algoritmo B&B.

O Grupo 1 possui instâncias com número de vértices variando de 10 a 50 e com k tomando os seguintes valores: 3, 4, 5, 7 e 10, conforme pode ser visto na Tabela 1. O Grupo 2 possui instâncias com número de vértices variando de 100 a 500 e k assumindo os valores 5, 10, 20 e 40, conforme mostrado na Tabela 2.

Em cada tabela, temos o número de vértices n e o valor de k para cada instância, seguidos dos valores da função objetivo e do tempo de execução relativos a cada algoritmo. No caso dos valores da função objetivo relativos ao Xpress-MP na Tabela 1, exibimos o valor da melhor solução obtida (coluna "Lim. Sup.", denotando "limite superior"), juntamente com o valor do "limite inferior" (coluna "Lim. Inf.") obtido pelo software. Estes limites fornecem uma estimativa de quão próximo do custo ótimo se encontra o custo da melhor solução obtida pelo Xpress-MP.

5.1 Características dos Experimentos

Nossos experimentos foram executados em um computador com processador Intel Core 2 Duo CPU E8400, de GHz 3.00GHz, memória RAM de 4GB RAM e com sistema operacional Windows 7 de 64 bits. O resolvedor linear inteiro utilizado para os testes do nosso modelo de programação inteira mista foi o Xpress-MP Optimizer Version 21.01.00 64 bits. As heurísticas foram implementadas na linguagem Java 1.6 e executadas no ambiente JRE 1.6.0_23 64 bits.

O algoritmo B&B teve sua busca limitada, sendo forçado a interromper sua execução assim que a busca explorasse um determinado número máximo de nós da árvore de B&B. Para as instâncias do Grupo 1 este número foi escolhido como 10.000, enquanto o limite de 20.000 foi

utilizado para instâncias do Grupo 2. Tais valores foram determinados após extensa experimentação com os algoritmos. Sob este ponto de vista, devemos encarar os resultados baseados no algoritmo B&B como uma heurística, já que estamos reportando resultados baseados em uma busca possivelmente truncada. Deve ficar claro, no entanto, que o algoritmo obtém a solução ótima para o grafo fornecido, dados tempo e memória suficientes. Por sua vez, o software Xpress-MP teve seu tempo máximo de execução limitado a 5 minutos em todos os experimentos. Tanto no caso de nosso algoritmo B&B, como no do resolvedor Xpress, reportamos o valor da função objetivo na melhor solução obtida.

5.2 Análise dos Resultados

De acordo com a Tabela 1, podemos observar que o resolvedor Xpress-MP obteve soluções inteiras para quase todas as instâncias do Grupo 1, tendo obtido soluções ótimas para 9 das 24 instâncias. Podemos concluir, ainda, que o modelo (ND) mostrou-se bastante robusto para esse grupo de instâncias, já que, em aproximadamente 80% delas, ou obteve a solução ótima ou uma solução inteira que se mostrou melhor do que as soluções da heurística HEF e do B&B.

		Valor da Função Objetivo				Tempo de Execução (s)		
				Xpress-MP				
n	k	HEF	B&B	Lim. Sup.	Lim. Inf.	HEF	B&B	Xpress-MP
10	3	2.37	2.52	**2.09**	2.09	0.00	0.00	0.40
15	3	1.97	1.97	**1.90**	1.90	0.00	0.00	1.70
20	3	**2.09**	**2.09**	**2.09**	2.09	0.00	0.01	0.20
25	3	**1.99**	**1.99**	**1.99**	1.99	0.00	0.02	0.80
30	3	3.32	3.22	**3.16**	2.82	0.00	0.05	300.00
35	3	3.04	3.04	**2.86**	2.53	0.00	0.06	300.00
40	3	3.07	**2.97**	**2.97**	2.64	0.01	0.50	300.00
45	3	3.25	3.25	**3.22**	2.80	0.01	0.28	300.00
50	3	3.20	**3.12**	—	3.71	0.00	0.90	300.00
10	4	**1.96**	**1.96**	**1.96**	1.96	0.00	0.00	0.10
15	4	**2.47**	**2.47**	**2.47**	2.47	0.00	0.00	0.30
20	4	2.92	2.74	**2.69**	2.57	0.00	0.00	300.00
25	5	**2.74**	**2.74**	**2.74**	2.74	0.00	0.00	11.80
30	5	3.56	3.56	**3.43**	3.43	0.00	0.00	254.30
35	5	3.22	**3.20**	3.22	2.76	0.00	0.02	300.00
40	5	3.98	**3.85**	**3.85**	3.52	0.01	0.06	300.00
45	5	4.26	4.26	**4.21**	3.67	0.01	0.12	300.00
50	5	4.29	**4.26**	—	3.71	0.01	0.78	300.00
25	7	3.17	3.17	**3.03**	2.83	0.00	0.00	300.00
30	7	3.58	3.58	**3.45**	3.06	0.01	0.00	300.00
35	7	4.15	4.15	**3.98**	3.98	0.00	0.01	127.00
40	10	4.30	**4.24**	4.32	3.89	0.01	0.01	300.00
45	10	4.66	4.66	**4.56**	4.20	0.01	0.01	300.00
50	10	**4.41**	**4.41**	4.70	3.80	0.01	0.01	300.00

Tabela 1: Valores da função objetivo e de tempo de execução para instâncias do Grupo 1.

Ao compararmos os resultados do algoritmo B&B com aqueles obtidos pelo Xpress-MP na Tabela 1, verificamos que o algoritmo B&B se mostrou bastante competitivo. Em 5 casos, obteve uma solução inteira melhor do que a do Xpress-MP; em outras 5 instâncias atingiu o ótimo, e em duas instâncias obteve uma solução melhor do que as dos demais algoritmos. Além disso, verificamos que a solução obtida pelo B&B foi apenas 2,11% pior, em média, do que a melhor solução obtida pelo Xpress-MP em um tempo de execução bem maior.

		Valor da F. Obj.		Tempo de Execução (s)	
n	k	HEF	B&B	HEF	B&B
100	5	6.52	**6.46**	0.03	1.73
120	5	6.71	**6.54**	0.04	2.39
150	5	8.00	**7.96**	0.06	2.47
200	5	**9.10**	9.14	0.06	2.44
300	5	11.70	**11.65**	0.06	4.06
400	5	**13.53**	13.74	0.07	5.83
500	5	**15.92**	15.99	0.08	6.13
100	10	**6.74**	**6.74**	0.03	0.89
120	10	7.34	**7.27**	0.04	2.12
150	10	8.97	**8.94**	0.02	1.91
200	10	10.27	**10.18**	0.04	2.72
300	10	13.03	**12.96**	0.06	3.76
400	10	**15.47**	15.54	0.10	4.79
500	10	18.15	**18.12**	0.21	5.32
120	20	**8.01**	**8.01**	0.04	0.37
150	20	**8.89**	**8.89**	0.06	0.57
200	20	**10.85**	**10.85**	0.06	2.42
300	20	**13.62**	13.63	0.06	3.68
400	20	16.26	**16.23**	0.07	4.55
500	20	18.73	**18.66**	0.15	4.65
150	40	9.67	**9.65**	0.06	0.01
200	40	**11.37**	**11.37**	0.06	0.42
300	40	14.42	**14.36**	0.07	2.16
400	40	**16.83**	**16.83**	0.07	2.76
500	40	**19.51**	19.55	0.13	4.46

Tabela 2: Valores da função objetivo e de tempo de execução para instâncias grandes.

Ainda na Tabela 1, quando comparamos a heurística HEF com o algoritmo B&B, verificamos que o B&B foi melhor em 8 casos (33%). No entanto, o B&B obteve a mesma solução que HEF em 15 instâncias, ou seja, 62,5% dos casos. Isso mostra que a solução obtida por HEF é de muito boa qualidade, ainda mais quando observamos que as soluções obtidas pelo B&B são freqüentemente ótimas ou próximas disso. Para todas as instâncias testadas, o tempo de execução do modelo (ND) foi superior aos tempos de execução da heurística HEF e do B&B. Em 15 casos, foi necessário interromper a execução do modelo após 5 minutos, enquanto que HEF e B&B concluíram suas execuções em menos de um segundo para todas as instâncias.

Na Tabela 2, verificamos que houve um domínio significativo, em termos de qualidade da solução, do algoritmo B&B sobre a heurística HEF. Em mais da metade dos casos, o algoritmo

B&B obteve soluções melhores do que HEF, e, em outros 24%, o B&B obteve a mesma solução que HEF. Ademais, embora os tempos de execução do B&B tenham sido consistentemente maiores do que os da heurística HEF, um acréscimo de 2 a 3 segundos (média de 2.8s) sobre o tempo de execução de HEF é aceitável para a maioria dos propósitos práticos. Por fim, podemos perceber que, embora a heurística HEF tenha sido dominada pelo algoritmo B&B em relação à qualidade de solução, os custos das soluções do algoritmo B&B são apenas 0,22% menores, em média, do que aqueles das soluções obtidas por HEF.

6 Conclusões

Neste trabalho, apresentamos um novo algoritmo *branch-and-bound* (B&B) e um novo modelo de programação matemática para o problema PAkC. O algoritmo B&B mostrou-se bastante competitivo, tendo sido utilizado para resolver o PAkC sobre uma árvore geradora mínima do grafo original, fornecendo uma solução viável de boa qualidade. As soluções fornecidas pelo B&B foram melhores, em média, do que as encontradas pela heurística mais bem-sucedida para o PAkC, incorrendo em um pequeno custo adicional de tempo de execução.

O modelo de programação matemática mostrou-se útil para instâncias de pequeno porte, para as quais soluções ótimas ou quase ótimas podem ser encontradas em uma quantidade de tempo razoável. Outra característica interessante do modelo é sua natureza estática: para resolver o modelo não se faz necessário o uso de técnicas de geração de colunas e/ou de restrições. Por fim, os resultados obtidos por meio do uso do modelo sugerem que as soluções produzidas pelo algoritmo B&B tendem a apresentar custos bastante próximos do custos ótimos.

Como trabalhos futuros, podemos destacar: o uso de estratégias de busca local, possivelmente guiadas por uma meta-heurística; a proposta de novas heurísticas combinatórias; e a utilização de diferentes métricas para o cálculo de distâncias entre vértices.

Referências

O. Borůvka. O jistém problému minimálním. *Praca Moravske Prirodovedecke Spolecnosti*, 3: 37–58, 1926.

B. Chazelle. A minimum spanning tree algorithm with inverse-Ackermann type complexity. *Journal of the ACM*, 47(6):1028–1047, 2000.

P. Crescenzi and V. Kann. A Compendium of NP Optimization Problems, 1998. URL `http://www.csc.kth.se/ viggo/problemlist/`.

FICO. Xpress-MP Optimizer 21.01.00, 2010. URL `http://www.fico.com/xpress/`.

M. Goemans and D. Williamson. General approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

C. Imielińska, B. Kalantari, and L. Khachiyan. A Greedy Heuristic for a Minimum-Weight Forest Problem. *Operations Research Letters*, 14(2):65–71, 1993.

X. Ji. *Graph Partition Problems with Minimum Size Constraints*. PhD thesis, Rensselaer Polytechnic Institute, 2004.

J. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

M. Laszlo and S. Mukherjee. Another greedy heuristic for the constrained forest problem. *Operations Research Letters*, 33(6):629–633, 2005a.

M. Laszlo and S. Mukherjee. Minimum Spanning Tree Partitioning Algorithm for Microaggregation. *IEEE Transactions on Knowledge and Data Engineering*, 17(7):902–911, 2005b.

M. Laszlo and S. Mukherjee. A Class of Heuristics for the Constrained Forest Problem. *Discrete Applied Mathematics*, 154(1):6–14, 2006.

R. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401, 1957.

H. Sherali and J. Smith. Improving Discrete Model Representations via Symmetry Considerations. *Management Science*, pages 1396–1407, 2001.

R. Tarjan. *Data Structures and Network Algorithms*, volume 44. Society for Industrial Mathematics (SIAM), 1983. ISBN 0898711878.