EasyChair Preprint

№ 562

Exact method for maximum diversity problem

Pablo Luiz Braga Soares, Manoel Bezerra Campêlo Neto and Daniel Nogueira Rebouças

October 6, 2018

Método exato para o problema da diversidade máxima

Pablo Luiz Braga Soares
Universidade Federal do Ceará, Campus de Russas, Russas/CE, Brasil.
pablo.soares@ufc.br

Manoel Campêlo*
Universidade Federal do Ceará, Campus do Pici, Bloco 910, DEMA, Fortaleza/CE, Brasil.
mcampelo@lia.ufc.br

Daniel Nogueira Rebouças
Universidade Federal do Ceará, Campus de Russas, Russas/CE, Brasil.
danielnreboucas@hotmail.com

Resumo

Revisitamos uma formulação quadrática para o problema da diversidade máxima (MDP). Aplicamos a técnica da t–linearização e fortalecemos as restrições resultantes. Propomos novas regras para fixação de variáveis, também traduzidas como restrições válidas para o problema. A partir desses ingredientes, propomos um algoritmo exato para o MDP, baseado no método branch-and-bound. Resultados computacionais obtidos com instâncias da literatura mostram que o método proposto é capaz de resolver instâncias com até 125 elementos, sendo mais eficiente que o melhor algoritmo exato da literatura.

Keywords: Problema da Diversidade Máxima; t–Linearização; Lifting; Desigualdade válida.

1 Introdução

O problema da diversidade máxima (*Maximum Diversity Problem* – MDP) pertence a uma família de problemas de dispersão e diversidade, cujo objetivo é identificar, a partir de um conjunto V com n elementos, um subconjunto S com m ($m < n$) elementos, de tal forma que a distância entre os pares dos elementos em S seja máxima [11]. Genericamente, o MDP pode ser definido sobre um grafo não-direcionado $G = (V, E)$, ponderado em arestas, onde $V = \{1, \ldots, n\}$ e $c_{ij} \in \mathbb{R}_+$ denota o peso da aresta (i, j). Por simplicidade, consideramos $c_{ji} = c_{ij}$, e $c_{ij} = 0$ quando $(i, j) \notin E$. Formalmente, MDP pode ser formulado como um problema quadrático binário [6], da seguinte forma:

$$(F_1) \quad \max \left\{ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} x_i x_j : \sum_{i=1}^{n} x_i = m, \quad x_i \in \{0,1\}, \quad 1 \leq i \leq n \right\}, \tag{1}$$

onde a variável binária $x_i = 1$, se o vértice $i \in S$, e $x_i = 0$ se $i \in V \backslash S$. A restrição de cardinalidade $\sum_{i=1}^{n} x_i = m$ garante que uma solução viável x contém exatamente m vértices.

*Parcialmente financiado por CNPq 443747/2014-8, 305264/2016-8.

Sendo um problema NP-Difícil [6], a variedade dos métodos que têm sido propostos são divididos em duas principais categorias: algoritmos exatos e heurísticas. Na categoria dos algoritmos exatos, onde se enquadra nosso estudo, destacamos o trabalho de [7], que propôs um método exato baseado em um brand-and-bound com ramificação n-ária. Os autores também mostram propriedades usadas pelo método na obtenção de bons limites superiores.

A não-linearidade em problemas de programação inteira é normalmente tratada com técnicas que envolvem uma aproximação linear por partes [2, 5] ou a transformação da função não-linear em uma função polinomial, a seguir convertida em uma função linear de variáveis 0–1 [1, 4]. Na maioria das vezes, a não-linearidade em problemas de programação inteira aparece já na forma polinomial, sendo que um número significativo dos casos envolve apenas termos de segunda ordem [3].

Recentemente, a linearização proposta em [8], no contexto do Problema Quadrático da Mochila (PQM), tem sido usada para tratar outros problemas que possuem função objetivo quadrática [9, 10]. Essa linearização consiste basicamente de duas etapas. Primeiro, substitui-se o termo quadrático da função objetivo por uma variável real t, que é limitada superiormente pela expressão quadrática, com a inclusão de uma restrição adicional. Depois, essa restrição quadrática é substituída por um conjunto exponencial de restrições lineares, que definem os mesmos pontos inteiros.

Propomos nesse trabalho um algoritmo exato, branch-and-bound, para o MDP, composto por 3 elementos principais, que serão descritos nas próximas seções. Na Seção 2, mostramos como é feita a aplicação da t–linearização ao MDP. Na Seção 3, mostramos como fortalecer as desigualdades da t-linearização, usando a restrição de cardinalidade do MDP. Já na Seção 4, derivamos novas restrições válidas para o MDP, generalizando propriedades mostradas em [7]. Descrevemos o método proposto na Seção 5 e fechamos o artigo com um estudo computacional e as conclusões.

2 Aplicação da t–linearização ao MDP

Temos que F_1 equivale a $\max\left\{t : t \leq \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij}x_ix_j, \quad \sum_{i=1}^{n} x_i = m, \quad (x,t) \in \mathbb{B}^n \times \mathbb{R}_+\right\}$, que denotamos $(F_1)_t$. Seja S_n o conjunto de todas as permutações de $\{1,\ldots,n\}$. Note que

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij}x_ix_j = \sum_{i=1}^{n}\sum_{j=i}^{i-1} c_{ji}x_ix_j = \sum_{i=1}^{n}\sum_{j=i}^{i-1} c_{\pi(j)\pi(i)}x_{\pi(i)}x_{\pi(j)} \quad \forall \pi \in S_n$$

e, pelo teorema mostrado em [8], podemos reescrever F_1 como

$$(F_1)_t^{\pi} \quad \max\left\{t : t \leq \sum_{i=1}^{n}\sum_{j=1}^{i-1} c_{\pi(j)\pi(i)}x_{\pi(i)} \quad \forall \pi \in S_n, \quad \sum_{i=1}^{n} x_i = m, \quad (x,t) \in \mathbb{B}^n \times \mathbb{R}_+\right\} \qquad (2)$$

Mais ainda, a separação das restrições lineares que definem $(F_1)_t^{\pi}$ pode ser feita em tempo polinomial, como mostrado em [8]. A extensão desses resultados para coeficientes c_{ij} arbitrários pode ser encontrado em [9].

3 Fortalecimento das restrições da t–linearização

A presença da restrição de cardinalidade permite fortalecer as restrições da t–linearização. Sendo os pesos não-negativos, elas podem ser substituídas conforme o seguinte teorema, que se baseia na ideia apresentada em [8] para o problema quadrático da mochila.

Teorema 3.1. *Para $i = 1, \ldots, n$ e $\pi \in S_n$, seja $s_{\pi(i)}$ a soma dos $r := \min\{i-1, m-1\}$ maiores valores do conjunto $\{c_{\pi(j)\pi(i)} : j = 1, \ldots, i-1\}$. Então as desigualdades*

$$t \leq \sum_{i=1}^{n} s_{\pi(i)} x_{\pi(i)}, \text{ para todo } \pi \in S_n, \tag{3}$$

são válidas para (2).

Demonstração. Sejam $(x, t) \in \mathbb{B}^n \times \mathbb{R}_+$ viável para $(F_1)_t$ e $\pi \in S_n$. Então

$$t \leq \sum_{i=1}^{n} \sum_{j=1}^{i-1} c_{ji} x_i x_j \quad = \quad \sum_{i=1}^{n} \sum_{j=1}^{i-1} c_{\pi(j)\pi(i)} x_{\pi(i)} x_{\pi(j)} = \sum_{i=1}^{n} \left(\sum_{j=1}^{i-1} c_{\pi(j)\pi(i)} x_{\pi(j)} \right) x_{\pi(i)}$$

Como $x \in \mathbb{B}^n$ e $\sum_{i=1}^{n} x_{\pi(i)} = m$, $x_{\pi(i)} = 1$ implica $\sum_{j=1}^{i-1} x_{\pi(j)} \leq \min\{i-1, m-1\} = r$. Portanto, se $x_{\pi(i)} = 1$, temos que $\sum_{j=1}^{i-1} c_{\pi(j)\pi(i)} x_{\pi(j)} \leq s_{\pi(i)}$. Segue então que $t \leq \sum_{i=1}^{n} s_{\pi(i)} x_{\pi(i)}$. □

4 Novas restrições válidas para MDP

Uma solução parcial $\bar{x} \in \mathbb{B}^n$ para MDP é o vetor de incidência de um subconjunto de k vértices de V, com $k \leq m$. Dado uma solução parcial $\bar{x}$, definimos

$$V_0(\bar{x}) = \{v \in V : \bar{x}_v = 0\}, \quad V_1(\bar{x}) = \{v \in V : \bar{x}_v = 1\} \text{ e } V_2(\bar{x}) = V \backslash (V_1(\bar{x}) \cup V_0(\bar{x})).$$

Uma solução parcial $\hat{x}$ é descendente de outra solução parcial $\bar{x}$ se $V_1(\hat{x}) \supseteq V_1(\bar{x})$ e $V_0(\hat{x}) \supseteq V_0(\bar{x})$. A solução parcial é completa quando $|V_1(\bar{x})| = m$, $|V_0(\bar{x})| = n - m$ e $V_2(\bar{x}) = \emptyset$. Partindo de uma solução parcial $\bar{x}$, podemos obter uma solução completa descendente, transferindo $m - |V_1(\bar{x})|$ elementos de $V_2(\bar{x})$ para $V_1(\bar{x})$ e os demais para $V_0(\bar{x})$. Nessa seção $d(a, b)$ representa o peso da aresta entre os vértices a e b, ou seja, $d(a, b) = c_{ab}$.

Seja $S(x) = \{s_1, s_2, \ldots, s_m\}$ o conjunto de vértices selecionados em uma solução completa x. Podemos expressar a função objetivo de (1) em x como

$$f(x) = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} d(s_i, s_j) = \sum_{j=1}^{m} d(s_i),$$

onde $d(s_i) = \frac{1}{2} \sum_{j=1}^{m} d(s_i, s_j)$ representa a contribuição de s_i em $f(x)$. A partir dessa expressão, podemos introduzir o potencial máximo($d_{max}(v)$) e mínimo($d_{min}(v)$) que um vértice v pode contribuir para qualquer solução, da seguinte forma:

$$d_{min}(v) = \frac{1}{2} \sum_{j=1}^{m-1} d(v, v_{\sigma(n-j)}), \quad d_{max}(v) = \frac{1}{2} \sum_{j=1}^{m-1} d(v, v_{\sigma(j)}),$$

onde $d(v, v_{\sigma(1)}), d(v, v_{\sigma(2)}), \ldots, d(v, v_{\sigma(n-1)})$ representa o peso das arestas que conectam v aos outros $n-1$ vértices do grafo em ordem descendente. Observe que, a partir da definição acima, qualquer solução contendo o vértice v irá satisfazer as desigualdades

$$d_{min}(v) \leq d(v) \leq d_{max}(v), \tag{4}$$

mostradas e usadas por [7] para estabelecer a Proposição 4.1.

Proposição 4.1. *Dado uma solução ótima x e dois vértices $u, v \in V$, se $d_{max}(u) < d_{min}(v)$ e v não está selecionado na solução x, então u não pode ser selecionado em x.*

A Proposição 4.1 foi fortemente usada no desenvolvimento do algoritmo branch-and-bound proposto por [7]. Observe que podemos calcular melhores valores para d_{max} e d_{min} quando consideramos que alguns vértices já foram selecionados, ou seja, quando temos uma solução parcial $\bar{x}$. Dada uma solução parcial $\bar{x}$ e $v \in V_2(\bar{x})$, denotemos por $V_2^{max}(\bar{x}, v)$ e $V_2^{min}(\bar{x}, v)$ os $m - |V_1(\bar{x})| - 1$ vértices de $V_2(\bar{x})$ com os maiores e menores valores $d(v, w)$, para $w \in V_2(\bar{x})$. A partir dessas definições, vamos introduzir o potencial máximo($d_{max}(\bar{x}, v)$) e mínimo($d_{min}(\bar{x}, v)$) que um vértice $v \in V_2(\bar{x})$ pode contribuir, caso seja selecionado para fazer parte da nova solução parcial $\hat{x}$ que descende de $\bar{x}$.

$$d_{min}(\bar{x}, v) = \sum_{w \in V_1(\bar{x})} d(v, w) + \sum_{w \in V_2^{min}(\bar{x}, v)} d(v, w), \quad d_{max}(\bar{x}, v) = \sum_{w \in V_1(\bar{x})} d(v, w) + \sum_{w \in V_2^{max}(\bar{x}, v)} d(v, w).$$

Usando essas definições, derivamos propriedades para fixação de variáveis como a seguir.

Proposição 4.2. *Seja $\bar{x}$ uma solução parcial e x^* uma melhor solução completa que descende de $\bar{x}$. Sejam $u, v \in V_2(\bar{x})$ tais que $d_{max}(\bar{x}, u) < d_{min}(\bar{x}, v)$. Se $x_v^* = 0$ então $x_u^* = 0$ (ou equivalentemente, Se $x_u^* = 1$ então $x_v^* = 1$).*

Demonstração. Suponha, por absurdo, que $x_v^* = 0$ e $x_u^* = 1$. Então $V_1(x^*) = V_1(\bar{x}) \cup \{u\} \cup W$, onde $W \subseteq V_2(\bar{x}) \backslash \{u, v\}$, $|W| = m - |V_1(\bar{x})| - 1$, e $v \notin V_1(x^*)$. Defina $\hat{x}$ uma solução completa que também descende de $\bar{x}$ tal que $V_1(\hat{x}) = V_1(\bar{x}) \cup \{v\} \cup W$. Seja $f(x^*)$ o valor da solução completa x^* e $f(\hat{x})$ o valor da solução completa $\hat{x}$. Temos que

$$\begin{aligned} f(x^*) - f(\hat{x}) &= \sum_{w \in V_1(\bar{x})} d(u, w) + \sum_{w \in W} d(u, w) - \left(\sum_{w \in V_1(\bar{x})} d(v, w) + \sum_{w \in W} d(v, w) \right) \\ &\leq d_{max}(\bar{x}, u) - d_{min}(\bar{x}, v) < 0 \end{aligned}$$

Temos um absurdo, pois x^* é ótimo. □

Podemos fortalecer a Proposição 4.2 como segue. Dados $u, v \in V_2(\bar{x})$, seja $V_{uv}(\bar{x})$ o conjunto com $m - |V_1(\bar{x})| - 1$ vértices $w \in V_2(\bar{x}) \backslash \{u, v\}$ com as maiores diferenças $[d(u, w) - d(v, w)]$ e defina

$$\delta_{uv}(\bar{x}) = \sum_{w \in V_1(\bar{x})} [d(u, w) - d(v, w)] + \sum_{w \in V_{uv}(\bar{x})} [d(u, w) - d(v, w)].$$

Proposição 4.3. *Seja $\bar{x}$ uma solução parcial e x^* uma melhor solução completa que descende de $\bar{x}$. Sejam $u, v \in V_2(\bar{x})$ tais que $\delta_{uv}(\bar{x}) < 0$. Se $x_v^* = 0$ então $x_u^* = 0$.*

Demonstração. Suponha, por absurdo, que $x_v^* = 0$ e $x_u^* = 1$. Então $V_1(x^*) = V_1(\bar{x}) \cup \{u\} \cup W$, onde $W \subseteq V_2(\bar{x}) \backslash \{v, u\}$, $|W| = m - |V_1(\bar{x})| - 1$. Note que $v \notin V_1(\bar{x})$. Defina a solução completa $\hat{x}$, que também descende de $\bar{x}$, tal que $V_1(\hat{x}) = V_1(\bar{x}) \cup \{v\} \cup \{W\}$. Temos que

$$\begin{aligned} f(x^*) - f(\hat{x}) &= \sum_{w \in V_1(\bar{x}) \cup W} d(u, w) - \sum_{w \in V_1(\bar{x}) \cup W} d(v, w) = \sum_{w \in V_1(\bar{x}) \cup W} [d(u, w) - d(v, w)] \\ &\leq \sum_{w \in V_1(\bar{x}) \cup V_{uv}(\bar{x})} [d(u, w) - d(v, w)] = \delta_{uv}(\bar{x}) < 0 \end{aligned}$$

Temos um absurdo, pois x^* é ótimo. □

Alternativamente às proposições de podas, podemos usar tais critérios para definir desigualdades válidas, como veremos na Proposição (4.4). Dados uma solução parcial $\bar{x}$ e $u, v \in V_2(\bar{x})$, definimos:

$$\delta^1_{uv}(\bar{x}) = \sum_{w \in V_1(\bar{x})} [d(u,w) - d(v,w)],$$

$$M^c_{uv}(\bar{x}) = \text{ soma dos } m - |V_1(\bar{x})| \text{ menores valores } \{d(u,w) - d(v,w) : w \in V_2(\bar{x}) \backslash \{u,v\}\},$$
$$M^b_{uv}(\bar{x}) = \text{ soma dos } m - |V_1(\bar{x})| - 1 \text{ menores valores } \{d(u,w) - d(v,w) : w \in V_2(\bar{x}) \backslash \{u,v\}\},$$
$$M^a_{uv}(\bar{x}) = \text{ soma dos } m - |V_1(\bar{x})| - 2 \text{ menores valores } \{d(u,w) - d(v,w) : w \in V_2(\bar{x}) \backslash \{u,v\}\},$$
$$M_{uv}(\bar{x}) = \min\{\delta^1_{uv}(\bar{x}) + M^a_{uv}(\bar{x}), \delta^1_{uv}(\bar{x})/2 + M^b_{uv}(\bar{x})/2, \delta^1_{uv}(\bar{x}) + M^c_{uv}(\bar{x})\}.$$

Proposição 4.4. *Sejam $\bar{x}$ uma solução parcial e $u, v \in V_2(\bar{x})$. A desigualdade $\delta_{uv}(x, \bar{x}) \geq M_{uv}(\bar{x})(1 - x_u + x_v)$ é válida para as soluções completas de melhor valor que descendem de $\bar{x}$, onde*

$$\delta_{uv}(x, \bar{x}) = \sum_{w \in V_1(\bar{x})} [d(u,w) - d(v,w)] + \sum_{w \in V_2(\bar{x}) \backslash \{u,v\}} [d(u,w) - d(v,w)] x_w.$$

Demonstração. Seja x uma solução de melhor valor descendente de $\bar{x}$. Considere os seguintes casos:
1) $x_u = 0, x_v = 0$: $\delta_{uv}(x, \bar{x}) \geq \delta^1_{uv}(\bar{x}) + M^c_{uv}(\bar{x}) \geq M_{uv}(\bar{x}) = M_{uv}(\bar{x})(1 - x_u + x_v)$;
2) $x_u = x_v = 1$: $\delta_{uv}(x, \bar{x}) \geq \delta^1_{uv}(\bar{x}) + M^a_{uv}(\bar{x}) \geq M_{uv}(\bar{x}) = M_{uv}(\bar{x})(1 - x_u + x_v)$;
3) $x_u = 0, x_v = 1$: $\delta_{uv}(x, \bar{x}) \geq \delta^1_{uv}(\bar{x}) + M^b_{uv}(\bar{x}) \geq 2M_{uv}(\bar{x}) = M_{uv}(\bar{x})(1 - x_u + x_v)$.
4) $x_u = 1, x_v = 0$: Obtenha solução completa $\hat{x}$, que também descende de $\bar{x}$, tal que $V_1(\hat{x}) = (V_1(x) \backslash \{u\}) \cup \{v\}$. Note que $V_1(x) \backslash \{u\} = V_1(\bar{x}) \cup \{w \in V_2(\bar{x}) \backslash \{u,v\} : x_w = 1\}$. Temos que $f(x) \geq f(\hat{x})$ e, portanto,

$$\begin{aligned} 0 \leq f(x) - f(\hat{x}) &= \sum_{w \in V_1(x) \backslash \{u\}} d(u,w) - \sum_{w \in V_1(x) \backslash \{u\}} d(v,w) \\ &= \sum_{w \in V_1(\bar{x})} [d(u,w) - d(v,w)] + \sum_{w \in V_2(\bar{x}) \backslash \{u,v\}} [d(u,w) - d(v,w)] x_w = \delta_{uv}(x, \bar{x}) \end{aligned}$$

Logo, $\delta_{uv}(x, \bar{x}) \geq M_{uv}(\bar{x})(1 - x_u + x_v) = 0$. □

5 O algoritmo *branch and bound*

O método exato aqui proposto para o MDP compõe-se de 4 procedimentos principais:
1− Conversão de F_1 em $(F_1)^\pi_t$;
2− Separação das restrições de $(F_1)^\pi_t$ conforme procedimento descrito em [8], e fortalecimento da restrição violada segundo Teorema 3.1;
3− Obtenção de uma solução viável x_h (limite inferior) através de uma heurística gulosa, e aplicação do procedimento de separação em x_h, para gerar a primeira restrição fortalecida;
4− Adição à $(F_1)^\pi_t$ das desigualdades válidas $\delta_{uv}(x, \bar{x}) \geq M_{uv}(\bar{x})(1 - x_u + x_v)$, para $\bar{x} = \mathbf{0}$, obtendo assim o modelo $(BB)^\pi_t$:

O modelo $(BB)^\pi_t$ é resolvido pelo algoritmo branch and bound padrão. É importante destacar que o procedimento de separação é usado dentro do branch and bound sempre que uma solução inteira é encontrada, ou seja, as restrições da t-linearização são usadas como *lazy constraints*.

$$
\begin{aligned}
(BB)_t^{\pi} \quad \max \quad & t \\
\text{s.a} \quad & t \leq \sum_{i=1}^{n} s_{\pi(i)} x_{\pi(i)} \quad \forall \pi \in S_n, \\
& \textstyle\sum_{w \in V \setminus \{u,v\}} [d(u,w) - d(v,w)] x_w \geq M_{uv}(\mathbf{0})(1 - x_u + x_v) \quad \forall (u,v) \text{ e } \forall (v,u), \\
& \sum_{i=1}^{n} x_i = m, \quad (x,t) \in \mathbb{B}^n \times \mathbb{R}_+.
\end{aligned}
$$

6 Experimentos Computacionais

Nessa seção, comparamos o desempenho do nosso *branch and bound*, denominado aqui de $(BB)_t^{\pi}$, com o *branch and bound* apresentando por [7], denominado aqui por $BBmax$. Utilizamos a mesma máquina para melhor comparação entre os métodos: processador Intel® Core™ i5 − 4570 com 3.20 GHz, 16 GB RAM e sistema operacional Ubuntu 16.04 LTS. Para implementar o $(BB)_t^{\pi}$, assim como o $BBmax$, usamos o Ambiente de Desenvolvimento Integrado (IDE - Integrated Development Environment) *Code::Blocks* e linguagem C++.

Algumas observações merecem ser feitas com respeito à implementação de $BBmax$. Infelizmente, os autores não mais dispunham do código original que pudesse ser executado agora. Sendo assim, a implementação aqui testada é nossa. A eficiência do $BBmax$ é bastante influenciada pela escolha das estruturas de dados utilizada para armazenar a árvore de enumeração e manter informações necessárias ao longo do processo. Observe que $BBmax$ não faz uso de um solver, como é o caso do $(BB)_t^{\pi}$, que emprega o software comercial IBM/ILOG CPLEX 12.7 para a resolução dos subproblemas gerados pela t-linearização.

Para essa comparação, utilizamos um conjunto de instâncias denominado *Glover2*, disponível em www.optsicom.es/mdp. O conjunto é constituído de 50 instâncias, sendo 5 instâncias para cada par (n, m), onde $n \in \{25, 50, 100, 125, 150\}$ e $m = 0.1n, 0.3n$. Utilizamos três parâmetros de medição: o melhor limite inferior obtido por cada um dos métodos($BBmax$ e $(BB)_t^{\pi}$), LoWeRB; a prova de otimalidade #Opt, ou seja, se o método conseguiu encontrar a solução ótima; e o tempo CPU em segundos, sendo esse último parâmetro limitando a 3600 segundos de execução em cada instância.

A Tabela 1 mostra, a partir da linha 3, o contraste entre os desempenhos dos dois métodos, onde destacamos em negrito os melhores resultados para cada instância. A coluna MelhorSol está preenchida com a melhor solução encontrada entre os dois métodos. Podemos ver que o nosso método conseguiu provar a otimalidade em 68%(34 de 50) das instâncias testadas, enquanto que o $BBmax$ o conseguiu em 52%(26 de 50). É importante destacar também que os tempos de computação de $(BB)_t^{\pi}$ são inferiores em quase todas as instâncias. Vale notar que alguns tempos de execução são um pouco maior que 3600s pelo fato de o CPLEX contabilizar o tempo apenas a cada término de iteração.

7 Conclusão

Nesse trabalho, avançamos um passo à cerca do conhecimento do problema da diversidade máxima (MDP). Desenvolvemos um método baseado em um branch and bound que calcula soluções exatas para o problema. O método proposto é composto de três elementos principais: aplicamos a t–linearização à formulação quadrática do MDP, mostramos como gerar restrições fortalecidas devido

à natureza do problema, e introduzimos novas restrições válidas, que foram desenvolvidas com base em propriedades de poda do problema.

Os experimentos computacionais indicam que nosso método é capaz de resolver, de forma ótima, instâncias de tamanho moderado(até 125 vértices). Comparamos nosso método com uma implementação nossa do melhor método exato proposto na literatura, ou seja, o branch and bound desenvolvido por [7]. Os resultados favorecem o método aqui proposto. No entanto, podemos notar que nenhum dos dois conseguiu resolver de forma ótima instâncias de tamanhos maiores. Vale destacar que nosso método usa fortemente o CPLEX, enquanto o BB_{max} não possui essa dependência e seu desempenho computacional é mais influenciado pela implementação adotada. Mesmo considerando que uma implementação mais cuidadosa possa reduzir seu tempo computacional, a diferença observada em nossos experimentos parece manter a vantagem para o novo método.

O valor de nosso trabalho reside na inovação dos três elementos propostos, corroborados pelo desenvolvimento teórico apresentado e pelos resultados práticos que se mostraram competitivos na aplicação desses elementos, o que possibilita futuras explorações nessa mesma direção.

Referências

[1] Egon Balas. Extension de l'algorithme additif à la programmation en nombres entiers et à la programmation non linéaire. *Comptes Rendus Hebdomadaires des Seances de L'Academie des Sciences*, 258(21):5136, 1964.

[2] George B Dantzig. Linear programming and extensions, 1963.

[3] Fred Glover. Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22(4):455–460, 1975.

[4] Peter L Hammer and S. Rudeanu. Boolean methods in operations research, 1968.

[5] Te C Hu. Integer programming and network flows. Technical report, DTIC Document, 1969.

[6] Ching-Chung Kuo, Fred Glover, and Krishna S Dhir. Analyzing and modeling the maximum diversity problem by zero-one programming. *Decision Sciences*, 24(6):1171–1185, 1993.

[7] Rafael Martí, Micael Gallego, and Abraham Duarte. A branch and bound algorithm for the maximum diversity problem. *European Journal of Operational Research*, 200(1):36–44, 2010.

[8] Carlos Diego Rodrigues, Dominique Quadri, Philippe Michelon, and Serigne Gueye. 0-1 quadratic knapsack problems: an exact approach based on a t-linearization. *SIAM Journal on Optimization*, 22(4):1449–1468, 2012.

[9] Pablo Soares, Manoel Campêlo, Carlos Diego Rodrigues, and Philippe Michelon. t-linearizalização de funções quadráticas de variáveis binárias. *Anais do XLIX SBPO*, pages 2569–2580, 2017.

[10] Pablo Soares and Manoel Campêlo. Limite superior para o problema da diversidade máxima. In *CSBC 2018 - 3º ETC*, jul 2018.

[11] Yangming Zhou, Jin-Kao Hao, and Béatrice Duval. Opposition-based memetic search for the maximum diversity problem. *IEEE Transactions on Evolutionary Computation*, 21(5):731–745, 2017.

Tabela 1: Resultados com as instâncias *Glover2* obtidos pelos métodos $BBmax$ e $(BB)_t^\pi$

Glover2		$BBmax$			$(BB)_t^\pi$		
(n, m)	MelhorSol	LoWeRB	CPU(s)	#Opt	LoWeRB	CPU(s)	#Opt
(25, 2)	121.249	**121.249**	**0.00**	**Sim**	**121.249**	0.06	**Sim**
	247.396	**247.396**	**0.00**	**Sim**	**247.396**	0.08	**Sim**
	131.361	**131.361**	**0.00**	**Sim**	**131.361**	0.07	**Sim**
	215.298	**215.298**	**0.00**	**Sim**	**215.298**	0.09	**Sim**
	254.721	**254.721**	**0.00**	**Sim**	**254.721**	0.07	**Sim**
(25, 7)	4165.534	**4165.534**	**0.02**	**Sim**	**4165.534**	0.14	**Sim**
	4285.734	**4285.734**	**0.02**	**Sim**	**4285.734**	0.17	**Sim**
	3873.512	**3873.512**	**0.04**	**Sim**	**3873.512**	0.15	**Sim**
	1813.782	**1813.782**	**0.02**	**Sim**	**1813.782**	0.12	**Sim**
	3418.310	**3418.310**	**0.02**	**Sim**	**3418.310**	0.15	**Sim**
(50, 5)	1795.206	**1795.206**	**0.1**	**Sim**	**1795.206**	4.90	**Sim**
	2182.365	**2182.365**	**0.03**	**Sim**	**2182.365**	2.22	**Sim**
	1365.702	**1365.702**	**0.08**	**Sim**	**1365.702**	2.55	**Sim**
	1291.361	**1291.361**	**0.09**	**Sim**	**1291.361**	1.81	**Sim**
	1890.258	**1890.258**	**0.05**	**Sim**	**1890.258**	3.62	**Sim**
(50, 15)	10852.354	**10852.354**	304.85	**Sim**	**10852.354**	**5.10**	**Sim**
	7659.776	**7659.776**	2304.34	**Sim**	**7659.776**	**3.73**	**Sim**
	13862.297	**13862.297**	220.24	**Sim**	**13862.297**	**11.15**	**Sim**
	14406.562	**14406.562**	369.90	**Sim**	**14406.562**	**8.63**	**Sim**
	14352.365	**14352.365**	503.35	**Sim**	**14352.365**	**24.54**	**Sim**
(100, 10)	5402.307	**5402.307**	532.90	**Sim**	**5402.307**	**124.87**	**Sim**
	8068.118	**8068.118**	**151.92**	**Sim**	**8068.118**	580.86	**Sim**
	5543.700	**5543.700**	1487.71	**Sim**	**5543.700**	**305.60**	**Sim**
	9480.842	**9480.842**	469.02	**Sim**	**9480.842**	**992.45**	**Sim**
	7711.022	**7711.022**	246.63	**Sim**	**7711.022**	**515.37**	**Sim**
(100, 30)	47683.494	**47683.494**	3600	Não	**47683.494**	**456.02**	**Sim**
	42690.596	**42690.596**	3600	Não	**42690.596**	**767.46**	**Sim**
	92960.153	**92960.153**	3600	Não	**92960.153**	**251.19**	**Sim**
	81921.723	**81921.723**	3600	Não	**81921.723**	**1363.04**	**Sim**
	53634.549	**53634.549**	3600	Não	**53634.549**	**776.16**	**Sim**
(125, 12)	5393.845	**5393.845**	3600	Não	**5393.845**	**109.49**	**Sim**
	12668.437	12659.353	3600	Não	12665.908	3809.35	Não
	8021.785	**8021.785**	3600	Não	**8021.785**	**1011.08**	**Sim**
	13328.914	**13328.914**	**749.30**	**Sim**	**13328.914**	1687.8	**Sim**
	7125.345	**7125.345**	3600	Não	**7125.345**	**1834.59**	**Sim**
(125, 37)	111485.584	111478.251	3600	Não	**111485.584**	3694.28	Não
	129434.479	**129434.479**	3600	Não	**129434.479**	3620.49	Não
	89474.979	**89474.979**	3600	Não	**89474.979**	3605.08	Não
	97638.684	**97638.684**	3600	Não	**97638.684**	3605.10	Não
	120923.326	**120923.326**	3600	Não	**120923.326**	3642.08	Não
(150, 15)	22674.921	**22674.921**	3600	Não	**22674.921**	3676.78	Não
	14419.886	**14419.886**	3600	Não	**14419.886**	3702.95	Não
	12908.482	12900.75	3600	Não	**12908.883**	3681.58	Não
	16344.653	**16344.653**	3600	Não	**16344.653**	3703.39	Não
	16665.049	**16665.049**	3600	Não	**16665.049**	3704.89	Não
(150, 45)	160263.199	**160263.199**	3600	Não	**160263.199**	3604.44	Não
	198323.556	**198323.556**	3600	Não	**198323.556**	3604.56	Não
	136023.331	136020.509	3600	Não	**136023.331**	3603.04	Não
	199702.546	**199702.546**	3600	Não	**199702.546**	3603.09	Não
	145685.389	**145685.389**	3600	Não	**145685.389**	3607.15	Não