Local branching

Matteo Fishetti e Andrea Lodi. Publicado em *Math. Program., Ser. B*, Vol 98, (2003) pp. 23-47.
por Luciana Brugiolo Gonçalves

1. Introdução

Programação inteira mista (PIM) exerce um papel importante na modelagem de problemas combinatórios NP-difíceis. Para tais problemas, frequentemente a solução exata destes modelos para instâncias de tamanhos relevantes demanda grande tempo computacional, o que torna importante o estudo de métodos heurísticos eficientes.

Mesmo contando como uma gama de sofisticadas ferramentas disponíveis para resolver estes modelos, estas, muitas vezes (quando tratamos de instâncias grandes, ou seja, tamanho relevante na prática), não são capazes de convergir para a soluções ótima num tempo aceitácel. Por isso, torna-se mais atraente utilizar estratégias que encontram melhores soluções em estágios iniciais àquelas que só alcançam boas soluções em passos mais tardios (visto que estes passos têm uma propabilidade menor de serem alcançados quando considerado um tempo limite).

Neste artigo os autores estudaram a utilização de resolvedores de propósito geral como uma ferramenta "caixa-preta"para explorar subespaços definidos e controlados por um outro nível externo de *branching*, denominado nível estratégico. O *framework* é uma técnica para melhorar o desempenho de resolvedores propósito geral de modelos de PIM, de forma a favorecer a obtenção de boas soluções em estágios iniciais.

Para definir os subespaços é utilizada uma idéia inspirada nas busca locais de metaheuríscas. Assim, é preciso determinar a vizinhaça de uma dada solução. De acordo com a proposta, para determinar estas vizinhanças são introduzidas no modelo inequações lineares inválidas, chamados de cortes de *local branching*.

A estratégia proposta encontra a solução exata dos modelos, sendo projetada para aperfeiçoar o comportamento heurístico dos resolvedores. Este nível estratégico de *branching* define a vizinhaça a ser explorada pelos resolvedores de propósito geral (QUE NORMALMENTE TRABALHAM COM A TÉCNICA DE *local-branching*). Desta forma, o framework pode ser definindo como uma estratégia de *branching* de dois níveis.

2. Heurísticas de fixação de variáveis

Assumindo haver um resolvedor exato ou heurístico para PIM, este é aplicado no modelo de forma a abortar sua execução assim que uma solução (possivelmente inviável), $\bar{x}$, é encontrada. Esta solução pode ser definida, por exemplo, como a solução da relaxação do modelo de PIM ou como alguma solução heurística do problema. A solução $\bar{x}$ é então analisada, algumas de suas variáveis são arredondadas para o valor inteiro mais próximo (se a variável for não inteira) e então fixadas neste valor. O método é então reaplicado no problema resultante do modelo acrescido da fixação. O resolvedor é chamado novamente, uma nova solução guia é encontrada, algumas de suas variáveis são fixadas, e assim por diante. Desta forma, a cada fixação o problema é reduzido, possibilitando ao resolvedor trabalhar em problemas cada vez mais restritos, aumentando a chance de se encontrar uma solução e provar a otimalidade.

O ponto crítico deste método de fixação de variáveis está associado à escolha das variáveis a serem fixadas a cada iteração. Para problemas difíceis, boas soluções podem ser encontradas apenas após vários ciclos de fixação.

A questão é como fixar um relevante número de variáveis sem perder a possibilidade de encontrar as boas soluções viáveis. Para ilustrar, supondo uma dada solução heurística 0-1 de um modelo puramente 0-1 com n variáveis. Considerando o sub-problema de fixar em 1 pelo menos 90% das variáveis não zero, uma forma flexível de fazer esta fixação seria inserindo a restrição (*soft*) $\sum_{i=1}^{n} \bar{x}_i \times x_i \geq \lceil 0,9 \times \sum_{i=1}^{n} \bar{x}_i \rceil$ no modelo original e, posteriormente, aplicando o resolvedor. Desta forma, evitar-se-ia uma rígida fixação das variáveis em favor de uma maior flexibilidade, definindo uma vizinhança da solução atual ($\bar{x}$).

3. *Local branching framework*

O mecanismo de fixação flexivel apresentado na seção anterior levou a um framework que pode ser descrito como segue: considerando um modelo (P) genérico onde o conjunto indexado de variáveis $\mathcal{N} = \{1, \ldots, n\}$ é dividido entre os subconjuntos $\mathcal{B}$, $\mathcal{G}$ e $\mathcal{C}$, onde $\mathcal{B} \neq \emptyset$ armazena os índices das variáveis 0-1, enquanto os conjuntos $\mathcal{G}$ e $\mathcal{C}$, que podem ser vazios, armazenam os índices das variáveis inteiras e contínuas, respectivamente, tem-se:

$$(P) \quad min\ c^T x \tag{3.1}$$
$$Ax \geq b \tag{3.2}$$
$$x_j \in \{0,1\} \quad \forall j \in \mathcal{B} \tag{3.3}$$
$$x_j \geq 0, \quad \text{inteiro } \forall j \in \mathcal{G} \tag{3.4}$$
$$x_j \geq 0 \quad \forall j \in \mathcal{C} \tag{3.5}$$

Dada uma solução de referência $\bar{x}$ válida em (P), o conjunto $\bar{S} = \{j \in \mathcal{B} : \bar{x}_j = 1\}$ representa o suporte de $\bar{x}$. Para um dado valor inteiro positivo k, foi definida a vizinhança K-OPT $N(\bar{x}, k)$ de $\bar{x}$ como o conjunto de soluções viáveis de (P) que satistaçam também a restrição de *local branching* (3.6), restrição esta que força o estado das variáveis 0-1 nas soluções x e $\bar{x}$ diferirem em, no máximo, k posições.

$$\Delta(x, \bar{x}) = \sum_{j \in \bar{S}} (1 - x_j) + \sum_{j \in \mathcal{B} \setminus \bar{S}} x_j \leq k \tag{3.6}$$

Como o próprio nome sugere, a restrição de *local branching* pode ser utilizada como critério dentro de um esquema enumerativo de (P). Assim, dada uma solução $\bar{x}$, o espaço de soluções associado ao *branching* corrente pode ser particionado através da disjunção $\Delta(x, \bar{x}) \leq k$

ou $\Delta(x, \bar{x}) \geq k + 1$, ramo esquerdo e direito, respectivamente.

O parâmetro k define o tamanho da vizinhança. Desta forma este deve ser escolhido de maneira que o subespaço associado ao ramo esquerdo deva ser pequeno o suficiente para que a solução ótima seja provada em um tempo computacional razoável, mas também grande o suficiente para, com certa probabilidade, conter soluções melhores que a incumbente.

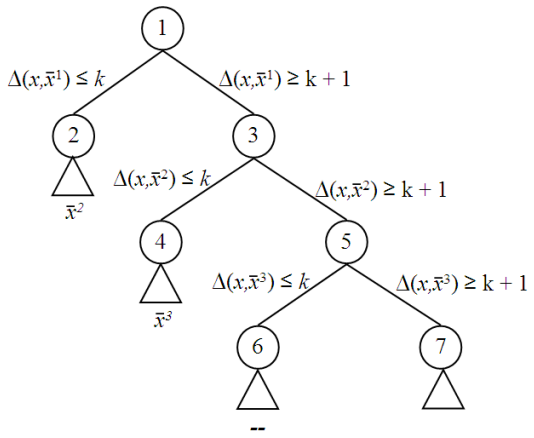

Figura 1: Estrutura básica do método

A idéia básica deste *framework* é ilustrada na Figura 1, onde os triângulos representam as subárvores exploradas pelo resolvedor exatos de modelos de PIM. Assumindo uma solução incumbente inicial $\bar{x}^1$ no nó raíz 1, o ramo esquerdo corresponde a vizinhança $N(\bar{x}^1, k)$ que é explorada pelo resolvedor (nó 2), convergindo para uma solução melhor que a atual, digamos $\bar{x}^2$. Esta nova solução tornar-se-á a nova solução incumbente e o processo é repetido no ramo direito, nó 3. O subespaço associado a este no é fracionado de acordo com a nova solução incumbente. Desta forma, o subespaço $N(\bar{x}^2, k) \backslash N(\bar{x}^1, k)$ será explorado no nó 4, produzindo a nova solução incumbente, $\bar{x}^3$, solução esta que será utilizada para definir o subespaço a ser explorado no ramo esquerdo do nó 5. Ainda na Figura /reffig1, considerando que nenhuma solução melhor seja encontrada no subespaço definido para o nó 6, a restrição $\Delta(x, \bar{x}^3) \geq k + 1$ é inserida no nó 7, definindo o último nó explorado pelo resolvedor.

Pode-se observar que a solução fracionária encontrada pelo nó 1 não é necesariamente podada em ambos os ramos 2 e 3, como ocorre quando se aplica o *branching* padrão de variáveis. O mesmo ocorre em todas as ramificações, ou seja, existe uma diferença entre as técnicas de *branching*, pois não se força o valor de variáveis fracionárias, mas sim, guia-se o resolvedor pelo espaço de soluções. A vantagem esperada é uma atualização mais antecipada (e mais freqüente) da solução incumbente. Em outras palavras, deseja-se encontrar uma boa solução até o momento em que a estratégia de *branching* proposta não possa mais ser aplicada, quando é necessário recorrer ao resolvedor para concluir a enumeração (nó 7 do exemplo).

Para adiar ainda mais o momento de aplicar o resolvedor no nó "final", duas estratégias de aperfeiçoamento foram desenvolvidas.

4. Aperfeiçoamento

- Primeira estratégia: impor um tempo limite para processar o ramo esquerdo.

Esta primeira estratégia se relaciona ao fato de que, em certos casos, dependendo do parâmetro k, o tempo para processar o ramo esquerdo é muito elevado. Assim, é razoável estipular um tempo limite para sua computação. O tempo limite pode ser alcançado em duas diferentes situações:

(a) Uma nova solução incumbente foi encontrada no tempo transcorrido. Neste caso, retorna-se para o nó pai realizando uma nova ramificação utilizando a nova solução, sem alterar o valor de k;

(b) Nenhuma solução melhor foi obtida. Retoma-se o nó pai e realiza-se uma outra fragmentação do espaço de soluções considerando um valor inferior para o parâmetro k reduzindo, desta forma, o espaço de soluções associado ao nó esquerdo.

- Segunda estratégia: diversificação

Esta estratégia é utilizada quando o subespaço de soluções é explorado pelo resolvedor e nenhuma solução melhor que a incumbente é encontrada. Segundo a descrição inicial do *framework*, deveria ser utilizado o resolvedor para explorar o ramo direito. Entretanto, na tentativa de adiar esta etapa, dois mecanismos de diversificação foram propostos *soft* e *strong*.

Soft: Trata de ampliar o espaço de soluções explorado. Para tanto, incrementa-se o valor de k. Com o novo valor de k, o modelo é submetido novamente ao resolvedor por um certo limite de tempo. Caso nenhuma solução melhor seja encontrada no tempo estabelecido, é aplicado o segundo mecanismo de diversificação.

Strong: Inspirada na técnica *Variable Neighborhood Search*, busca-se uma solução (geralmente pior que a solução incumbente) que não seja tão distante da incumbente para ser utilizada como próxima solução de referência. Neste trabalho, isto foi conseguido aumentando o valor de k para $k + 2 \times \lceil k/2 \rceil$ e submetendo o modelo para o resolvedor sem exigir melhoria na melhor solução conhecida. O resolvedor é interrompido logo que uma primeira solução viável é encontrada. Esta solução, normalmente pior que a melhor solução conhecida, é então utilizada como próxima solução de referência para guiar o processo de busca.

Para executar o *framework* proposto, além de informar o valor do parâmetro k e os tempo limites de computação (tempo total e tempo limite para cada execução do resolvedor), é necessário informar também o número máximo de vezes que se permitirá realizar a diversificação. Como regra, a diversificação *strong* somente é executada se a $soft$ foi executada na iteração anterior. A estratégia proposta neste trabalho se comporta como um método exato

quando o tempo total é igual a $+\infty$ e número máximo de diversificações é menor que este mesmo valor. Se considerarmos o número máximo de diversificações igual a $+\infty$, a estratégia pode ser vista como uma busca local heurística.

5. Resultados Computacionais

Para avaliar o método proposto, os autores utilizaram 29 instâncias de diferentes problemas. Para verificar o desempenho do algoritmo, comparou-se o resultado deste com o resolvedor ILOG-Cplex 7.0 usando duas diferentes configurações do parâmetro *emphasis* (*optmality* e *feasibility*). Como resolvedor para o *local branching* foi utilizado o mesmo ILOG-Cplex 7.0 com *emphasis = optmality*. Nos teste realizados utilizou-se um limite de tempo de 5 horas (apenas uma instância, por ser muito grande, teve um limite diferente - 10 horas). Nenhum algoritmo foi capaz de provar a otimalidade neste tempo pré-estabelecido. Nos testes, considerou-se $k = 20$, não limitou-se o número de diversificações e o tempo de cada execução do resolvedor foi determinado de acordo com o tamanho da instância. Analisando os resultados foi possível observar que o *local branching* teve um desempenho melhor na grande maioria das instâncias (23 das 29 instâncias).

Propositalmente não se verificou o ajuste dos parâmetros nos testes iniciais, o que mostra a **robustez** do método. Entretanto, ajustando os parâmetros de acordo com as características da instância observou-se signigicativa melhoria nos resultados (teste realizado para algumas instâncias).

6. Considerações Finais

Para finalizar, os autores apresentaram ainda quatro idéias de modificações que podem sem aplicadas na estrutura básica da estratégia de *local branching* apresentada.

- Maior integração com resolvedor de PIM: uma das formas de explorar as restrições de *local branching* é utilizando-as como uma estratégia de *branching* em um método de solução exata (abordagem esta que foi explorada neste trabalho). Da forma como foi apresentado é fácil de implementar, mas há a desvantagem ao disperdiçar parte do esforço despendido pelo resolvedor, caso que ocorre, por exemplo, quando nenhuma solução melhor é encontrada numa fração de tempo dado para o resolvedor. Consequentemente, é esperado que um *framework* mais integrado e flexível onde os dois níveis de *branching* trabalhem de forma mais cooperativa produza melhores resultados.

- Busca Local para *branch-and-cut* - **Metaheurísticas**: uma outra maneira de utilizar as restrições de *local branching* seria para implementar metaheurísticas como Busca Tabu e VNS. Os principais elementos destas metaheurísticas (definição da vizinhança da solução atual, soluções tabu e movimentos, diversificação, etc) podem ser modeladas em termos de cortes que seriam dinamicamente inseridos e removidos do modelo original, permitindo obter um (poderoso) *framework* Tabu ou VNS para MPIS.

- Trabalhar com soluções inviáveis: neste framework considerou-se que há uma solução de partida que assumiu-se ser fornecida pelo resolvedor. Contudo, para alguns problemas obter uma solução viável inicial pode demandar muito tempo. Nestes casos, deveria-se considerar a possibilidade de trabalhar com soluções inviáveis. Para tanto, variáveis de folga (em algumas restrições) e penalização na função objetivo, a exemplo de algumas metaheurísticas, pode apresentar um resultado interessante.

- Tratar variáveis inteiras quaisquer: o modelo foi baseado na suposição de que o MIP possui um conjunto de variáveis 0-1 que são relevantes na definição da distância entre duas soluções $\Delta(x, \bar{x})$. Contudo, em experimentos realizados pelos autores, esta definição é eficiente nos casos onde o modelo envolve variáveis inteiras quaisquer, onde as variáveis 0-1 são provavelmente as maiores responsáveis pela dificuldade do modelo. Contudo, podem haver casos em que os modelos não envolvem variáveis 0-1, ou que estas não são as mais responsáveis pela complexidade do modelo, de forma que as restrições de *local branching* não ajudam os resolvedores a tratar estes problemas. Nesta situação é interessante alterar o *framework* de modo a considerar variáveis inteiras quaisquer no computo de Δ. Para tanto, supondo uma modelo (P) envolvendo limites $l_j \leq x_j \leq u_j$ para as variáveis interias x_j $(j \in \mathcal{I} = \mathcal{B} \cup \mathcal{G})$. Uma restrição de *local branching* apropriada poderia ser definida como:

$$\begin{aligned} \Delta_1(x, \bar{x}) &= \sum_{j \in \mathcal{I}: \bar{x}_j = l_j} \mu(x_j - l_j) \\ &+ \sum_{j \in \mathcal{I}: \bar{x}_j = u_j} \mu(u_j - x_j) \\ &+ \sum_{j \in \mathcal{I}: l_j < x_j < u_j} \mu(x_j^+ + x_j^-) \leq k \end{aligned}$$

onde pesos μ_j são definidos, por exemplo, como $\mu_j = 1/(u_j - l_j)$ para todo $j \in \mathcal{I}$, enquanto os termos x_j^+ e x_j^- requerem restrições adicionais no modelo da forma abaixo:

$$\begin{aligned} x_j &= \bar{x}_j + x_j^+ - x_j^- \\ x_j^+ &\geq 0 \\ x_j^- &\geq 0 \\ \forall j &\in \mathcal{I} : l_j < x_j < u_j \end{aligned}$$

Neste trabalho os autores propuseram um *framework* de propósito geral para modelos de PIM baseado na exploração de vizinhanças definidas através de inequações lineares (cortes de *local branching*). Vizinhanças estas exploradas com auxílio de um resolvedor de MPI disponível.

Referências

[1] M. Fischetti, A. Lodi, Local Branching, *Mathematical Programming*, **98** (2003), 23-47.