MO420/MC908: Tópicos em Otimização Combinatória
Prof. Cid Carvalho de Souza - IC/UNICAMP
3º Trabalho Prático – 1º semestre de 2009

1 Introdução

Dado um conjunto S de segmentos de reta no plano, define-se o *número de interseção* (em inglês, *stabbing number*) de S, denotado por $\texttt{sn}(S)$, como sendo o **maior** número de segmentos de S que uma reta qualquer r no plano intercepta. Note que a interseção de um segmento s com r pode ocorrer só numa extremidade de s, só em um ponto interno de s ou incluir todo o segmento caso s esteja contido em r.
Por outro lado, dado um conjunto de pontos P no plano, onde $n = |P|$ é par, um *emparelhamento perfeito* é um conjunto M de segmentos com extremos nos pontos de P e tal que cada ponto de P é extremidade de exatamente um destes segmentos.
O problema minimização do número de interseções de um emparelhamento perfeito (STAB) (*minimum stabbing number perfect matching*) é definido da seguinte forma:

> **Instância**: um conjunto de n pontos no plano $P = \{p_1, \ldots, p_n\}$ dados pelas suas coordenadas de modo que $p_i = (x_i, y_i)$ para todo $i = 1, \ldots, n$.
>
> **Problema**: encontrar um emparelhamento perfeito M de P tal que $\texttt{sn}(M)$ seja **mínimo**.

Pela definição do problema, é evidente que n deve ser par para que o problema faça sentido. Na figura 1 vê-se um conjunto P de seis pontos e um emparelhamento perfeito M de P com $\texttt{sn}(M) = 2$.

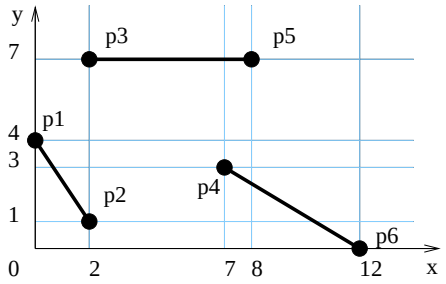

Figura 1: Exemplo de um emparelhamento M com $\texttt{sn}(M) = 2$.

O objetivo deste trabalho é desenvolver um algoritmo *branch-and-cut* para resolver STAB de forma exata usando as bibliotecas do resolvedor comercial de Programação Linear Inteira (PLI) XPRESS.

2 Uma formulação PLI para o STAB

Antes de passarmos à modelagem do problema, vamos enunciar um resultado relevante. Para tal, usa-se N para denotar o conjunto $\{1, \ldots, n\}$ e supõe-se que P seja um conjunto de n pontos do plano como definido anteriormente.

Proposição 1 *Para todo $i < j$ onde $(i, j) \in N \times N$, seja r_{ij} a reta que passa pelos pontos p_i e p_j de P e seja R o conjnto de todas as retas obtidas desta forma. Então, dado um emparelhamento perfeito M de P, o valor de $\texttt{sn}(M)$ sempre será igual ao maior número de interseções entre os segmentos de M e uma das retas de R.*

Ou seja, esta proposição mostra que, para resolver o STAB, podemos nos limitar a contar o número de interseções sobre um número finito de retas (da ordem de n^2). Assim, podemos fazer uma formulação compacta do problema.
Inicialmente, para todo par de pontos (p_i, p_j) de P, $i < j$, define-se a seguinte variável binária:

$$x_{ij} \quad = \quad \begin{cases} 1, & \text{se o segmento } \overline{p_i p_j} \text{ pertence ao emparelhamento } M, \\ 0, & \text{caso contrário.} \end{cases}$$

Define-se ainda uma variável inteira y, cujo significado ficará claro ao analisarmos o modelo PLI abaixo.

$$\min \quad y \tag{1}$$

$$\text{sujeito a} \quad \sum_{e \in \delta(i)} x_e \; = 1, \quad \forall \; i = 1, \ldots, n, \tag{2}$$

$$\sum_{e \in E(S)} x_e \; \leq (|S| - 1)/2, \quad \forall \; S \subset P, |S| \text{ ímpar}, \tag{3}$$

$$\sum_{e \in I(r_{ij})} x_e \; \leq y, \quad \forall \; (i, j) \in N \times N, i < j \tag{4}$$

$$y \text{ inteira e } x \text{ definida como acima.} \tag{5}$$

Nas restrições acima foram usadas as seguintes notações: *(i)* para todo $S \subseteq P$, $E(S)$ denota o conjunto de todos segmentos com ambas extremidades em S e *(ii)* para todo $(i, j) \in N \times N$, com $i < j$, $I(r_{ij})$ é o conjunto de todos os segmentos de $E(P)$ que interceptam a reta r_{ij} definida de acordo com a Proposição 1.
No texto escrito do trabalho, responda às seguintes perguntas relativas a esta formulação:

P0: Interprete o significado da variável y e da função objetivo.

P1: Dê os argumentos que mostram a validade das restrições (3).

P2: Interprete o significado de cada uma das restrições do modelo.

P3: Fixado um conjunto S, mostre que sua restrição correspondente em (3) é **equivalente** à restrição

$$\sum_{e \in \delta(S)} x_e \geq 1, \tag{6}$$

onde $\delta(S)$ é o conjunto de segmentos com uma extremidade em S e a outra em $P \setminus S$.

Como o número de restrições em (3) (ou, equivalentemente, em (6)) é exponencial em n, a única forma de usar esta formulação para resolver exatamente o STAB é através de um algoritmo *branch-and-cut*.
Para isto, note que o problema da separação relativo às restrições (6) pode ser resolvido em tempo polinomial através da resolução de um problema de corte **ímpar** mínimo em grafos (CIM), de modo bastante parecido com o que ocorre com a desigualdade de *eliminação de subciclos* para o *problema do caixeiro viajante*.

3 Implementação

Usando a as bibliotecas providas pelo XPRESS, implemente um algoritmo de *branch-and-cut* para o STAB que tem como formulação inicial àquela da seção 2 **sem nenhuma restrição do tipo (3) (ou equivalente)** e que usa como planos de corte as desigualdades (6).
Eis um resumo dos procedimentos que devem ser implementados para que se tenha o algoritmo *branch-and-cut* pedido: *(i)* uma rotina de separação heurística para as desigualdades (6), *(ii)* uma rotina de separação

exata para as desigualdades (6) e *(iii)* uma heurística primal para o STAB que, a partir da solução da relaxação linear corrente calcula uma solução viável primal do problema, atualizando o limitante primal sempre que uma solução melhor for encontrada.
No caso da rotina de separação, existe um algoritmo polinomial que resolve de modo exato o CIM. Este algoritmo começa computando a árvore de Gomory-Hu e depois usa o resultado a seguir.

Teorema 1 *Seja $G = (V, E)$ um grafo com $|V|$ par, seja $c \in \mathbb{R}_+^{|E|}$ uma função de custos sobre as arestas de G e seja $T = (V, F)$ uma árvore de Gomory-Hu para G e c. Então, um dos cortes fundamentais determinados por T é um corte ímpar mínimo em G.*

Algumas observações sobre este teorema: *(a)* corte está sendo usado aqui com sendo um conjunto de vértices; *(b)* um corte fundamental numa árvore T é uma das componentes conexas obtidas de T ao remover uma de suas arestas; *(c)* no STAB podemos associar os pontos de P aos vértices de um grafo completo G e *(d)* no caso do problema de separação associado ao modelo PLI da seção 2, o custo da aresta (i, j) ligando os vértices associados aos pontos p_i e p_j de P é dado por x^*_{ij} que é o valor da variável fracionária correspondente ao segmento $\overline{p_i P_j}$.
No texto do seu trabalho, você deve responder à seguinte pergunta:

P4: O que é uma árvore de Gomory-Hu para um grafo $G = (V, E)$ e uma função de custo (positiva) c associada às suas arestas ?
Nota: dê uma explicação sucinta, com pelo menos um exemplo e não deixe de citar a fonte de consulta que foi usada.

Em breve, será disponibilizado na página da disciplina um código em C que computa uma árvore de Gomory-Hu. Você pode usá-lo na sua implementação e, então, completar o código da rotina de separação exata aplicando o teorema 1.
Em resumo, no que diz respeito à implementação, o que você tem que fazer é desenvolver heurísticas que tentem resolver o problema da separação mais rapidamente, além de implementar um algoritmo exato para que seja executado sempre que a sua heurística falhar.

4 Testes

As instâncias finais de teste serão disponibilizadas **até o dia 19/06/2009** na página da disciplina. O formato dos arquivos de entrada é explicado abaixo para a instância de exemplo da figura 1. O arquivo é dividido em seções. Na seção `DIMENSION` temos o número de pontos (n) e na seção `NODE_COORD_SECTION` temos, para cada linha, o rótulo do ponto e as suas coordenadas (inteiros). Finalmente, na seção `INSTERSECTIONS_SECTION` temos $n(n-1)/2$ linhas e, para cada linha, temos os rótulos (inteiros) i e j de dois pontos de P, $i < j$, um inteiro k denotando o número de interseções da reta r_{ij} com os segmentos de $E(P)$ e os k pares de inteiros indicando os rótulos dos pontos de P que são extremidades destes segmentos.

```
NAME : Exemplo
COMMENT : Exemplo MO420
TYPE : TSP
DIMENSION: 6
EDGE_WEIGHT_TYPE : EUC_2D
NODE_COORD_SECTION
1 0 4
2 2 1
3 2 7
4 7 3
5 8 7
6 12 0
INTERSECTIONS_SECTION
1 2 9 1 2 1 3 1 4 1 5 1 6 2 3 2 4 2 5 2 6
1 3 9 1 2 1 3 1 4 1 5 1 6 2 3 3 4 3 5 3 6
1 4 11 1 2 1 3 1 4 1 5 1 6 2 3 2 4 2 5 3 4 4 5 4 6
```

```
1 5 12 1 2 1 3 1 4 1 5 1 6 2 3 2 5 3 4 3 5 3 6 4 5 5 6
1 6 12 1 2 1 3 1 4 1 5 1 6 2 3 2 4 2 5 2 6 3 6 4 6 5 6
2 3 12 1 2 1 3 1 4 1 5 1 6 2 3 2 4 2 5 2 6 3 4 3 5 3 6
2 4 10 1 2 1 4 1 6 2 3 2 4 2 5 2 6 3 4 4 5 4 6
2 5 13 1 2 1 4 1 5 1 6 2 3 2 4 2 5 2 6 3 4 3 5 3 6 4 5 5 6
2 6 9 1 2 1 6 2 3 2 4 2 5 2 6 3 6 4 6 5 6
3 4 11 1 3 1 4 1 5 2 3 2 4 2 5 3 4 3 5 3 6 4 5 4 6
3 5 9 1 3 1 5 2 3 2 5 3 4 3 5 3 6 4 5 5 6
3 6 12 1 3 1 5 1 6 2 3 2 5 2 6 3 4 3 5 3 6 4 5 4 6 5 6
4 5 10 1 4 1 5 2 4 2 5 3 4 3 5 3 6 4 5 4 6 5 6
4 6 9 1 4 1 6 2 4 2 6 3 4 3 6 4 5 4 6 5 6
5 6 9 1 5 1 6 2 5 2 6 3 5 3 6 4 5 4 6 5 6
EOF
```

Você já pode ir codificando a leitura de dados e testando o seu código com este exemplo antes de serem divulgadas as intâncias finais para teste. O arquivo pode ser baixado da página da disciplina.[1]
Faça **pelo menos** (!) os experimentos listados a seguir para todas as intâncias de teste disponibilizadas.

1. execute um algoritmo *branch-and-bound* puro usando o XPRESS e o modelo sem as restrições (3) ou suas equivalentes (6);

2. execute o *algoritmo de planos de corte* (sem enumeração ainda !) baseado na família de desigualdades (6) limitando o número total de iterações do algoritmo a 5 vezes o número pontos do conjunto P passado na entrada. Verifique como progride o limitante dual ao longo das iterações.

   Nestes testes resolva o problema de separação com a heurística que você desenvolveu para este fim, a qual deverá ser seguida do algoritmo exato para resolução do CIM sempre que falhar.

3. repita o experimento anterior sem usar a heurística para resolver o CIM, ou seja, use apenas o algoritmo exato.

4. execute um algoritmo *branch-and-cut* usando a sua rotina de separação e a sua heurística primal. Neste algoritmo você pode experimentar variar parâmetros como, por exemplo: o número de iterações do método de planos de corte por nó da árvore de enumeração (*branch-and-bound*), o número de cortes acrescentados a cada rodada do algoritmo de planos de corte em um nó, os nós nos quais serão gerados cortes, etc.

5 Forma de entrega do trabalho

O texto do trabalho não deve ultrapassar 15 páginas. Nele você deve reportar os resultados obtidos pelo algoritmo, fazendo uma análise comparativa das diferentes versões algorítmicas testadas. Isso inclui comparar os resultados: *(i)* do algoritmo de planos de corte quando se usa ou deixa de usar a heurística para resolver o problema da separação, CIM (estamos especialmente interessados no tempo de computação e na qualidade do limitante dual); *(ii)* dos algoritmos de *branch-and-bound* e *branch-and-cut* (estamos especialmente interessados no tempo de computação, nos limitantes duais e no número de nós explorados na enumeração) e *(iii)* do algoritmo *branch-and-cut* com e sem a heurística primal que você desenvolveu (estamos especialmente interessados no tempo de computação, nos limitantes duais e primais, no número total de nós explorados na enumeração e no número de nós explorados e tempo até encontrar a melhor solução inteira).
Sempre que possível, gráficos e tabelas devem ser usados para descrever os seus resultados e justificar suas afirmações.
O trabalho deve ser entregue por `email` no seguinte formato: arquivo `tgz` que ao ser aberto deverá criar (no diretório corrente) um subdiretório RAXXXXXX(-RAYYYYYY), onde XXXXXX (YYYYYY quando for o caso) é o número do RA do primeiro (segundo) integrante do grupo. Este diretório deverá conter dois subdiretórios:

[1] Observe que, a rigor, o conteúdo da seção `NODE_COORD_SECTION` não é necessário pois na seção `INTERSECTIONS_SECTION` já está resolvido o problema de calcular as interseções entre retas e segmentos.

1. TEXTO, contendo um arquivo `raxxxxxx(-rayyyyyy)-texto.pdf` (nenhum outro formato será aceito) com o relatório respondendo as questões da seção 2, reportando os resultados que você obteve nos seus testes e as suas análises sobre os mesmos, e

2. CODIGO, contendo os programas fonte que você implementou e um arquivo `makefile` que permita sua compilação através do comando `make` (a ausência deste arquivo será penalizada **com rigor**).

   O arquivo executável gerado pelo `makefile` deverá **obrigatoriamente** se chamar `bnc`. Para executar o programa, a seguinte linha de comando deverá ser dada:

   `bnc <tipo-exec> <time-limit> <heur-primal> <separa-heur> <arq-input>`,

onde

- `tipo-exec` é o tipo de execução dado por um caracter x tal que: b equivale ao algoritmo *branch-and-bound* puro do XPRESS, r equivale a rodar um algoritmo de planos de corte puro e simples (sem a enumeração) e f é o *branch-and-cut* completo.

- `time-limit` é o tempo máximo de execução do algoritmo.
  Nota importante: em todos os seus testes nas máquinas do IC este parâmetro nunca deverá exceder 30 minutos.

- `heur-primal` é um valor binário que vale 1 se a heurística primal for usada e 0 caso contrário.
  Nota: só terá efeito se rodar o *branch-and-cut* (i.e, quando `tipo-exec`$= f$).

- `separa-heur` é um valor binário que vale 1 se a heurística que você desenvolveu for usada para resolver o problema da separação (CIM) e 0 caso contrário.
  Nota: só **não** terá efeito se rodar o *branch-and-bound* (i.e, quando `tipo-exec`$= b$).

- `arq-input` é o nome do arquivo de entrada.

Considerando o arquivo de entrada correspondente à instância `arq-input` com n pontos, o seu programa deverá gravar na saída dois arquivos: `<arq-input>.sol` e `<arq-input>.est`. Estes arquivos devem ter o seguinte formato:

- `<arq-input>.sol`: $n/2$ linhas contendo pares de inteiros i e j sendo $i < j$ e tais que os segmentos $\overline{p_i p_j}$ estejam no emparelhamento ótimo M encontrado pelo algoritmo e uma linha contendo um inteiro y correspondente ao `sn`(M), ou seja, o valor ótimo. Caso o ótimo não tenha sido encontrado, os mesmos valores devem ser colocados no arquivo, porém, referentes a melhor solução primal encontrada.

- `<arq-input>.est`: este arquivo deve ter 9 linhas contendo as seguintes informações **nesta ordem**: tipo de execução (caracter b, r ou f conforme descrito acima), total de cortes do tipo (6) inseridos que foram separados pela heurística (inteiro), total de cortes do tipo (6) inseridos que foram separados de forma exata (inteiro), valor da FO da primeira relaxação (double), valor da FO no nó raiz (double), total de nós explorados (inteiro), nó da melhor solução inteira (inteiro), valor da melhor solução inteira (inteira), melhor limitante dual (double).

Importante: os arquivos de saída devem conter **apenas** os valores pedidos e nada mais !
Em particular, não coloque textos explicativos sobre os valores impressos (exemplo do que não fazer: imprimir algo como "`Valor da FO=4.000000`" ao invés de simplesmente imprimir "`4.000000`").

6 Bibliografia auxiliar

Três artigos estão disponibilizados no arquivo `zip` (senha dada em aula) que ajudam a entender o problema e a fazer a implementação pedida. Use-os para consulta e leitura. São eles:

1. *Minimizing the Stabbing Number of Matchings*, S. P. Fekete, M. E. Lübbecke, H. Meijer, Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, pp 437–446, 2004.
   Comentários: o artigo disponibilizado aqui é um relatório técnico que corresponde a uma versão anterior àquela publicada no SODA. Ele contém uma descrição do problema e do uso do modelo PLI para resolver o STAB.

2. *Solving matching problems with linear programming*, M. Grötschel e O. Holland, Mathematical Programming, volume 33, pp 243–259, 1985.
   Comentários: fala sobre resolução do problema de emparelhamentos em grafos usando planos de corte baseados nas desigualdades (6). Contém heurísticas de sepração.

3. *Minimizing the Stabbing Number of Matchings*, S. P. Fekete, M. E. Lübbecke, H. Meijer, Discrete & Computational Geometry, 40, 4, pp 595–621, 2008.
   Comentários: versão completa do artigo do SODA publicada no periódico *Discrete & Computational Geometry*.

7 Considerações finais

Algumas considerações importantes sobre este trabalho:

- Os trabalhos deverão ser feitos mantendo-se os mesmos grupos do segundo trabalho prático.
- Não serão aceitos trabalhos entregues fora do prazo.
- Os programas devem atender às especificações contidas neste documento. Qualquer desvio em relação às mesmas acarretará em descontos na nota final e poderão, inclusive, resultar em nota ZERO em casos mais graves (p.ex., programas que não compilam ou que geram saídas em formatos incompatíveis com aqueles especificados na seção 5).
- O texto do trabalho deve ter uma descrição em alto nível da heurística primal e da heurística usada para resolver o CIM na sepração dos cortes (6). Em ambos os casos, você deve fazer uma análise de complexidade de tempo dos algoritmos.
- Rotinas auxiliares poderão ser baixadas da rede e usadas em seu programa desde que o texto do trabalho mencione explicitamente o uso deste código e cite o endereço completo do *site* de onde ele foi baixado.
- A nota final do trabalho terá uma componente comparativa, ou seja, será considerado qual grupo fez mais coisas e melhor.