Árvores de Steiner

Felipe Schmidt, Gabriel Veras, João Gross

O Problema

Dado:

- Grafo não direcionado G=(V,E)
- Custos das arestas C_E
- Set de terminais T contido em V

Conectar todos nós de T, minimizando custo das arestas na solução.

Formulação Matemática

Com G=(V,E), criamos um grafo bidirecionado G=(V,A).

Minimize

$$\sum_{e \in E} c_e x_e$$

subject to:

$$\sum_{i \in V} f_{ij}^k - \sum_{i \in V} f_{ji}^k = \begin{cases} -1 & \text{if } j = 1 \\ 1 & \text{if } j = k \\ 0 & \text{otherwise for } j \in V \setminus \{1\} \end{cases}$$

for every 'k' in T - {1}

$f_{ij}^k \leq x_e$ for every edge e = (i,j), commodity k

$f_{ij}^k \geq 0$ for every i,j in A, k in T

x_e integer

Heurística - Busca Tabu

- Procedimento adaptativo, de busca local
  - Permite piora se não há melhora (sair de mínimos locais)
  - Noção de vizinhança

- A cada iteração:
  - Solução atual *s* muda para vizinha *s'*
  - *s'* difere de *s* por uma modificação

- Dado solução inicial *s*:
  - Explora vizinhança *N(s)*
  - Função de avaliação *f(s)* escolhe melhor solução de *N(s)*
    - *s'* vira nova solução atual, mesmo que *f(s')* > *f(s)* em um problema de minimização

Heurística - Solução Inicial

- Método iterativo de inserção e remoção de arestas
  - Testa se há ciclo após aresta inserida
    - Se sim, aresta removida
    - Se não, insere nova aresta
  - Continua até que todos nodos terminais estejam na solução

Heurística - Vizinhança

- Começa a partir de uma solução inicial

- Função iterativa consulta arestas da tabela de adjacência
  - Aresta não pertencente à solução inicial: inserida
  - Aresta pertencente à solução inicial: removida
  - Nova solução possível é inserida na vizinhança se:
    - Sem loops
    - Conexa
    - Conectar todos nós terminais

- Processo continua até que todas as arestas da lista de adjacência tenham sido testadas na solução inicial

Heurística - Lista Tabu

- Lista Tabu: lista de soluções diferentes encontradas
  - Usada para determinar se uma nova solução deve ser ou não explorada
  - Tempo de vida da solução na lista:
    - No mínimo o tamanho da lista *em iterações*
    - e.g: |Lista Tabu| = 10. Cada elemento fica na lista por pelo menos 10 iterações.

Heurística - Algoritmo

- Busca da melhor solução da vizinhança, fora da Lista Tabu

- Se achar solução melhor que atual
  - Quantidade de iterações é zerada
  - Valor ótimo global é atualizado na Lista Tabu
  - Recomeço das iterações

- Se não achar solução vizinha menor que a inicial
  - Procurada a menor solução vizinha
    - sem se preocupar com a solução inicial.
  - Se encontrar uma solução vizinha menor que não pertença à lista tabu
    - ela será a solução inicial da próxima iteração
  - Iterações seguem incrementando

Heurística - Critério de Parada

- Após N iterações sem atualização da solução ótima encontrada atingimos o **ponto de parada**.

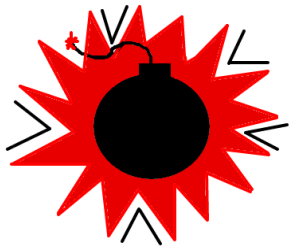
- No nosso algoritmo, impomos limite N = 100.

Algoritmo - Busca Tabu Recursiva

- Primeira implementação realizada
  - Produziu resultados satisfatórios

- Porém, …
  - Muitos laços empilhados
  - Consumo excessivo de recursos de memória

- Das 10 instâncias apenas 4 chegaram no ponto de paradas
  - As demais geraram estouro de memória



Algoritmo - Busca Tabu Iterativa

- Segunda implementação realizada
- Resultados melhores
  - Das 10 instâncias, 7 chegaram no ponto de parada do algoritmo
- Não houve mais estouro de memória, mas…
  - Convergência muito lenta em alguns casos
  - As 3 instâncias que não atingiram o ponto de parada ficaram rodando indefinidamente
- Gargalo observado:
  - Muito tempo de processamento para geração da vizinhança

Algoritmo - Busca Tabu Iterativa Otimizada

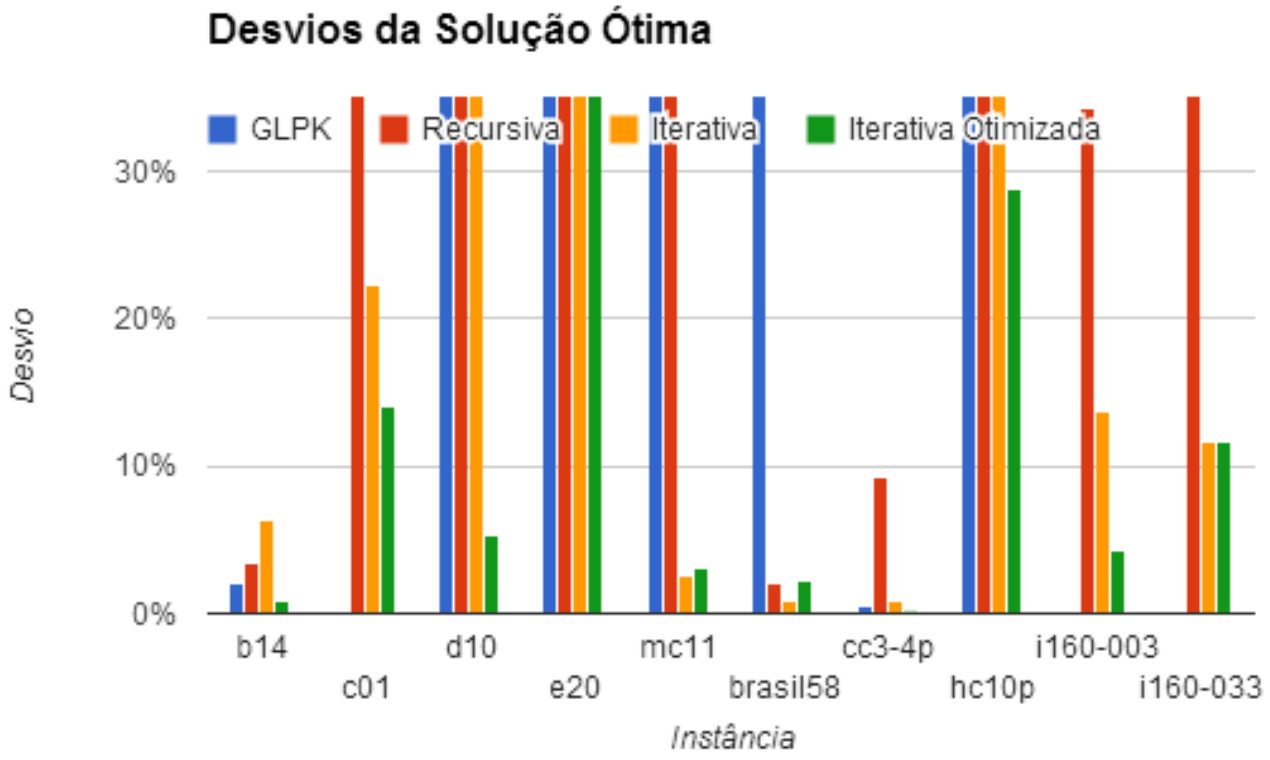
- Terceira implementação realizada
  - Variação otimizada da segunda implementação
  - 90% das instâncias chegaram ao ponto de parada
  - Solução encontrada muito mais rapidamente
  - Sem perda em qualidade dos resultados

- Geração da vizinhança:
  - Imposto um limite de variações na solução inicial para gerar candidatos a soluções vizinhas
  - Fator determina a proporção do limite
    - e.g: se há 1000 variações possíveis na solução inicial, mas o fator for ¼, apenas 250 variações serão realizadas
    - Assim, são gerados menos candidatos à vizinhança e consequentemente menos vizinhos
    - Variações são aleatórias

Ótima vs. GLPK vs. Heurística

Instância	Valor Ótimo	GLPK	Recursiva	Iterativa	Iterativa Otimizada
b14	235	240	243	250	**237**
c01	85	**85**	estouro	104	97
d10	2110	estouro	estouro	running	**2223**
e20	1342	running	estouro	running	running
mc11	11689	running	estouro	**11994**	12052
brasil58	13655	running	13943	**13781**	13950
cc3-4p	2338	2350	2553	2360	**2342**
hc10p	60679	running	estouro	running	**78160**
i160-003	2297	**2297**	3085	2612	2396
i160-033	2101	**2101**	estouro	2345	2345

Comparando Desvios



Solução Ótima vs. GLPK

Instância	Valor Ótimo	Sol. Encontrada	Tempo Exec (s)	Desvio %
b14	235	240	3600	-2.128
c01	85	85	36	0.000
d10	2110	estouro	*	*
e20	1342	running	*	*
mc11	11689	running	*	*
brasil58	13655	running	*	*
cc3-4p	2338	2350	3600	-0.513
hc10p	60679	running	*	*
i160-003	2297	2297	53	0.000
i160-033	2101	2101	1775	0.000

Ótima vs. Busca Tabu Iterativa Otimizada

Instância	Valor Ótimo	Sol. Inicial	Melhor Sol. Encontrada	Tempo Exec (s)	Desvio %	Redução de Vizinhança
b14	235	511	237	3.6	-0.851	1/4
c01	85	2840	97	637.7	-14.118	1/4
d10	2110	5512	2223	31746	-5.355	1/8
e20	1342	13611	running	*	*	1/16
mc11	11689	33696	12052	1872.6	-3.105	1/4
brasil58	13655	135559	13950	22.8	-2.160	1/4
cc3-4p	2338	7801	2342	2	-0.171	1/4
hc10p	60679	107323	78160	32410	-28.809	1/16
i160-003	2297	17273	2396	9.6	-4.310	1/4
i160-033	2101	15240	2345	10.5	-11.614	1/4

Referências

[1] - Sunil Chopra and Chih-Yang Tsai, “Polyhedral Approaches for the Steiner Tree Problem on Graphs”, *Kluwer Academic Publisher*

RESULTADOS COMPLEMENTARES

Solução Ótima vs. Busca Tabu Recursiva

Instância	Valor Ótimo	Sol. Encontrada	Tempo Exec (s)	Desvio %
b14	235	243	21	-3.404
c01	85	estouro	*	*
d10	2110	estouro	*	*
e20	1342	estouro	*	*
mc11	11689	estouro	*	*
brasil58	13655	13943	203	-2.109
cc3-4p	2338	2553	54	-9.196
hc10p	60679	estouro	*	*
i160-003	2297	3085	192	-34.306
i160-033	2101	estouro	*	*

Solução Ótima vs. Busca Tabu Iterativa

Instância	Valor Ótimo	Sol. Encontrada	Tempo Exec (s)	Desvio %
b14	235	250	17	-6.383
c01	85	104	11325	-22.353
d10	2110	running	running	*
e20	1342	running	running	*
mc11	11689	11994	16783	-2.609
brasil58	13655	13781	110	-0.923
cc3-4p	2338	2360	17	-0.941
hc10p	60679	running	running	*
i160-003	2297	2612	131	-13.714
i160-033	2101	2345	141	-11.614

Ótima vs. Busca Tabu Iterativa Otimizada (Completa)

Instância	Valor Ótimo	Sol. Inicial	Sol. Encontrada (mín)	Sol. Encontrada (máx)	Sol. Encontrada (média)	Tempo Exec (s)	Desvio %	Redução de Vizinhança
b14	235	511	237	254	244.3	3.6	-0.851	1/4
c01	85	2840	97	110	100.7	637.7	-14.118	1/4
d10	2110	5512	2223	2223	2223	31746	-5.355	1/8
e20	1342	13611	running	running	running	*	*	1/16
mc11	11689	33696	12052	12259	12202.6	1872.6	-3.105	1/4
brasil58	13655	135559	13950	14856	14378	22.8	-2.160	1/4
cc3-4p	2338	7801	2342	2446	2374.9	2	-0.171	1/4
hc10p	60679	107323	78160	78160	78160	32410	-28.809	1/16
i160-003	2297	17273	2396	2815	2595.9	9.6	-4.310	1/4
i160-033	2101	15240	2345	2871	2558.1	10.5	-11.614	1/4

***10 execuções por instância, exceto d10 e hc10p, que rodaram apenas uma vez.**