

Laboratório 5 – s-t Caminhos com Coleta de Prêmios

Erik Perillo, RA135582

2 de julho de 2017

1 Problema

O problema consiste em, dado um grafo orientado $G = (N, A)$, encontrar o melhor caminho $P = (N_P, A_P)$ de s a t que maximize uma certa função de custo.

2 Metaheurística

De modo a obter um bom *lower bound* para o problema, foi desenvolvida uma metaheurística baseada em *GRASP - Greedy Randomized Adaptive Search Procedure*. Os pseudo-códigos são descritos a seguir.

Algorithm 1

```

1: procedure GRASP-HEURISTICA( $G, s, t, \text{prize}, \text{cost}$ )
2:    $S \leftarrow \emptyset$ 
3:    $S_+ \leftarrow \emptyset$ 
4:   while not timeout and not limit_its and not stuck_in_local_opt do
5:      $S \leftarrow \text{RANDOM-GREEDY-SOL-DFS}(G, s, t, \text{prize}, \text{cost})$ 
6:      $S \leftarrow \text{LOCAL-SEARCH}(S)$ 
7:     if  $\text{cost}(S) \geq \text{cost}(S_+)$  then
8:        $S_+ \leftarrow S$ 
9:   return  $S_+$ 
10:

```

Algorithm 2

```

1: procedure RANDOM-GREEDY-SOL-DFS( $S, s, t, \text{prize}, \text{cost}$ )
2:   set  $s$  as visited
3:   if  $s = t$  then
4:     return TRUE
5:    $A \leftarrow \text{Adj}(s)$ 
6:   while  $A \neq \emptyset$  do
7:     select  $(s, v)$  with probability proportional to  $\text{prize}(s) - \text{cost}((s, v))$ 
8:      $A \leftarrow A - \{(s, v)\}$ 
9:     if not visited( $v$ ) and RANDOM-GREEDY-SOL-DFS( $S, v, t, \text{prize}, \text{cost}$ ) then
10:       $S \leftarrow S \cup (s, v)$ 
11:     return TRUE
12: return FALSE
=0

```

Algorithm 3

```

1: procedure LOCAL-SEARCH( $S, \text{prize}, \text{cost}$ )
2:   while not timeout and not limit_its and not stuck_in_local_opt do
3:     for  $(u, v) \in S$  do
4:       if exists  $(u, w)$  and  $(w, v)$  with  $w \notin S$  then
5:         if  $\text{prize}(w) - \text{cost}((u, w), (w, v)) \geq -\text{cost}(u, v)$  then
6:            $S \leftarrow S \cup \{(u, w), (w, v)\} - \{(u, v)\}$ 
return  $S$ 

```

O algoritmo 1 é a metaheurística principal: enquanto o critério de parada não é verdade, ele busca uma solução gulosa-probabilística inicial e a refina com busca local. O algoritmo 2 explica a ideia da busca probabilística gulosa: Faz-se o caminho em DFS de s a t , escolhendo nós/arcos com probabilidades proporcionais a seus custos: quanto maior, maior a chance de escolher. O algoritmo 3 explica a ideia da busca local: para cada arco da solução aleatória-gulosa, ele procura outros dois arcos que formam um caminho entre os nós do arco original mas com custo melhor. Ele repete isso no máximo 3 vezes. A heurística obteve resultados sempre muito próximos do *upper bound*.

3 Solução exata com Programação Linear Inteira

Seja x_v a variável binária que é 1 se o nó v está na solução e 0 caso contrário. Seja x_a a variável binária que é 1 se o arco a está na solução e 0 caso contrário. Seja π_v os prêmios de cada nó v e c_a os custos de cada arco a .

Queremos encontrar o caminho de s a t $P = (N_P, A_P)$ que maximize:

$$\max z = \sum_{v \in N} x_v \pi_v - \sum_{a \in A} x_a c_a$$

Sujeito a:

- Do nó s , só tem um arco de saída e nenhum de entrada:

$$\sum_{x_a \in \delta_+(s)} x_a = 1$$

$$\sum_{x_a \in \delta_-(s)} x_a = 0$$

- Do nó t , só tem um arco de entrada e nenhum de saída:

$$\sum_{x_a \in \delta_+(t)} x_a = 0$$

$$\sum_{x_a \in \delta_-(t)} x_a = 1$$

- Um nó que não seja s e t é selecionado se e somente se ele tem exatamente um arco de entrada e um de saída:

$$\sum_{x_a \in \delta_+(v)} x_a = x_v, \forall v \in N - \{s, t\}$$

$$\sum_{x_a \in \delta_-(v)} x_a = x_v, \forall v \in N - \{s, t\}$$

- Restrição de integralidade:

$$x_a \in \{0, 1\}, \forall x_a \in A$$

$$x_v \in \{0, 1\}, \forall x_v \in N$$

4 Implementação

A implementação foi feita com o uso do gurobi (versão 7.0.2) e lemon.

5 Avaliação do modelo proposto

Na tabela a seguir, temos os valores para alguns arquivos de entrada. O algoritmo exato recebia o *lower bound* obtido pela heurística como *cutoff*. A heurística mostrou-se muito eficiente, obtendo *lower bounds* muito próximos do valor ótimo. O valor ótimo foi obtido em quase todos os casos. No caso em que o OPT não foi obtido, a heurística obteve um *lower bound* muito próximo do valor ótimo (3.00684e+09). O arquivo 100000_700000.in não pôde ser executado na máquina dos testes pois ela não tinha memória suficiente.

Tabela 1: Resultados de execução com lower bounds (LB) e upper bounds (UB) obtidos.

entrada	LB	UB	OPT obtido?	tempo limite (s)
3_2.in	6	13	sim	1
30_600_1.in	9.33478×10^6	9.33510×10^6	sim	1
100_5000_1.in	2.86350×10^7	2.86361×10^7	sim	1
1000_20000_1.in	3.07347×10^8	3.07367×10^8	sim	1
10000_9000000_1.in	3.00666×10^9	3.00666×10^9	não	60