Lista 2

Victor Sena Molero - 8941317

30 de agosto de 2016

1 Exercícios

Ex 3. *Construa instâncias do* MinCC *com custos unitários, ou seja, instâncias* $(E, \mathcal{S}, c)$ *com* $c_S = 1$ *para todo* S *em* $\mathcal{S}$*, para as quais o custo da cobertura produzida pelo algoritmo* MinCC-Chvátal *pode chegar arbitrariamente perto de* $H_n \mathrm{opt}(E, \mathcal{S}, c)$*, onde* $n := |E|$*.*

Resposta. Seja n um inteiro positivo e $m = n^2$. Vamos construir uma instância $I = (E, \mathcal{S}, c)$ com $|E| = m$ para a qual a resposta do algoritmo MinCC-Chvátal se aproxima de $H_m \mathrm{opt}(I)$. Podemos indexar os elementos de E em uma matriz $n \times n$, ou seja, identificar cada um dos elementos de E por um par (i, j) e denotar o elemento em questão por $E_{i,j}$.

Precisamos agora descrever os conjuntos contidos em $\mathcal{S}$. Teremos n conjuntos que contém, cada um, uma coluna distinta da matriz, ou seja, para todo $i \in [1, n]$ existe exatamente um $S_i^* \in \mathcal{S}$ tal que $S_i = \{E_{j,i} \mid j \in [1, n]\}$. Denotaremos o conjunto de todos os S_i^* por $\mathcal{S}^*$.

Além disso, teremos vários outros conjuntos em $\mathcal{S}$ que particionam cada uma das linhas da matriz E separadamente. Cada linha será particionada em 1 ou mais conjuntos de mesmo tamanho. Mais especificamente, a i-ésima linha será dividida em $\lceil \frac{n}{i} \rceil$ conjuntos de tamanho $n/\lceil \frac{n}{i} \rceil$ cada. O conjunto destes conjuntos vai ser chamado $\bar{\mathcal{S}}$.

Se o algoritmo MinCC-Chvátal sempre der prioridade para os elementos de $\bar{\mathcal{S}}$ quando os custos deles empatarem com os de $\mathcal{S}^*$, vai selecionar todos os elementos de $\bar{\mathcal{S}}$ e nenhum do outro conjunto, enquanto a solução ótima era exatamente oposta (selecionar todo $\mathcal{S}^*$ e nada mais). Portanto, a razão entre a solução encontrada pelo algoritmo é $|\bar{\mathcal{S}}|/|\mathcal{S}^*|$. Sabemos que $|\mathcal{S}^*| = n$, basta calcular $|\bar{\mathcal{S}}|$.

Pela descrição de $|\bar{\mathcal{S}}|$ sabemos quantos elementos existem em cada linha, então, podemos escrever

$$|\bar{\mathcal{S}}| = \sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \geq \sum_{i=1}^{n} \frac{n}{i} = n * H_n$$

, portanto,

$$|\bar{\mathcal{S}}|/|\mathcal{S}^*| \geq H_n$$

, ou seja, para instâncias geradas pela maneira descrita, a solução gerada pelo algoritmo é pelo menos $H_n \mathrm{opt}(I)$, com o crescimento de n, H_n se aproxima de $H_{n^2} = H_m$. Não ficou claro aqui se H_n chega arbitrariamente próximo de H_m, como pedido no enunciado. $\square$

Ex 4. *Lembre-se que* $\ln x$ *é a primitiva da função* $\frac{1}{x}$. *Usando esse fato, deduza que* $H_n \leq 1 + \ln n$. *Conclua que o algoritmo* MinCC-Chvátal *é uma* $O(\log n)$*-aproximação polinomial para o* MinCC.

Resposta. Se $\ln n$ é primitiva de $\frac{1}{n}$, pela Soma de Riemann, para qualquer m inteiro positivo e partição $x_0 < x_1 < \cdots < x_m$ do intervalo $[1, n]$ temos que existe uma sequência c onde $c_i \in [x_{i-1}, x_i]$ para todo $i \leq n$ inteiro positivo tal que $\ln n = \sum_{i=1}^{m} \frac{1}{c_i}(x_i - x_{i-1})$.

Podemos escolher $m = n-1$ e tal partição como sendo $x_0 = 1 < x_1 = 2 < \cdots < x_{n-1} = n$ e escrever $\ln n = \sum_{i=1}^{n-1} \frac{1}{\bar{c}_i}(i+1-i)$ para alguma sequência $\bar{c}$. E, com isso, temos

$$\ln n \geq \sum_{i=1}^{n-1} \frac{i}{i+1} = H_n - 1$$

, já que $\bar{c}_i \leq i+1$ para todo i, portanto

$$H_n \leq \ln n + 1$$

.

Com isso, concluimos que $H_n = O(\lg n)$, portanto, MinCC-Chvátal é uma $O(\log n)$-aproximação polinomial para o MinCC. $\square$

2 Exercícios

Ex 5. *Construa uma família de instâncias* (G, c) *do* TSPM *para as quais o custo do circuito hamltoniano obtido pelo algoritmo* TSPM-RSL *pode ser arbitrariamente próximo de* $2opt(G, c)$*. Construa uma família de instâncias* (G, c) *do* TSPM *para as quais o custo do circuito hamiltoniano obtido pelo algoritmo* TSPM-Christofides *pode ser arbitrariamente próximo de* $\frac{3}{2}opt(G, c)$*.*

Resposta. Queremos construir uma família de instâncias (G, c) para o qual TSPM-RSL encontra uma solução arbitrariamente próxima de $2\text{opt}(G, c)$. Para isso, devemos definir o grafo e os custos. Vamos construir uma família de grafos G de tamanho (quantidade de vértices) n par com os vértices indexados a partir de 1, denomindados $v_1, v_2, \ldots, v_n$ e vamos denotar o custo entre dois vértices distintos v_i e v_j por $c_{i,j}$.

Para que o custo da solução seja próxima do dobro da solução ótima vamos definir os custos de algumas arestas. Para todo $i \leq n/2$, $c_{i,2i} = 1$, para todo $i \leq n-2$, $c_{i,i+2} = 1$, para todo $1 < i \leq n/2$, $c_{i,2i-2}$ e, além disso, $c_1, n = n/2$. Note que estes custos respeitam a desigualdade triangular. Agora, basta completar as arestas de G adicionando entre todo par de vértices não adjacentes uma aresta de custo igual ao caminho mínimo entre os dois vértices, preservando a desigualdade triangular.

Uma possível árvore geradora T criada pelo algoritmo vai escolher todas as arestas (v_i, v_{2i}) com $i \leq n/2$ e (v_i, v_{i+2}) com $i < n/2$. E próximo passo é então encontrar atalhos no caminho euleriano formado pela duplicação das arestas de T e adicionar uma aresta entre v_1 e v_n para fechar um ciclo hamiltoniano. Os atalhos tomados sempre serão entre v_{2i-2} e v_i para todo i tal que $1 < i \leq n/2$. Como descrito acima, o custo de cada uma dessas arestas é 2. Concluímos que o custo da solução encontrada pelo algoritmo é $n/2 * 1$ por todas as arestas que vão de v_i para v_{2i} somado a $(n-1)/2 * 2$ pelos atalhos tomados e $n/2$ pelo retorno de v_n a v_1. Ao todo, obtemos um custo de $2n - 2$ que chamaremos $\bar{t}$.

Finalmente, temos que a solução ótima do algoritmo é n. Ela é pelo menos n pois toda aresta custa pelo menos 1 e não é possível obter uma solução com menos do que n arestas e ela é no máximo n pois existe uma solução de custo n formada por toda aresta da forma (v_i, v_{i+2}) com $i/leqn - 2$ além das arestas (v_1, v_2) e (v_{n-1}, v_n), todas de custo 1 formando um circuito de custo n que chamaremos t^*. Assim, podemos ver que $\bar{t}/t^*$ se aproxima de 2 com o crescimento de n, logo, esta família de instâncias gera faz com que a solução encontrada chegue arbitrariamente próxima de $2\text{opt}(G, c)$.

Para adaptar esta família para o algoritmo TSPM-Christofides basta exigir que n seja multiplo de 4. Assim, podemos considerar a mesma árvore T que seria encontrada no caso anterior. O emparelhamento M encontrado pelo algoritmo envolveria as arestas da forma (v_{i-2}, v_i) para todo i multiplo de 4 e todas elas, juntamente com a aresta (v_1, v_n) formariam a solução encontrada pelo algoritmo. Esta solução tem custo $n-1+n/2$, $n-1$ por todas as arestas que originalmente estavam em T ($n/2+n/4-1$ delas) mais as arestas de M ($n/4$ delas) somado ao $n/2$ referente ao custo da aresta (v_1, v_n).

Temos então que, neste caso, $\frac{\bar{t}}{t^*} = \frac{n-1+n/2}{n} = 1+1/2-1/n = 3/2-1/n$ que se aproxima de 3/2 com o crescimento de n, ou seja, é arbitrariamente próximo de 3/2. □

Ex 6. *Mostre que os algoritmos* TSPM-RSL *e* TSPM-CHRISTOFIDES *podem produzir péssimos resultados se aplicados a instâncias do* TSP *que não satisfazem a desigualdade triangular.*

Resposta. Basta montar uma instância onde $G \simeq K_4$, $c_{1,4}$ é arbitrariamente grande e o custo das outras arestas é único e arbitrariamente pequeno. Ambos os algoritmos podem encontrar como árvore geradora um caminho entre v_1 e v_4 que só vai poder ser completado com a adição da aresta (v_1, v_4), tornando a solução gigante, enquanto uma solução de custo pequeno era possível com o ciclo v_1, v_2, v_4, v_3, v_1. □

Ex 7. *Considere a seguinte variante do* TPSM*: queremos encontrar um caminho hamiltoniano de custo mínimo no grafo dado que começa em um dado vértice s. Modifique o algoritmo* TPSM-CHRISTOFIDES *e obtenha um algoritmo de aproximação com razão menor que 2 para essa variante.*

Resposta. Não consegui algo menor do que 2. Consigo dizer que é fácil achar um algoritmo com razão 2. Basta usar o TSPM-RSL, porém, ao duplicar as arestas da árvore T encontrada, não duplicamos uma das arestas que chegam a s e uma das arestsas que chegam a um vértice t qualquer diferente de s e, depois de fazer isso, encontramos uma trilha euleriana entre s e t e encontramos atalhos como fazem ambos os algoritmos normalmente.

É fácil ver que esta é uma 2-aproximação para o problema, pois o custo da árvore T gerada é menor do que a resposta ótima e o custo da resposta dada pelo algoritmo é menor do que duas vezes o custo da árvore. $\square$