Exercício 8.19 (Tardos)

Alice Duarte Scarpa, Bruno Lucian Costa

2015-06-23

1 Enunciado

Um comboio de navios chega ao porto com um total de n vasilhames contendo tipos diferentes de materiais perigosos. Na doca, estão m caminhões, cada um com capacidade para até k vasilhames. Para cada um dos dois problemas, dê um algoritmo polinomial ou prove NP-completude:

- Cada vasilhame só pode ser carregado com segurança em alguns dos caminhões. Existe como estocar os n vasilhames nos m caminhões de modo que nenhum caminhão esteja sobrecarregado, e todo vasilhame esteja num caminhão que o comporta com segurança?
- Qualquer vasilhame pode ser colocado em qualquer caminhão, mas alguns pares de vasilhames não podem ficar juntos num mesmo caminhão. Existe como estocar os n vasilhames nos m caminhões de modo que nenhum caminhão esteja sobrecarregado e que nenhum dos pares proibidos de vasilhames esteja no mesmo caminhão?

2 Item a

Uma solução força-bruta para esse problema seria:

- Estenda a lista de vasilhames com vasilhames vazios, até que ela tenha tamanho mk (a capacidade total de todos os caminhões). Vasilhames vazios podem ser transportados em qualquer caminhão (e correspondem a um lugar sobrando no mesmo).
- Para cada uma das $(mk)!$ ordenações da lista acima, considere que os k primeiros vasilhames vão para o primeiro caminhão, os k próximos para o segundo e assim até o final da lista. Se cada vasilhame estiver

em um camihão que o comporta com segurança, retorne essa solução, se não, tente com uma nova ordem.

Esse algoritmo faz $(mk)!$ iterações do loop principal no pior caso, cada iteração tem custo mk para conferir se é uma solução válida. Isso dá uma complexidade total de $O(mk(mk)!)$

Esse é um algoritmo super-exponencial para o problema, mas isso não significa que o problema é NP-completo. Na verdade, como veremos a seguir, esse problema não é NP-completo pois aceita uma solução polinomial usando fluxos.

2.1 Solução com fluxos

Podemos transformar esse problemas em um problema de encontrar o fluxo máximo de um grafo usando a seguinte construção:

- Criamos um vértice s representando a fonte e um vértice t representando o dreno
- Para cada vasilhame $v_i \in \{v_1, v_2, \ldots, v_n\}$ criamos um vértice v_i e uma aresta (s, v_i) capacidade 1
- Para cada caminhão $C_i \in \{C_1, C_2, \ldots, C_m\}$ criamos um vértice C_i. Se o vasilhame v_j puder ser transportado com segurança no caminhão C_i criamos uma aresta (v_j, C_i) de capacidade 1. Para cada caminhão criamos também uma aresta (C_i, t) de capacidade k.

Dessa forma, existe uma configuração possível de caminhões se e somente se o fluxo máximo tem valor m.

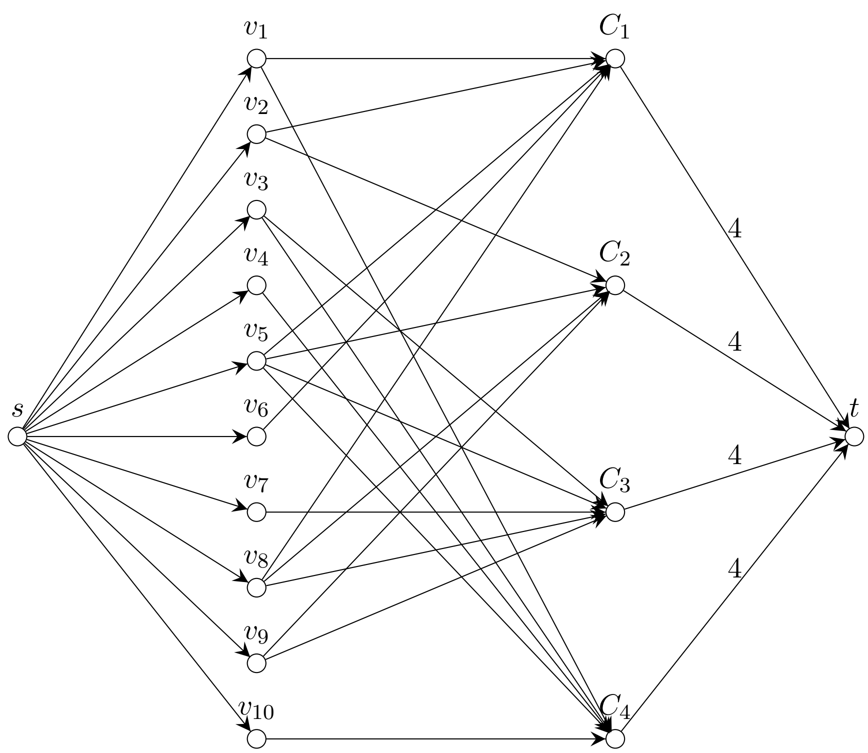
De fato, se encontramos um fluxo máximo de valor m então exatamente uma aresta com origem em cada vasilhame terá fluxo 1. Se colocarmos cada vasilhame no caminhão de destino da aresta de fluxo 1 obtemos um posicionamento válido. Por outro lado, se existe um arranjo válido, colocando 1 de fluxo nas arestas entre os vasilhames e os caminhões que os contém nesse arranjo obtemos um fluxo de valor m.

Para a seguinte situação:

Capacidade	3
Total de cam.	4
Total de vas.	10

Vasilhame	Cam. seguros
1	1, 4
2	1, 2
3	3, 4
4	4
5	1, 2, 3, 4
6	1
7	3
8	1, 2, 3
9	2, 3
10	4

A construção seria como ilustrado na figura abaixo. Omitimos as capacidades iguais a 1 para não poluir demais a imagem.



2.1.1 Implementação

Primeiramente, precisamos ser capazes de ler a tabela acima para passar os valores para o nosso algoritmo.

```
capacidade_por_caminhao = regras[0][1]
total_de_vasilhames = regras[2][1]

vasilhames = collections.OrderedDict()
caminhoes = []
for line in seguros[1:]:
    # Nomeando os vasilhames
    vasilhame = 'v_%s' % line[0]
    vasilhames[vasilhame] = []
    for caminhao in str(line[1]).split(','):
        nome = 'C_%s' % caminhao.strip()
        vasilhames[vasilhame].append(nome)
        if nome not in caminhoes:
            caminhoes.append(nome)
```

Vamos usar a classe RedeDeFluxo, que definimos para a questão 7.28.

```
def cria_grafo(vasilhames, caminhoes, capacidade_por_caminhao):
    G = RedeDeFluxo()
    G.novo_vertice('Fonte')
    G.novo_vertice('Dreno')

    # Criando um vertice para cada caminhao e ligando esse vertice ao dreno
    for caminhao in caminhoes:
        G.novo_vertice(caminhao)
        G.nova_aresta(caminhao, 'Dreno', capacidade_por_caminhao, 0)

    for vasilhame, caminhoes in vasilhames.iteritems():
        # Criando um vertice para cada vasilhame e conectando a fonte a
        # cada um dos vasilhames
        G.novo_vertice(vasilhame)
        G.nova_aresta('Fonte', vasilhame, 1, 0)

        # Conectando o vasilhame a cada caminhao que pode transporta-lo
        for caminhao in caminhoes:
            G.nova_aresta(vasilhame, caminhao, 1, 0)

    return G
```

Como nesse problema as demandas já são 0, podemos aplicar Ford-Fulkerson diretamente, usando a mesma implementação que fizemos para o exercício 7.28.

Podemos então rodar Ford-Fulkerson e ver se o fluxo máximo encontrado é igual ao total de vasilhames. Se for, isso significa que o problema tem uma solução, que vamos retornar. Caso contrário não existe arranjo possível.

```
G = cria_grafo(vasilhames, caminhoes, capacidade_por_caminhao)
fluxo = G.fluxo_maximo('Fonte', 'Dreno')
if fluxo == total_de_vasilhames:
    tabela_de_vasilhames = []
    for vasilhame in vasilhames:
        for w in G.adj[vasilhame]:
            if G.fluxo[w] == 1:
                tabela_de_vasilhames.append([w.origem, w.destino])
    return tabela_de_vasilhames
else:
    return 'Impossivel'
```

A solução para a nossa entrada:

v_1	C_4
v_2	C_2
v_3	C_3
v_4	C_4
v_5	C_1
v_6	C_1
v_7	C_3
v_8	C_1
v_9	C_2
v_{10}	C_4

2.1.2 Complexidade

Como vimos no exercício 7.28, O algoritmo de Ford-Fulkerson tem complexidade $O((V + E)F)$ em que V é a quantidade de vértices, E é a quantidade de arestas e F é o maior valor possível para o fluxo.

No caso, $V = m + n + 2$, $E \leq n + nm + m$ e o maior fluxo possível é n, totalizando uma complexidade máxima $O(n^2m)$, o que é polinomial na entrada.

3 Item B

Vamos mostrar que é possível reduzir uma instância do 3-SAT a um problema de colocar vasilhames em caminhões seguindo as restrições do enunciado. De modo que, como 3-SAT é NP-completo, nosso problema também é.

3.1 Definindo 3-SAT

3-SAT é o problema de dado um conjunto de variáveis $v_1, \ldots, v_x$ e cláusulas $K_1, \ldots, K_y$, onde cada cláusula é constituída de no máximo três elementos do universo de *literais*, $\{v_1, \overline{v_1}, \ldots, v_x, \overline{v_x}\}$, encontrar uma atribuição de valores Verdadeiro/Falso para cada variável que faça com que todas as cláusulas sejam verdadeiras.

3.2 3-SAT $\rightarrow$ Caminhões

Dada uma instância do 3-SAT, vamos construir uma instância do problema enunciado que admite solução se e somente se tal instância do 3-SAT admite solução.

Primeiramente, note que podemos assumir que nosso problema de 3-SAT tem pelo menos quatro variáveis, adicionando variáveis que não aparecem em cláusula alguma se necessário.

3.2.1 Construção

Vamos começar a construção sem as cláusulas:

- São $x + 1$ caminhões, cada um de capacidade $3x + y$
- Existe um vasilhame para cada um dos $2x$ literais em $\{v_1, \overline{v_1}, \ldots, v_x, \overline{v_x}\}$. Esses vasilhames têm o mesmo nome do literal a que correspondem.
- Existe um vasilhame adicional w_i para cada variável v_i

Queremos criar restrições entre os vasilhames de modo que, em uma atribuição válida:

1. x dos caminhões contenham exatamente um literal verdadeiro cada.
2. O caminhão restante contenha todos os literais falsos;

Para garantir as condições acima, vamos criar os seguintes conflitos:

- w_i conflita com w_j para todo $i \neq j$.
- w_i e conflita com v_j e com $\overline{v_j}$, se $i \neq j$.
- v_i conflita com $\overline{v_i}$.

Com isso, garantimos as seguintes propriedades:

(P_1) Como todos os w_i conflitam entre si, é necessário um caminhão por w_i.

(P_2) Como w_i conflita com todos os literais tais que $i \neq j$, o caminhão que contém w_i só pode conter literais correspondentes a i-ésima variável.

(P_3) v_i e $\overline{v_i}$ não podem estar ambas no caminhão do w_i, pois elas conflitam entre si.

(P_4) Mais ainda, *exatamente* um elemento do par $\{v_i, \overline{v_i}\}$ está no caminhão do w_i numa atribuição válida: Se nenhuma delas estivesse no caminhão do w_i, estariam ambas no único caminhão que não contém nenhum w (pois todos os outros caminhões contém um w_j com $i \neq j$, o que conflita com v_i e $\overline{v_i}$), o que também não pode acontecer.

Agora, vamos adicionar as cláusulas à nossa construção:

- Existe um vasilhame para cada uma das y cláusulas $K_1, \ldots, K_y$

Com seguinte conflito:

- K_i conflita com v_j se $v_j \notin K_i$. Similarlmente, K_i conflita com $\overline{v_j}$ se $\overline{v_j} \notin K_i$.

Ou seja, permitimos colocar o vasilhame da cláusula K_i num caminhão apenas se a cláusula contém todos os literais que vão viajar no caminhão.

Dessa forma, uma cláusula nunca pode viajar no caminhão dos literais falsos, pois cada a cláusula contém no máximo três literais e temos no mínimo quatro literais falsos, de modo que há garantidamente um literal que não aparece na cláusula e portanto conflita com ela.

3.2.2 Obtendo uma solução

Para completar a nossa redução, precisamos de duas coisas:

- A partir de uma solução do problema dos caminhões que construimos, encontrar em tempo polinomial uma solução do 3-SAT correspondente

- Provar que quando nenhuma solução do problema dos caminhões existe o 3-SAT também não tem solução

1. Solução caminhões $\rightarrow$ Solução 3-SAT

   Se existe uma solução para o problema, então todo caminhão que contém um vasilhame do tipo w_i também contém um vasilhame correspondente a um literal, pela propriedade P_4; esse literal será marcado como verdadeiro. Todos os outros literais serão marcados como falsos. Essa marcação é consistente, pois para cada $i \in \{1, 2, \ldots, x\}$ exatamente um literal entre $v_i, \overline{v_i}$ que está no mesmo caminhão que w_i. Como todas as cláusulas têm que estar em um caminhão que contém um vasilhame do tipo w_i e esse caminhão tem que conter um literal que está na cláusula, essa marcação faz com que todas as cláusulas sejam verdadeiras.

2. $\nexists$ solução caminhões $\Rightarrow \nexists$ solução 3-SAT

   É mais fácil provar a contrapositiva, isso é, $\exists$ solução 3-SAT $\Rightarrow$ $\exists$ solução caminhões.

   Seja S os conjuntos dos literais verdadeiros na solução do 3-SAT. Então:

   - $S \subset \{v_1, \overline{v_1}, \ldots, v_x, \overline{v_x}\}$
   - $\forall\ 1 \leq i \leq x, |S \cap \{v_i, \overline{v_i}\}| = 1$
   - $\forall\ 1 \leq j \leq y, S \cap K_j \neq \emptyset$

   Podemos construir uma solução válida para o problema dos caminhões da seguinte forma:

   - $\forall\ 1 \leq i \leq x$, coloque o vasilhame w_i no caminhão C_i
   - $\forall\ 1 \leq i \leq x$, coloque o vasilhame $S \cap \{v_i, \overline{v_i}\}$ em C_i
   - $\forall\ 1 \leq j \leq y$, seja i o menor valor tal que v_i ou $\overline{v_i}$ está em $S \cap K_j$. Coloque o vasilhame K_j em C_i.
   - Coloque todos os literais em $\{v_1, \overline{v_1}, \ldots, v_x, \overline{v_x}\} - S$ no caminhão C_{x+1}

   Isso respeita todas as restrições. De fato, cada K_j está num caminhão que só contém um vasilhame correspondente a um literal e, pelo item 3, o vasilhame do literal não conflita com o vasilhame da cláusula. Além disso, cada w_i está em seu próprio caminhão, nenhum par $\{v_i, \overline{v_i}\}$ aparece num mesmo caminhão e nenhum w_i aparece no mesmo caminhão de um literal v_j ou $\overline{v_j}$ com $i \neq j$.