Projeto e Análise de Algoritmos: Trabalho Prático 1

João Mateus de Freitas Veneroso

Departamento de Ciência da Computação da Universidade Federal de Minas Gerais

June 13, 2017

1 Introdução

Este relatório descreve a implementação do trabalho prático 2 da disciplina Projeto e Análise de Algoritmos. O trabalho consistiu em resolver um problema de compartilhamento de viagens por meio de três paradigmas diferentes: força bruta, programação dinâmica e algoritmos gulosos.

2 Força Bruta

O algoritmo de força bruta descrito abaixo é o mesmo implementado no Trabalho Prático 1.

O algoritmo 1 implementa uma solução de força bruta para o problema de compartilhamento de viagens. Primeiro inicializa-se o benefício máximo b^* com um valor negativo, dessa forma qualquer configuração válida vai proporcionar um benefício maior. A partir daí, no loop das linhas 5-10, para cada configuração válida, calcula-se o benefício total e atualiza-se o valor de b^* se este benefício for maior do que qualquer um encontrado até então. Além disso, a variável G_p^* guarda a configuração que proporcionou o maior benefício. Ao final do algoritmo, b^* guarda o valor do benefício máximo para o grafo G e G_p^* guarda a configuração que proporcionou este benefício.

A complexidade deste algoritmo depende do número de configurações G_p diferentes e do custo da função $ConstraintsAreValid$. O número de configurações G_p diferentes é 2^m para m igual ao número de arestas no grafo G. Pois, cada aresta pode estar presente ou não em G_p e nós queremos as combinações possíveis para todas as arestas m. O custo da função $ConstraintsAreValid$ depende do número de arestas, pois, para cada aresta (u, v), temos de verificar se ela é a única aresta que sai do vértice u, se o vértice v é um motorista e se v

possui espaço para acomodar todos os passageiros de u. Como todas estas operações tem custo constante, a função $ConstraintsAreValid$ tem custo $O(m)$. Por último, a linha 7 calcula a soma dos pesos de todas as arestas também com custo $O(m)$. Logo, a complexidade total do algoritmo é $2m2^m$ e o algoritmo é $O(m2^m)$.

Algorithm 1 MaximizeBenefit

```
1: procedure MaximizeBenefit(G = (V,A))
2:     b* ← −1
3:     G*_p ← ∅
4:
5:     for all G_p = (V, A_p) : A_p ⊆ G.A do
6:         if ConstraintsAreValid(G_p) then
7:             b ← Σ_{(v_i,v_j)∈A_p} B(v_i, v_j)
8:             if b > b* then
9:                 b* ← b
10:                G*_p ← G_p
```

Uma melhora ainda pode ser alcançada se deixarmos de testar parte das configurações inválidas de G. Sabemos que um passageiro só pode pegar carona em uma rota, portanto, qualquer combinação G_p onde existirem arestas $(u, v_i), (u, v_j) : i \neq j$ é inválida. Dessa forma, basta manter uma única aresta ativa por vez na lista de adjacência de cada vértice. Por exemplo, na figura 1 apenas duas combinações precisam ser testadas: $G_0(V, A_0) : A_0 = \{(X, Y)\}$ e $G_1(V, A_1) : A_1 = \{(X, Z)\}$.

Além disso, é necessário lembrar que um vértice pode representar um motorista sem passageiros ou um passageiro que não vai pegar carona com ninguém, portanto, o número total de combinações se torna $\sum_{v \in V} G \rightarrow Adj(v) + 2$ e, no pior caso (um grafo completo), o número de combinações se torna $O(n^n)$ onde n é o número de vértices no grafo G. Feitas estas considerações, a complexidade assintótica do algoritmo é $O(mn^n)$. O algoritmo implementado em Python para este trabalho utiliza este método.

A complexidade de espaço do algoritmo original e da versão aprimorada é $O(n + m)$, pois o grafo é armazenado na forma de listas de adjacência.

3 Programação Dinâmica

No problema do compartilhamento de viagens modelado em grafos, os vértices do grafo representam viagens e as arestas do grafo representam compartilhamentos. Para fins de simplificação da explicação, consideremos que os vértices podem representar passageiros exclusivos, motoristas exclusivos ou passageiros-motoristas. Em qualquer solução válida os passageiros-motoristas tem de decidir se vão atuar como passageiros ou motoristas devido às restrições do problema. Dessa forma, sendo k o número de passageiros-motoristas, ao variar quais deles

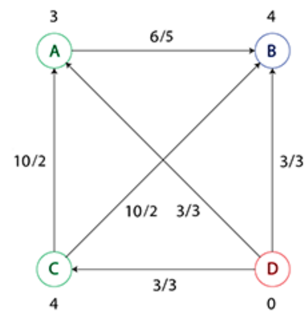

Figure 1: Exemplo do problema de compartilhamento de viagens modelado em forma de grafo

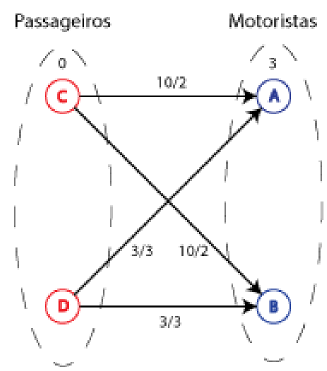

Figure 2: Grafo bipartido após definição dos passageiros e motoristas no grafo da figura 1

atuam como passageiros e quais atuam como motoristas, existem 2^k combinações possíveis que dão origem a problemas distintos. Cada um destes problemas pode ser pensado como um grafo bipartido G(V,A) onde os vértices de passageiros pertencem a um subconjunto de V e os vértices de motoristas pertencem a um outro subconjunto de V, sendo que todas as arestas conectam um vértice passageiro a um vértice motorista e os dois subconjuntos são disjuntos.

A figura 1 mostra um exemplo do probema de compartilhamento de viagens modelado na forma de um grafo, onde vértices vermelhos representam grupos de passageiros, vértices azuis representam motoristas e vértices verdes representam passageiros-motoristas. Os números próximos aos vértices representam a capacidade do veículo e os números próximos às arestas (v_i, v_j) representam o benefício do compartilhamento $B(v_i, v_j)$ e o número de passageiros na viagem $Q(v_i, v_j)$ separados por uma barra. Perceba que os vértices A e C são passageiros-motoristas e os demais vértices são passageiros ou motoristas exclusivos. Ao selecionar o vértice A como motorista e o vértice C como passageiro, podemos modelar o problema com o grafo bipartido da figura 2. Para cada uma das combinações de passageiros e motoristas podemos construir um grafo bipartido similar. Nesta forma, o problema passa a se parecer bastante com o *0-1 Multiple Knapsack Problem*, com a diferença que alguns items podem estar restritos à sacolas específicas. No caso, as sacolas representam os carros e os itens representam os grupos de passageiros. Ao definir a composição de benefício máximo para cada um destes problemas, basta escolher a composição de maior benefício para obter uma solução ótima para o problema completo.

Digamos que existem n grupos de passageiros $p_1, p_2, ..., p_n$ e m carros $c_1, c_2, ..., c_m$ no grafo bipartido, sendo que $|p_i|$ é o número de passageiros no grupo i e $|c_j|$ é a capacidade máxima do carro j. Agora, digamos que $P_i = p_1, p_2, ..., p_i$, $C_j = c_1), C(c_2), ..., C(c_j)$ e $B*(P_i, C_j)$ é o conjunto de arestas de benefício máximo para P_i e C_j, respeitando as restrições impostas pelas capacidades máximas dos carros. Para calcular o valor de $B*(P_i, C_j)$ temos três possibilidades:

1. o grupo de passageiros p_i viajará no carro c_j
2. o grupo de passageiros p_i viajará em algum carro no conjunto $c_1, c_2, ..., c_{j-1}$
3. o grupo de passageiros p_i não viajará em nenhum carro

No caso 1,

sendo que todo carro possui uma capacidade sendo que toda aresta (p_i, c_j) liga um passageiro a um carro.

Para cada aresta do grafo bipartido (v_i, v_j) que representa O problema descrito acima Subestrutura ótima

sobreposição de subproblemas

K matrizes com P linhas e C colunas. K = motoristas P = passageiros (p1, p2, p3, ..., pn) C = capacidade do carro