Problema do Maior Caminho

Cléber Machado
João Luiz Grave Gross

4 de junho, 2012

1 Caracterização do problema

O problema do caminho mais longo é o problema de encontrar o caminho simples de maior comprimento em um grafo, ou seja, dentre todos os possíveis caminhos simples presentes em um grafo o problema é encontrar o maior. [URUY] Um caminho é chamado de simples se ele não contiver vértices repetidos.

Ao contrário do problema do menor caminho, onde busca-se encontrar o menor caminho entre dois vértices e pode ser solucionado em tempo polinomial para grafos sem ciclos de pesos negativos, este problema de decisão é NP-completo, logo não possui solução ótima em tempo polinomial a não ser que P = NP.

A versão de decisão padrão do problema do maior caminho em grafos procura encontrar um caminho simples de comprimento maior or igual a k, onde o comprimento do caminho é definido pelo número de arestas ao longo do caminho. Para um grafo não valorado é suficiente encontrar o caminho mais longo, que tenha a maior quantidade de arestas e não repita vértices. Para grafos com arestas valoradas é preciso considerar seu peso ao invés da quantidade. [KHM] O foco deste artigo serão grafos com arestas não valoradas.

2 Prova que pertence a NP

Podemos provar que o problema pertence a NP mostrando que existe um algoritmo que recebe uma entrada e verifica em tempo polinomial se ela é solução do problema. Ou seja, dado um inteiro k e um grafo G, se existe um caminho de comprimento maior ou igual a k em G. Essa possível solução é chamada de certificado.

2.1 Algoritmo de verificação

A versão deste problema que pode ser chamada de problema de decisão é a seguinte:

Problema Caminho (k, G): Dado um grafo G e um natural k, decidir se existe em G um caminho de comprimento maior ou igual a k.

Se houvesse um algoritmo polinomial para resolver este problema de decisão, então have-ria um algoritmo polinomial para resolver o problema do caminho mais longo (e, como provaremos a seguir, para resolver todos os problemas NP-completos).

Suponha que a função caminho (k, G) receba um inteiro k e um grafo G e devolva sim se existe um caminho de comprimento maior ou igual a k em G e devolva não, caso contrario. Se esta função fosse polinomial, então o seguinte algoritmo, que encontra um caminho de comprimento máximo em um grafo G, seria polinomial. [RSF]

```
CAMINHO MAIS LONGO(G)
1   k ← ||V(G)|| - 1              //k é recebe número de vértices menos 1
2   enquanto (CAMINHO(k, G) = não)
3       k ← k - 1
4   H ← G                         //copia G para H
5   para i em E(H)                //seleciona uma aresta i de H
6       remova i de H
7       se (CAMINHO(k, H) = não)  //chama recursivamente CAMINHO
```

8 insira i de volta em H
9 retorne H

2.2 Análise da Complexidade

No algoritmo apresentado, há três etapas majoritárias durante sua execução, inicialização, iteração e finalização. Inicialmente k é inicializado com o número de vértices do grafo G, decrescido em uma unidade, visto que o maior caminho em um grafo conexo é a quantidade de arestas que ligam todos os vértices, ou seja, n - 1. Das linhas 2 até a linha 8 ocorre a iteração do programa. Nas linhas de 2 a 3 há um laço que a cada iteração computa uma função CAMINHO(k, G) de custo computacional n^2, realiza uma comparação e uma atribuição (linha 3). Na linha 4 também ocorre uma atribuição simples, ou seja, custo computacional constante O(1). Nas linhas de 5 a 8 há outro laço. A linha 6 realiza a remoção de um elemento, operação com custo constante. A linha 7 computa novamente a função CAMINHO() e também uma comparação, ou seja, custo $n^2 + 1$ e a linha 8 realiza uma inserção, operação também de custo constante. Por fim na linha 9, o resultado é retornado.

Acompanhe com mais detalhes a análise da complexidade:

A complexidade do algoritmo é dada pela soma das complexidades das linhas:
Calg = C[1] + C[2-3] + C[4] + C[5-8] + C[9]

C[1] = O(1) → Operação constante
C[2 - 3] = $O(n^2)$ → Função CAMINHO com custo n^2
C[4] = O(1) → Atribuição com custo constante

A complexidade das linhas 5 a 8 é dada pela soma das complexidades das linhas 6 a 8.
C[5-8] = C[6] + C[7] + C[8]

C[6] = O(1) → Remoção com custo constante
C[7] = $O(n^2)$ → Função CAMINHO com custo n^2
C[8] = O(1) → Inserção com custo constante

C[9] = O(1) → Retorno com custo constante

Complexidade final:
Calg = C[1] + C[2-3] + C[4] + **C[5-8]** + C[9]
Calg = C[1] + C[2-3] + C[4] + **C[6] + C[7] + C[8]** + C[9]
Calg = $1 + n^2 + 1 + 1 + n^2 + 1 + 1$
Calg = $5 + 2 * n^2$
Calg = $O(n^2)$

3 Prova que pertence a NP-completo

Um grafo G = (V, n), onde V é um grafo com n vértices. Definimos f(<G>) = <G, n - 1>, onde n é o número de vértices em G. Claramente f é computável em tempo polinomial: um algoritmo para computar f simplesmente conta o número de vértices do grafo G e anexa o valor decrescido de uma unidade ao seu resultado (saída), n - 1.

Assumimos que G possui um caminho hamiltoniano, ou seja, possui um caminho que contempla todos os vértices do grafo. Aplicamos a redução:

Caminho-Hamiltoniano ≤p Caminho-Mais-Longo

Se <G> ∈ Caminho-Hamiltoniano, então G tem um caminho hamiltoniano, i. e. G possui um caminho simples de comprimento n - 1, implica que para f(<G>) = <G, n - 1> ∈ Caminho-Mais-Longo.

Analogamente, se f(<G>) = <G, n - 1> ∈ Caminho-Mais-Longo, então G tem um caminho simples com n - 1 arestas, e também com n vértices. Como uma caminho simples com n vértices é um caminho hamiltoniano, confirmamos que <G> ∈ Caminho-Hamiltoniano.

Logo, mostrou-se que f é computável em tempo polinomial, e para todos os grafos G, <G> ∈ Caminho-Hamiltoniano ó f(<G>) ∈ Caminho-Mais-Longo. Assim Caminho-Hamiltoniano ≤p Caminho-Mais-Longo. Como Caminho-Hamiltoniado-Existe é NP-completo também podemos afirmar que Caminho-Mais-Longo é NP-completo. [MCP]

3.1 Redução inst a -> inst b

A redução do problema foi apresentada no item 3. Definimos que um grafo G é composto de V vértices e n arestas. Também definimos uma função f que recebe como parâmetro um grafo G e tem como saída o grafo com o tamanho do maior caminho, ou seja, f(<G>) = (G, n - 1).

Assumimos em G a existência de um caminho hamiltoniano e aplicamos G a f. O resultado foi um caminho simples de comprimento n - 1. Como a solução de f retorna apenas caminhos simples e sabendo que o caminho encontrado é hamiltoniano, comprovamos a existência do maior caminho simples em G, de tamanho n - 1.

3.2 Algoritmo de redução

A algoritmo de redução consiste em avaliar o grafo G, procurando a existência de um caminho hamiltoniado (Caminho-Hamiltoniado-Existe()). Caso G seja hamiltoniano, f retorna a quantidade de vértices de G decrescida de uma unidade (sempre considerando um grafo conexo). Eis o algoritmo:

ReduceHamiltonianToLongestPath(G)

```
1    path ← Ø        //path armazena o tamanho do caminho simples hamiltoniano
2    se (CAMINHO-HAMILTONIANO-EXISTE(G) = true)
3            path ← ‖V(G)‖ - 1
4    retorna <G, path>
```

3.3 Análise da complexidade

O algoritmo de redução é bastante simples de compreender. Inicialmente é inicializada uma das variáveis de saída, isto é feito em tempo fixo, ou seja, O(1). Nas linhas de 2 a 4 ocorre a execução principal do algoritmo. Na linha 2 um algoritmo de complexidade polinomial verifica a existência de um ciclo hamiltoniano em G. Se houver um caminho hamiltoniano, é feita uma atribuição simples do número de vértices do grafo decrescido de uma unidade em tempo O(1). Na linha 4 é realizada uma operação constante O(1). Considerando a complexidade pessimista, nas linhas 3 e 4 assumimos a complexidade O(n). Por fim, os resultados são retornados, O(1).

Logo, somando-se as complexidades de inicialização, execução e finalização, ficamos com uma complexidade final de O(n^m), onde m > 1 e n^m corresponde à complexidade polinomial da função CAMINHO-HAMILTONIANO-EXISTE().

Acompanhe com mais detalhes a análise da complexidade:

A complexidade do algoritmo é dada pela soma das complexidades das linhas:
Calg = C[1] + C[2-3] + C[4]

C[1] = O(1) → Atribuição simples, operação de custo constante
C[2 - 3] = O(n^m) → Função CAMINHO-HAMILTONIANO-EXISTE() com custo n^m
C[4] = O(1) → Retorno de custo constante

Complexidade final:
Calg = C[1] + C[2-3] + C[4]
Calg = 1 + n^m + 1
Calg = 2 + n^m
Calg = O(n^m)

4 Conclusão

Com este trabalho conseguimos realizar a prova de que o problema do maior caminho em grafos é um problema NP-completo, pois não possui solução em tempo polinomial. As provas e reduções nos mostraram que o problema em questão apenas é verificável em tempo polinomial, considerando grafos genéricos.

Grafos direcionados acíclicos por exemplo, apresentam solução em tempo linear, através da utilização de programação dinâmica. Porém nosso foco contempla toda a gama de grafos, e portanto apenas a verificação se deu em tempo polinomial.

Referências

[KHM] Khan, Mumit. CSE 221: Longest path in a directed acyclic graph (DAG). April 10, 2011.

[MCP] McCabe, Paul. University of Toronto. Professor of CSC363, Computational Complexity and Computability on Spring 2005. (http://www.cs.toronto.edu/~pmccabe/csc363-2005S/)

[RSF] Rezende, Susanna Figueiredo. Caminhos mais longos em grafos. Instituto de Matemática e Estatstica, Universidade de São Paulo, Brasil. 17 de fevereiro de 2012.

[URUY] Uehara, Ryuhei and Uno, Yushi. On Computing Longest Paths in Small Graph Classes. Department of Information Processing, School of Information Science, Japan Advanced Institute of Science and Technology (JAIST). Japan, July 28, 2005.