Animação de algoritmos sobre grafos cobertos por emparelhamentos

Guillermo Enrique Junchaya Heredia **Orientador: Cláudio Leonardo Lucchesi**

Departamento de Ciência da Computação, Instituto de Matemática e Estatística, Universidade de São Paulo

Introdução

Um *emparelhamento* de um grafo G é um subconjunto das arestas de G tal que, em cada vértice de G, incide no máximo uma aresta de M.
O conjunto M é um *emparelhamento máximo* se não existe nenhum outro emparelhamento de maior cardinalidade em G.

Problema do emparelhamento máximo: *Determine um emparelhamento máximo de um grafo dado qualquer.*

O primeiro algoritmo eficiente para resolver esse problema foi descoberto por Edmonds no seu famoso artigo de 1965, *"Paths, trees, and flowers"* [3].
O conjunto M é um *emparelhamento perfeito* se incide em todos os vértices do grafo. Uma aresta e do grafo é *emparelhável* se existe um emparelhamento perfeito do grafo que a contém. Um grafo é *coberto por emparelhamentos* se é conexo, emparelhável e todas as suas arestas são emparelháveis.

Problema dos grafos cobertos por emparelhamentos: *Determine se um grafo dado qualquer é coberto por emparelhamentos.*

O algoritmo mais eficiente conhecido para resolver esse problema foi descoberto por Carvalho e Cheriyan em 2005 [2].
Neste trabalho, estudamos e implementamos o algoritmo de Edmonds e o algoritmo de Carvalho e Cheriyan. Devido à natureza visual dos grafos, o software desenvolvido gera animações desses algoritmos, as quais ajudam o entendimento destes.

Caminhos aumentantes

Seja G um grafo e M um emparelhamento de G.

Um caminho (ou ciclo) *M-alternante* de G é um caminho (ou ciclo) que não tem duas arestas consecutivas fora de M. Um caminho *M-aumentante* de G é um caminho M-aumentante cujos vértices extremos não são cobertos por M.

Seja P um caminho M-aumentante de G. Note que $M' = M \triangle E(P)$ é um emparelhamento de G e que $|M'| = |M| + 1$. Logo, podemos aumentar o tamanho de um emparelhamento através de diferenças simétricas com o conjunto de arestas de caminhos aumentantes.
Em 1957, Berge enunciou o seguinte teorema [1].

Teorema de Berge: *Um emparelhamento M de um grafo G é máximo se e somente se não existem caminhos M-aumentantes em G.*

Deficiência de emparelhamentos

A *deficiência* de um emparelhamento M de um grafo G, escrita como $\mathrm{def}(M)$, é o número de vértices de G não cobertos por M.

Denotamos por $o(G)$ a cardinalidade do conjunto de componentes ímpares do grafo G.
Para todo subconjunto S dos vértices do grafo G, temos a seguinte proposição.

Certificado de otimalidade: *Se $\mathrm{def}(M) = o(G - S) - |S|$, então M é máximo.*

Um conjunto S que cumpre com as condições da anterior proposição é um certificado da otimalidade de M.

Vértices supérfluos

Um vértice v de um grafo G é um vértice *supérfluo* se G tem um emparelhamento máximo que não cobre v.

Um certificado S da otimalidade de um emparelhamento de um grafo G é um *certificado especial* se um vértice de G é supérfluo se e somente se pertence a alguma componente ímpar de $G - S$.

Flores

Os ciclos ímpares têm um papel importante no algoritmo de Edmonds. Por tal motivo, apresentamos as seguintes definições para um grafo G e um emparelhamento M de G.

Um *pedúnculo* é um caminho par M-alternante cuja origem é um vértice não coberto por M. Uma *corola* C é um ciclo ímpar M-alternante cuja origem não está coberta por $M \cap E(C)$. Uma *flor* é a concatenação de um pedúnculo e uma corola, de forma que o término do pedúnculo é a origem da corola.

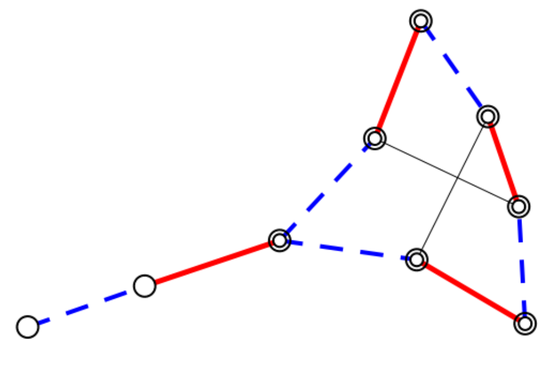

Figura 1: Uma flor. As arestas de M estão coloridas de vermelho e os vértices da corola são círculos duplos.

Algoritmo de Edmonds

A entrada do algoritmo de Edmonds é um grafo G e um emparelhamento M deste grafo. A saída do algoritmo é um emparelhamento máximo de G e um certficado especial da otimalidade desse emparelhamento.
Na sua execução, o algoritmo determina ou um caminho M-aumentante, ou uma flor ou um certificado especial da otimalidade de M. No caso em que é achado um certificado especial, o algoritmo para com o emparelhamento máximo M. No caso em que é achado um caminho aumentante P, o algoritmo determina recursivamente o resultado da sua execução com o grafo G e o emparelhamento $M \triangle E(P)$. Finalmente, no caso em que é achada uma flor, contrai-se sua corola C em um novo vértice, obtendo como resultado o grafo H. Então, o algoritmo determina recursivamente o resultado da sua execução com o grafo H e o emparelhamento $M - E(C)$.

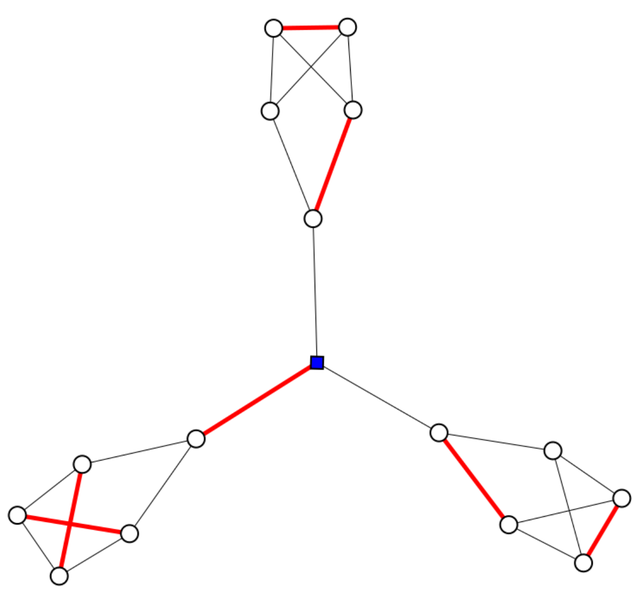

Figura 2: O resultado da execução do algoritmo de Edmonds no grafo de Sylvester. As arestas do emparelhamento são coloridas de vermelho e o certificado especial é o vértice quadrado azul.

Algoritmo de Carvalho e Cheriyan

A entrada do algoritmo de Carvalho e Cheriyan é um grafo G conexo e emparelhável, e um emparelhamento perfeito M deste grafo. A saída do algoritmo é uma coleção de arestas não emparelháveis de G. Se essa coleção for vazia, o algoritmo determina que o grafo G é coberto por emparelhamentos.
Seja u um vértice de G e uv a aresta de M que cobre u. O algoritmo de Carvalho e Cheriyan executa o algoritmo de Edmonds com o grafo $H := G - u$ e o emparelhamento $N := M - uv$ de H. Como N é um emparelhamento máximo de H, o resultado da execução do algoritmo de Edmonds é um certificado especial S da otimalidade de N em H.

Caracterização de arestas emparelháveis: *Uma aresta uw é emparelhável se e somente se o vértice w é supérfluo em H.*

Logo, como S é um certificado especial, uma aresta uw é emparelhável se e somente se o vértice w está em uma componente ímpar de $H - S$.
Repetindo essa iteração para todo vértice de G, o algoritmo de Carvalho e Cheriyan acha todas as arestas não emparelháveis do grafo.

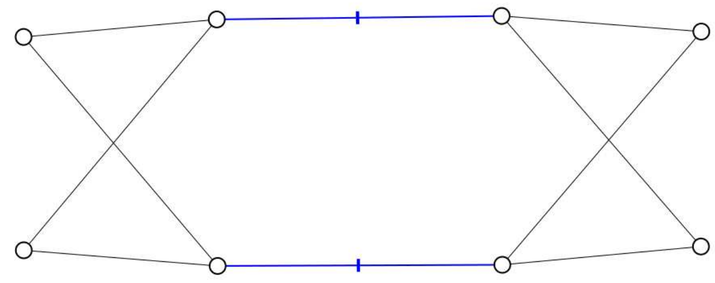

Figura 3: O resultado da execução do algoritmo de Carvalho e Cheriyan num grafo não coberto por emparelhamentos. As arestas não emparelháveis do grafo são coloridas de azul e têm uma barra no meio.

Implementação

O software deste trabalho foi desenvolvido usando a biblioteca gráfica `graph-tool` de Python [4]. Quando um algoritmo é executado com animação, é possível salvar todos os quadros da animação em uma pasta, ou abrir uma janela para ver a animação sendo executada. Além disso, podem-se executar os algoritmos sem animação e também testar as implementações com todos os arquivos de entrada disponíveis.
Esse programa é de código aberto e é possível adicionar novas funcionalidades, principalmente novos algoritmos, facilmente.

Informações e contato

A monografia do trabalho, o código fonte de software desenvolvido, algumas animações geradas e outras informações podem ser encontradas na página deste trabalho: `https://linux.ime.usp.br/~ejunchaya/tcc`

Endereço para contato:
`enriquejh@usp.br`

Referências

[1] Claude Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences of the United States of America*, 43(9):842–844, 1957. ISSN 00278424. URL `http://www.jstor.org/stable/89875`.

[2] Marcelo Henriques de Carvalho and Joseph Cheriyan. An $O(VE)$ algorithm for ear decompositions of matching-covered graphs. *ACM Trans. Algorithms*, 1(2):324–337, 2005. doi: 10.1145/1103963.1103969. URL `https://doi.org/10.1145/1103963.1103969`.

[3] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[4] Tiago P. Peixoto. The graph-tool python library. *figshare*, 2014. doi: 10.6084/m9.figshare.1164194. URL `http://figshare.com/articles/graph_tool/1164194`.