Emparelhementos em Grafos: Algoritmos e Implementações

Giovana Gomes Delfino

Instituto de Matemática e Estatística - Universidade de São Paulo

Objetivos

Neste trabalho apresentamos algoritmos relacionados ao problema do emparelhamento máximo desde os mais clássicos e simples como o aplicado a grafos bipartidos até os mais complexos e não tão conhecidos como o algoritmo de Edmonds-Blossom.
Diferentemente da maior parte dos livros e outras ferramentas de estudo as implementações aqui apresentadas serão bastante discutidas e a teoria necessária para chegar em tal implementação também será paralelamente introduzida. Todas as ideias apresentadas são acompanhadas de figuras que tentam ilustrar o argumento apresentado e às vezes de figuras que simulam o código em questão. Conforme os algoritmos são apresentados as complexidades são provadas da maneira mais clara e intuitiva possível.

Emparelhamento Máximo

Um emparelhamento em um grafo qualquer é um conjunto de arestas tais que duas a duas não sejam adjacentes, ou seja, quaisquer duas arestas escolhidas não possuem extremos em comum.

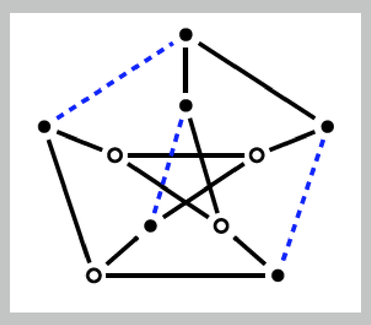

Figure 1: Arestas tracejadas representam um emparelhamento.

Portanto encontrar um emparelhamento mínimo é muito simples, o conjunto vazio, por exemplo, satisfaz as exigências. Por isso estamos interessados em encontrar um emparelhamento máximo, ou seja, um emparelhamento tal que a quantidade de arestas nele seja máxima.

Ideia

Muitos algoritmos foram desenvolvidos para resolver o problema do emparelhamento máximo, e todos se baseiam na ideias de encontrar caminhos de aumento o que muda é a forma que cada algoritmo os encontra.
Um caminho de aumento em um grafo é um caminho cujas arestas alternam em arestas que não pertencem ao emparelhamento e arestas que pertencem, além disso, para que seja um caminho de aumento o caminho deve começar e terminar com arestas que não pertencem ao emparelhamento.

Figure 2: Exemplo de caminho de aumento.

Caminhos de aumento recebem esse nome pois com eles é possível aumentar a cardinalidade do emparelhamento aplicando a operação da diferença simétrica entre o conjunto de arestas do caminho e o conjunto de arestas do emparelhamento.

Figure 3: Caminho resultante após o procedimento descrito.

Teorema de Berge

Seja G um grafo. M é um emparelhamento máximo em G se e somente se G não possui caminhos de aumento com relação a M.
Prova:
Se M possui caminhos de aumento acabamos de descrever um procedimento para aumentá-lo, logo M não pode ser máximo.
Se M não possui caminhos de aumento, suponha M^* um emparelhamento máximo, e considere os caminhos e circuitos do grafo induzido por $M \cup M^*$. Como M^* é máximo existe algum caminho com mais arestas de M^* do que de M, como não podem existir duas arestas consecutivas do mesmo emparelhamento (pois são emparelhamentos), algum caminho começa e termina com arestas de M^*, existe portanto um caminho de aumento com relação a M, o que contradiz a hipótese.

Algoritmo simples

O algoritmo mais simples encontra um emparelhamento em grafos bipartidos em tempo $O(|V|^2 + |V||E|)$. Para isso, o algoritmo começa em vértices não emparelhados e tenta encontrar um caminho que leve a outro não emparelhado, usando o caminho encontrado para aumentar o emparelhamento. Esse procedimento é repetido até que não existam mais caminhos de aumento, o que pelo teorema de Berge nos garante que o emparelhamento é máximo.

```
1: function AchaCaminhoAumento(u, G, mA, mB)
2:     for v ∈ adj[u] and seen[v] = false :
3:         seen[v] = true
4:         if mB[v] = v or achaCaminhoAumento(mB[v], G, mA, mB) = true :
5:             mA[u] ← v
6:             mB[v] ← u
7:             return true
8:     return false
```

Figure 4: Algoritmo que encontra emparelhamentos máximos em grafos bipartidos.

Hocroft Karp

O algoritmo de Hopcroft Karp também encontra emparelhamentos máximos em grafos bipartidos porém é mais eficiente pois ao invés de encontrar um caminho de aumento por vez, encontra um conjunto maximal de caminhos de aumento de comprimento mínimo. Para isso mantém uma árvore de caminhos.

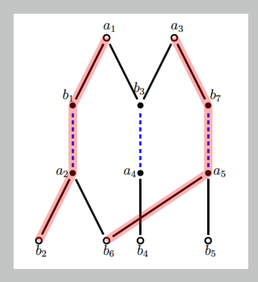

Figure 5: Exemplo de árvore de caminhos.

A imagem acima mostra uma iteração do algoritmo na qual dois caminhos de aumento são encontrados (os que estão em destaque). O algoritmo roda em tempo $O(\sqrt{|V|}(|V| + |E|))$

Edmonds Blossom

O algoritmo de Edmonds encontra emparelhamentos máximos em grafos genéricos. Assim como Hopcroft-Karp usa a ideia de contruir uma floresta de caminhos para encontrar um caminho de aumento, o que torna o algoritmo especial é a forma como lida com os ciclos ímpares de caminhos alternantes (chamados de blossom) que encontra, contraindo-os em um único vértice e repetindo o procedimento até que não existam mais caminhos de aumento.

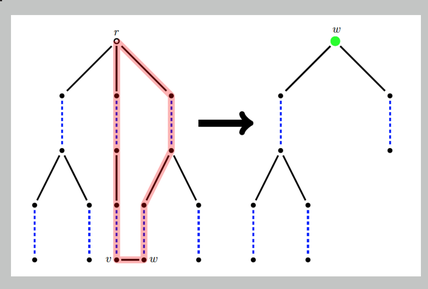

Figure 6: Exemplo de contração de blossom.

O algoritmo roda em tempo $O(|V|^2|E|)$

Método Húngaro

Método Húngaro resolve o problema de encontrar um emparelhamento máximo de peso máximo em grafos bipartidos, imagine que agora as arestas tenham peso, queremos então encontrar um conjunto de arestas cujos pesos somem o máximo possível. Para isso método basicamente resolve um problema de programação linear.

$$\begin{aligned} \text{Maximizar} \quad & \sum_{ij \in E} w(ij) x_{ij} \\ \text{sujeito a} \quad & \sum_{j \in B} x_{ij} = 1, \text{ para todo } i \in A \\ & \sum_{i \in A} x_{ij} = 1, \text{ para todo } j \in B \\ & x_{ij} \geq 0, \text{ para todo } ij \in E \end{aligned}$$

Figure 7: Primal do problema descrito.