“Método Húngaro para encontrar o emparelhamento de custo máximo”

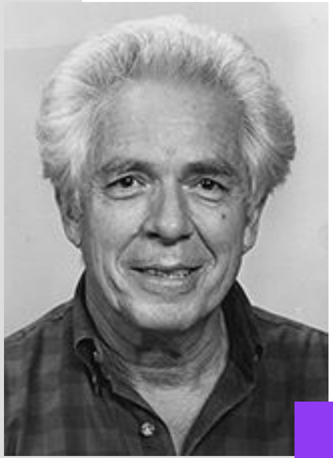
</>

~ João Gabriel & Alyson Noronha

[ARM/Linux - Desenvolvimento de SW embarcado]

{ História }

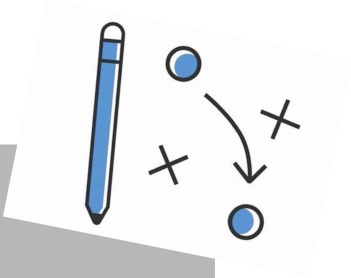
- Autor: Harold William Kuhn. *(Imagem a direita)*
- Data de publicação: 1955.
- Amplamente baseado nos trabalhos anteriores de dois matemáticos húngaros: Dénes Kőnig e Jenő Egerváry(daí o nome "método húngaro").



{ Motivações }

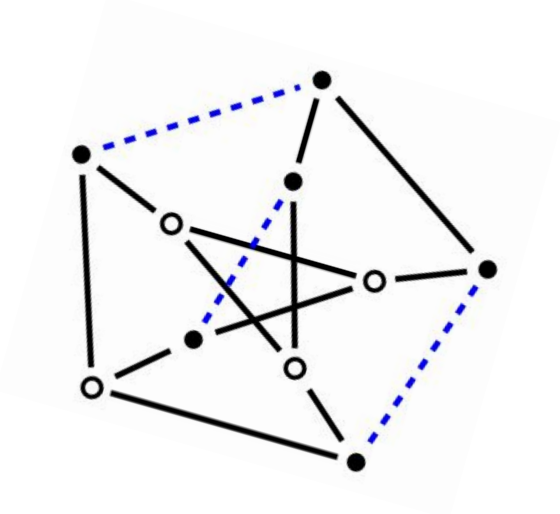
- Otimização combinatória.
- Tempo polinomial.
- < Resolução de problemas de alocação de tarefas/atribuição (***problema do emparelhamento***) />

{ Metodologia Central }



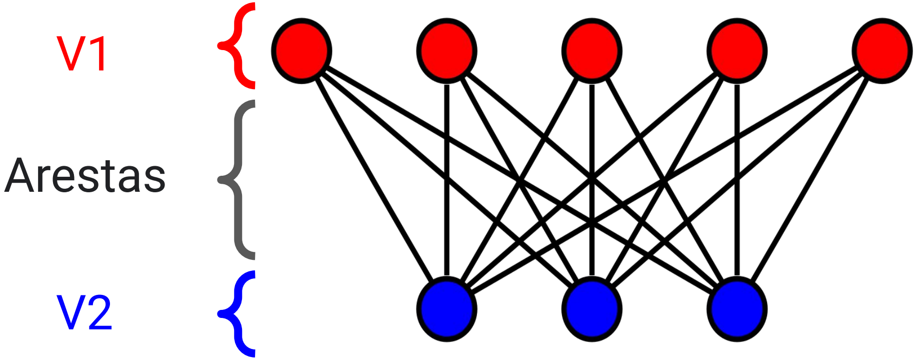
{Emparelhamento}

- Subgrafo onde não existem arestas que possuem vértices em comum.
[Imagem a direita.](#) (Arestas tracejadas formam um emparelhamento, mas arestas contínuas não.) //

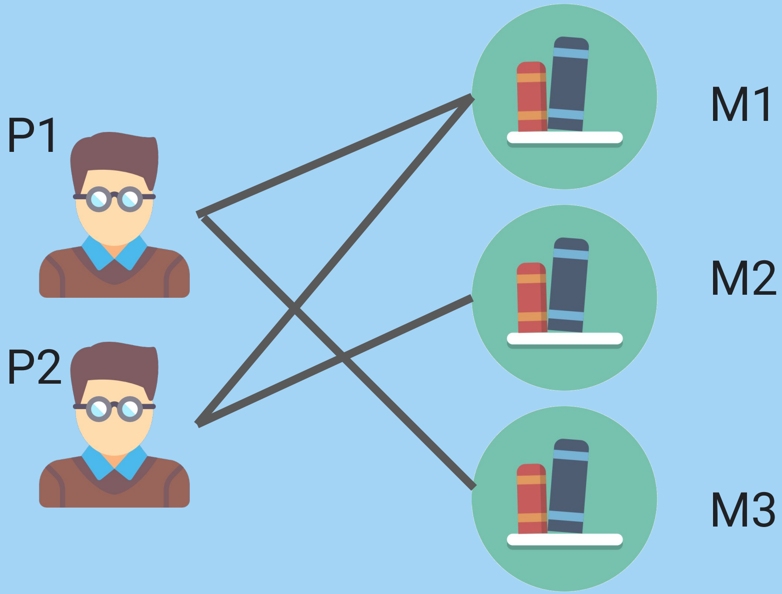


{ Grafo bipartido }

- Que pode repartir seus vértices em dois subconjuntos V1 e V2 de modo que todas as arestas de tal liguem um vértice de V1 com um vértice de V2.

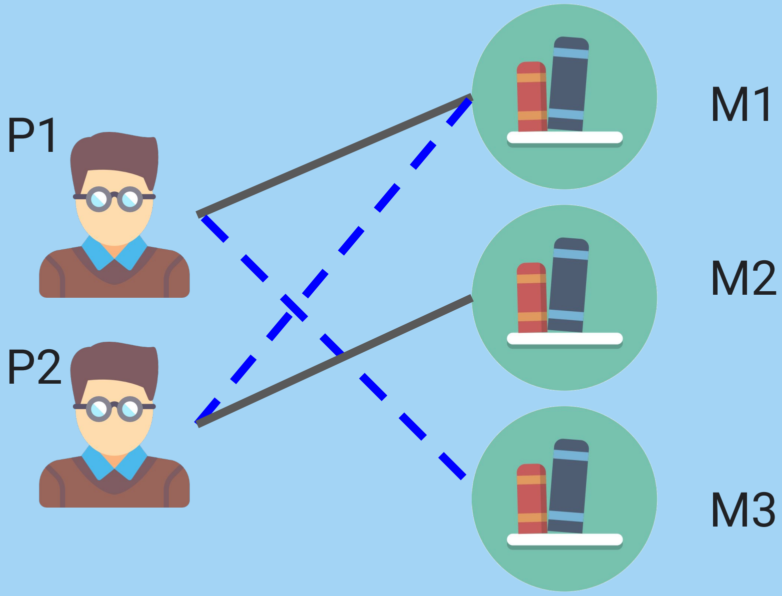


{ Problema }



- Um grafo bipartido que representa a relação de professores e determinadas disciplinas.
- Arestas Indicam que o professor pode dar a matéria ao qual está ligado.
- Disciplinas: M1, M2 e M3.
- Professores: P1 e P2.

{ Problema }



- Arestas tracejadas indicam que o professor de sua ponta dará a disciplina de sua outra ponta.
- Mesmo definindo a disciplina dada pelos professores essa foi a <u>melhor distribuição a ser feita</u>?
- Posso distribuir de forma que todas as matérias sejam oferecidas? Se não, qual o número máx de matérias que podem ser dadas? E o mín?

{ Problema - Cenários }

- *Atividades/Problemas cotidianos.*
- *“Atribuição de tarefas pessoais”.*
- *“Escalonamento de processos”.*
- *“Distribuição de funcionários e seus ofícios”.*
- *“Seleção de adversários em eventos esportivos”.*
- *etc ...*

{ Objetivo }

- Encontrar o **emparelhamento** de **custo mínimo ou máximo** para um **grafo**.
- Usar da resolução de problemas de emparelhamento que o método Húngaro traz para efetivar a obtenção de respostas satisfatórias em serviços de distribuição, alocação ou organização em setores profissionais do mundo atual.

{ Objetivo - Implementado }

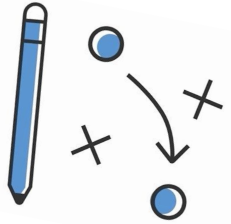
- Encontrar o **emparelhamento** de **custo máximo** para um **grafo bipartido**.

{ Problemas }

- Muitas restrições ao trabalhar com grafos.
- Muitas formas de achar emparelhamentos dos mais diversos tipos(custo máximo, mínimo e etc) em grafos.
- Complexidade da implementação do algoritmo aumenta.

{ Solução }

- Tratar o grafo como bipartido.
- Trabalhar apenas com o custo máximo.
- Implementação menos complexa.

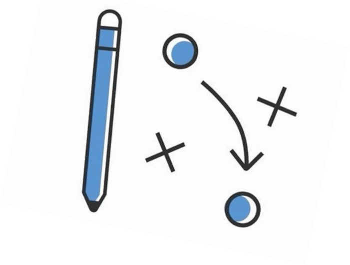


“Matemática do algoritmo”

{ Programação Linear }

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

$$
\begin{aligned}
\text{Minimizar} \quad & \sum_{i \in A} y_i + \sum_{j \in B} z_j \\
\text{sujeito a} \quad & y_i + z_j \geq w(ij), \text{ para todo } ij \in E
\end{aligned}
$$



"Matemática do algoritmo"

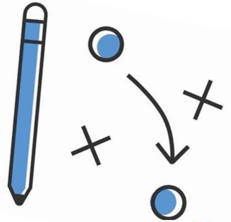
Teorema da Dualidade

$$\sum_{ij\in E} w(ij)x_{ij} \le \sum_{i\in A} y_i + \sum_{j\in B} z_j.$$

Folgas Complementares

$$\sum_{ij\in M^*} w(ij) \le \sum_{i\in A} y_i + \sum_{j\in B} z_j = \sum_{ij\in M} y_i + z_j = \sum_{ij\in M} w(ij).$$

{ Programação Linear }

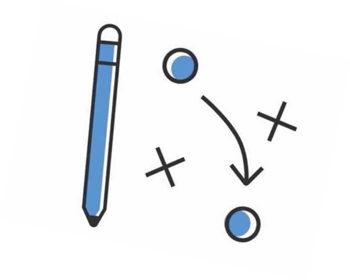


"Teoria do algoritmo"

1: **function** ATUALIZADUAL(y, z, d, S, T)
2: $\quad \delta \leftarrow min_{j \in B \setminus T} d_j$
3: $\quad$ **for all** $i \in S$: $y_i \leftarrow y_i - \delta$
4: $\quad$ **for all** $j \in B$:
5: $\qquad$ **if** $j \in T$:
6: $\qquad\quad z_j \leftarrow z_j + \delta$
7: $\qquad$ **else**
8: $\qquad\quad z_j \leftarrow z_j - \delta$

1: **function** MÉTODOHÚNGARO(G, w)
2: $\quad$ inicializa (y, z) e M
3: $\quad$ **while** M não é perfeito em G_{yz} :
4: $\qquad i \leftarrow i \in A$ *livre*
5: $\qquad S \leftarrow \{i\}$
6: $\qquad T \leftarrow \{\emptyset\}$
7: $\qquad$ atualiza d
8: $\qquad$ **while** Enquanto não encontrar caminho de aumento :
9: $\qquad\quad$ **if** $N_{yz} = T$:
10: $\qquad\qquad y, z, d \leftarrow atualizaDual(y, z, d, S, T)$
11: $\qquad\quad j \leftarrow j \in N_{yz}(S) \setminus T$
12: $\qquad\quad$ **if** j é livre :
13: $\qquad\qquad P \leftarrow$ caminho de i a j
14: $\qquad\qquad M \leftarrow M \triangle P$
15: $\qquad\qquad$ volte para linha 3
16: $\qquad\quad$ **else**
17: $\qquad\qquad i' \leftarrow$ vértice ao qual j está emparelhado
18: $\qquad\qquad S \leftarrow S \cup \{i'\}$
19: $\qquad\qquad T \leftarrow T \cup \{j\}$
20: $\qquad\qquad$ atualiza d
21: $\quad$ **return** M

{ Pseudo - Código }



Pior caso: $O(n^3)$
Melhor caso: $O(n^3)$

{ Pseudo - Código }

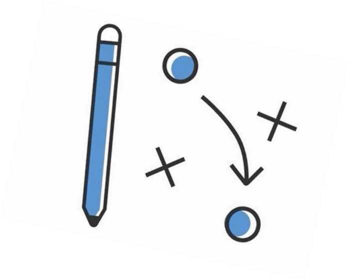
1: **function** MÉTODOHÚNGARO(G, w)
2: inicializa (y, z) e M
3: (**while** M não é perfeito em G_{yz} :) "N vezes"
4: $i \leftarrow i \in A$ *livre*
5: $S \leftarrow \{i\}$
6: $T \leftarrow \{\emptyset\}$
7: atualiza d
8: (**while** Enquanto não encontrar caminho de aumento :) "N vezes"
9: **if** $N_{yz} = T$:
10: $y, z, d \leftarrow atualizaDual(y, z, d, S, T)$
11: $j \leftarrow j \in N_{yz}(S) \setminus T$
12: **if** j é livre :
13: $P \leftarrow$ caminho de i a j
14: $M \leftarrow M \triangle P$
15: volte para linha 3
16: **else**
17: $i' \leftarrow$ vértice ao qual j está emparelhado
18: $S \leftarrow S \cup \{i'\}$
19: $T \leftarrow T \cup \{j\}$
20: atualiza d
21: **return** M

"Complexidades do algoritmo"

$O(n)$ (Linhas 9-20)

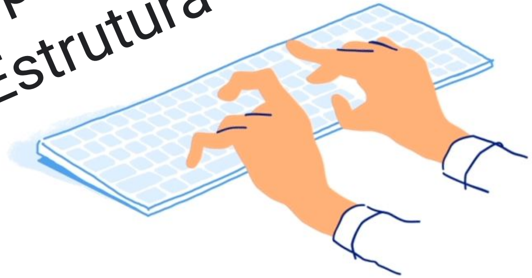
$O(n^2)$ (Linhas 8-20)

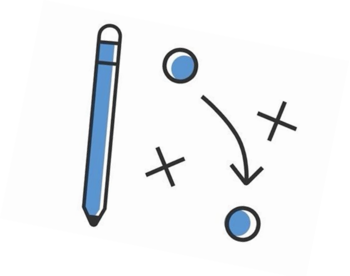
$O(n^3)$ (Linhas 3-20)



“Vamos ao algoritmo !”

{ Implementações
& Estrutura de dados usadas }





Ferramenta/Implementações de ajuda

- Implementação do método Húngaro pelo Eng.Software Lucas França. Principal ferramenta utilizada pela equipe para implementar seu próprio código.
- O código está em c++ e tivemos o desafio de implemntá-lo para C portanto os resultados de uma mesma entrada podem dar diferentes mas no quesito de se tratar de emparelhamento máximo nos grafos gerados pelas saídas os mesmos resultados se aproximam!

https://github.com/splucs/Competitive-Programming/blob/master/Macac%C3%A1rio/Graphs/hungarian.cpp

{ Testes e Validações }

"Algoritmo MUITO difícil de se implementar só!"

{ referências bibliográficas }

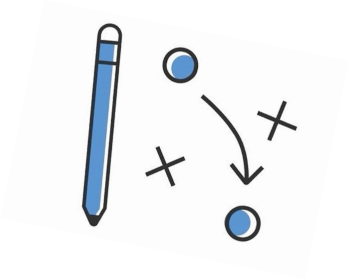
https://github.com/gidelfino/MAC0499/blob/master/monografia.pdf

https://en.wikipedia.org/wiki/Hungarian_algorithm

https://github.com/gidelfino/MAC0499/blob/master/codigos/metodo_hungaro.cpp

https://github.com/splucs/Competitive-Programming/blob/master/Macac%C3%A1rio/Graphs/hungarian.cpp

https://www.youtube.com/watch?v=VR9TXIG0MLA



“Método Húngaro para encontrar o emparelhamento de custo máximo”

</>

~ Obrigado!

Sim, essa é bandeira da hungria!