Computação Científica
Combinatória

Primeiros passos

Março/2009

Tópicos

- Introdução
- Otimização
- Problemas identificados
- Métodos de solução

Definição

- **Computação Científica Combinatória (CCC)**
  - um novo nome para pesquisa em uma área interdisciplinar que abrange computação científica e teoria dos algoritmos e otimização (Hendrickson e Pothen)

Pesquisa em CCC

- Três componentes básicos:
  - Identificação de um problema em Computação Científica e construção de um modelo combinatório para ele;
  - Projeto, análise e implementação de algoritmos para resolução dos problemas combinatórios levantados;
  - Desenvolvimento de *softwares*.

Pesquisa em CCC

rigor teórico + impacto prático

Modelagem de problemas

Para resolver um problema prático, muitas vezes o modelamos por uma formulação matemática e posteriormente aplicamos alguma técnica para obter uma solução ótima ou aproximada

Problemas e Gestão

Para entender um *problema*, temos que tentar ao menos algumas soluções mais óbvias, e descobrir que elas falham: então, redescobrimos que existe uma dificuldade - *um problema*

Karl R. Popper

Problemas e Gestão

Um *Problema* é uma dificuldade que impede que uma vontade seja concretizada.
Solucionar *Problemas* exige a capacidade de criar adequadas representações da realidade (*modelos*) e, com ajuda delas, encontrar um *algoritmo de solução* que explique como remover ou superar tal dificuldade

Problemas e Gestão

A construção de um *algoritmo de solução* é profundamente influenciada pelo modelo utilizado.

Problemas e Gestão

Solucionar problemas é, portanto, uma arte de *criar* ou escolher *modelos*, e com eles *construir algoritmos* que *funcionem* na prática e sejam *rápidos* o *suficiente* para ainda encontrarem o problema quando oferecerem a solução.

Problemas e Gestão

Tipos de Problemas:

- ✓ Decidíveis
- ✓ Não Decidíveis

Problemas e Gestão

Tipos de Problemas Decidíveis:

- ✓ Decisão
- ✓ Localização
- ✓ Otimização

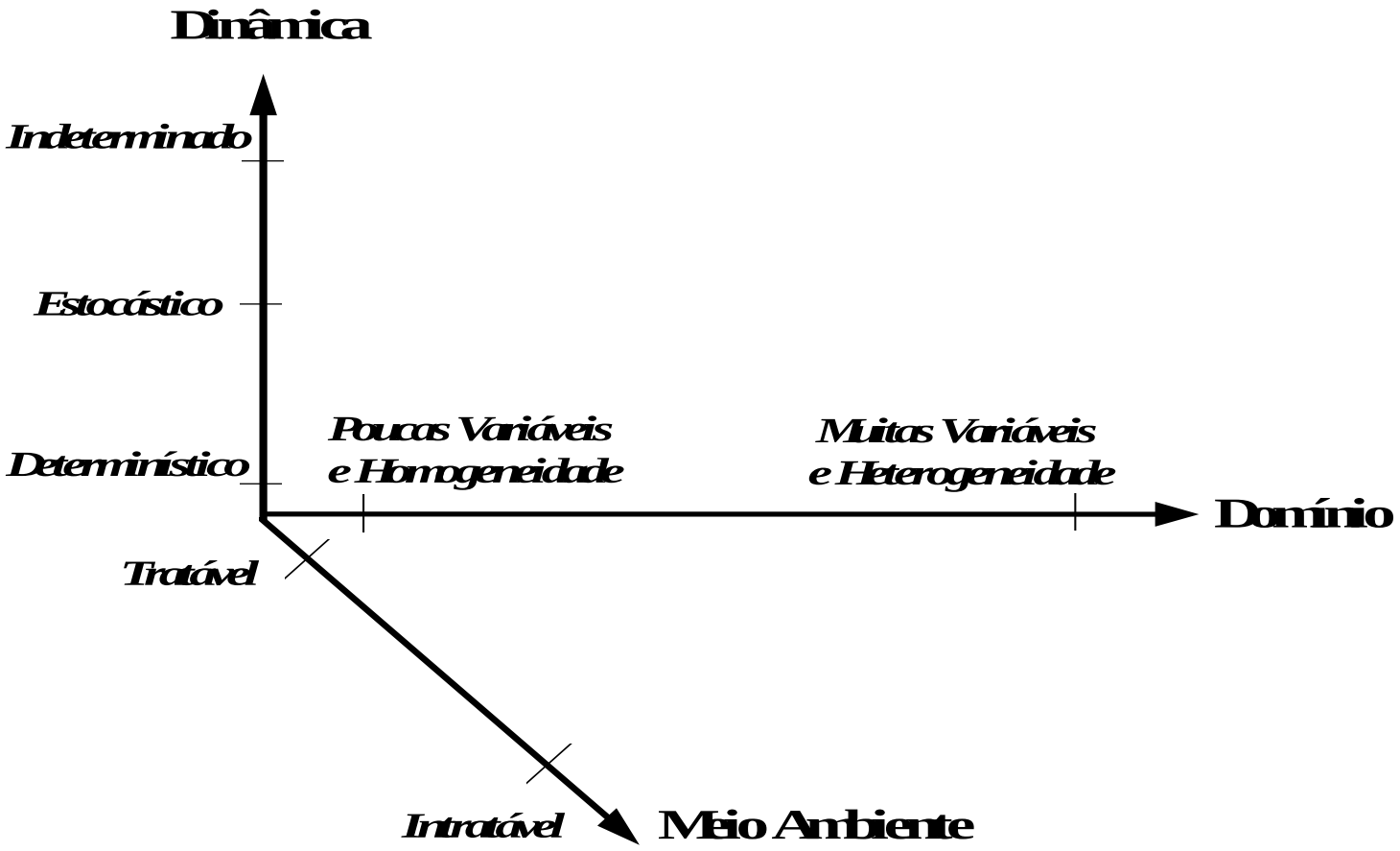
Conceito de Modelo

Os *Modelos* são representações simplificadas da realidade que preservam, para determinadas situações e enfoques, uma equivalência adequada

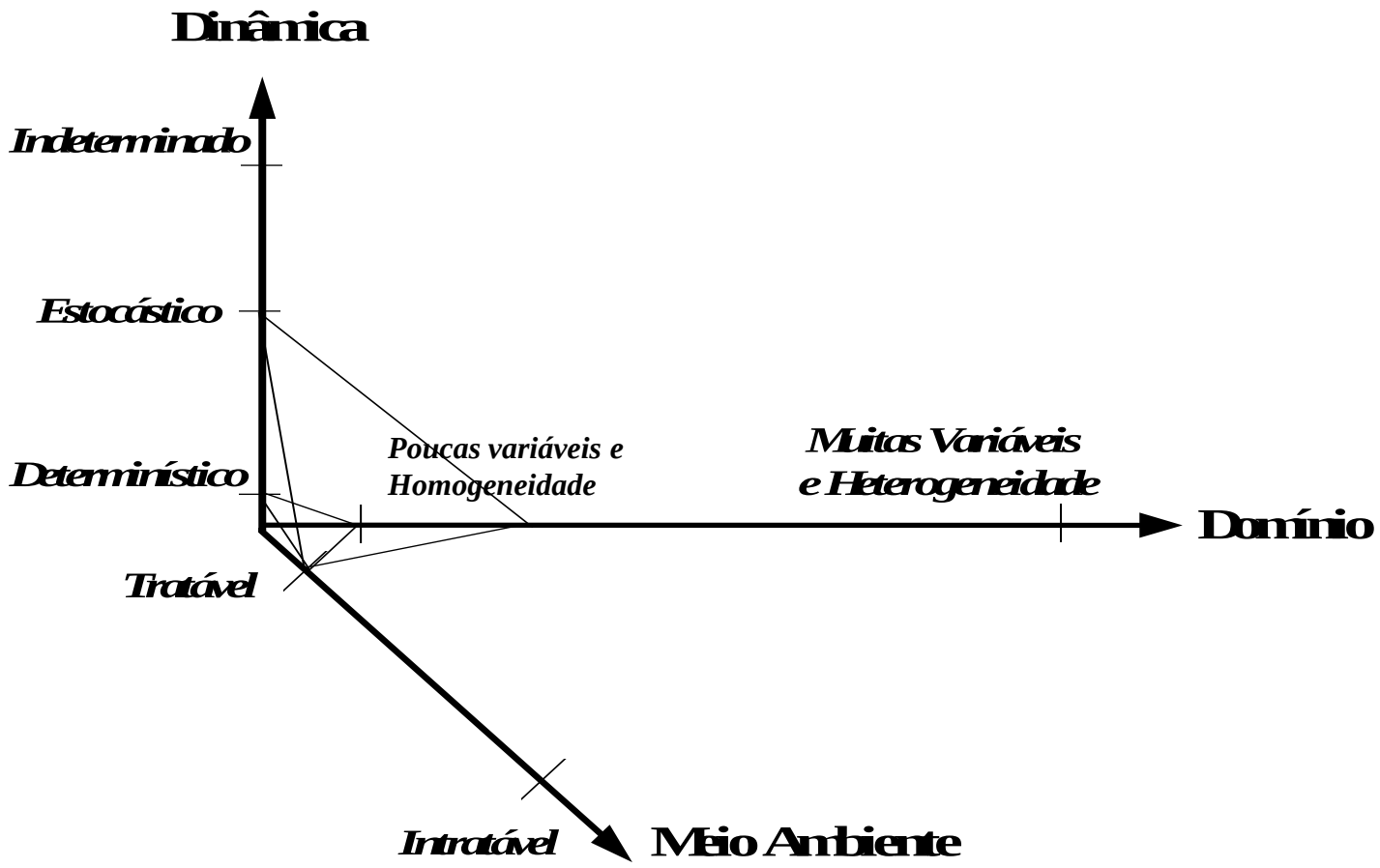
O Modelo Sistêmico

Sistemas são unidades conceituais ou físicas, compostas de partes interrelacionadas, interatuantes e interdependentes

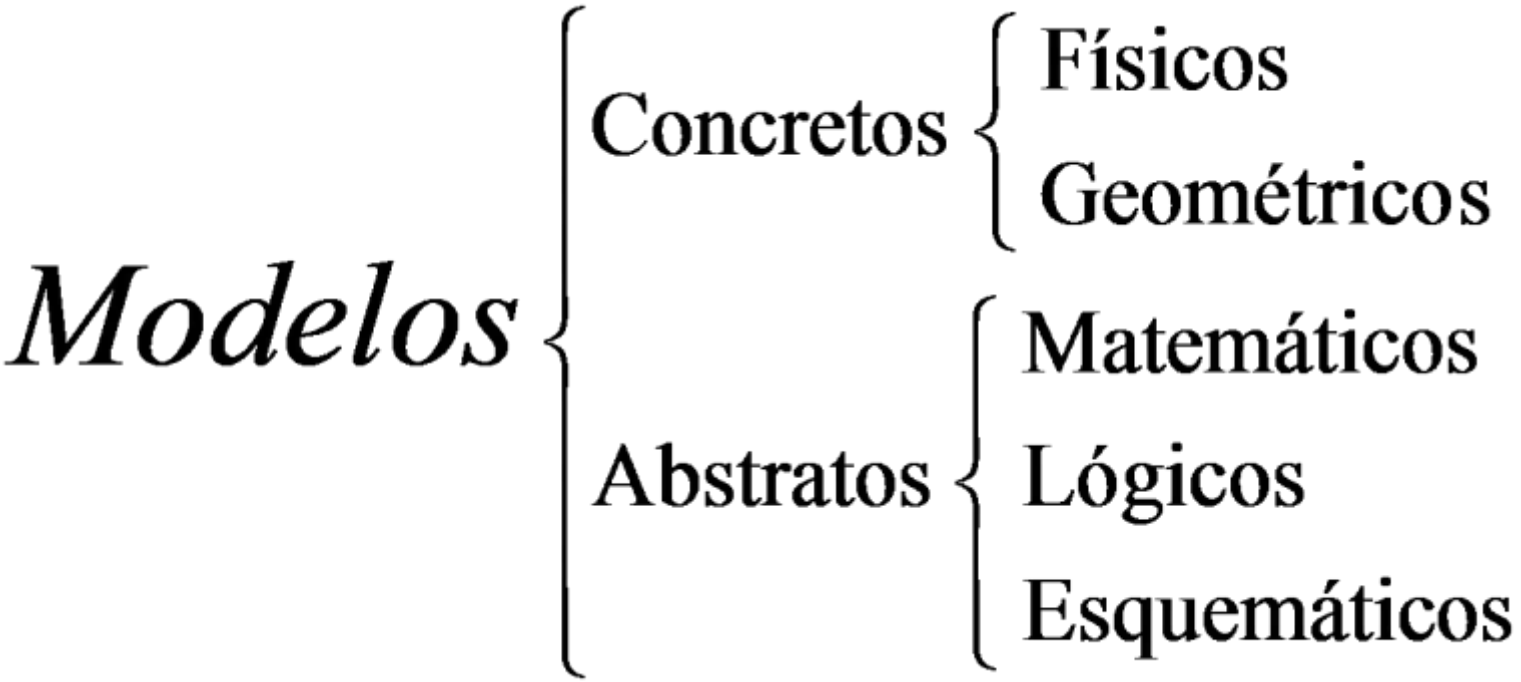
A dimensão da Complexidade



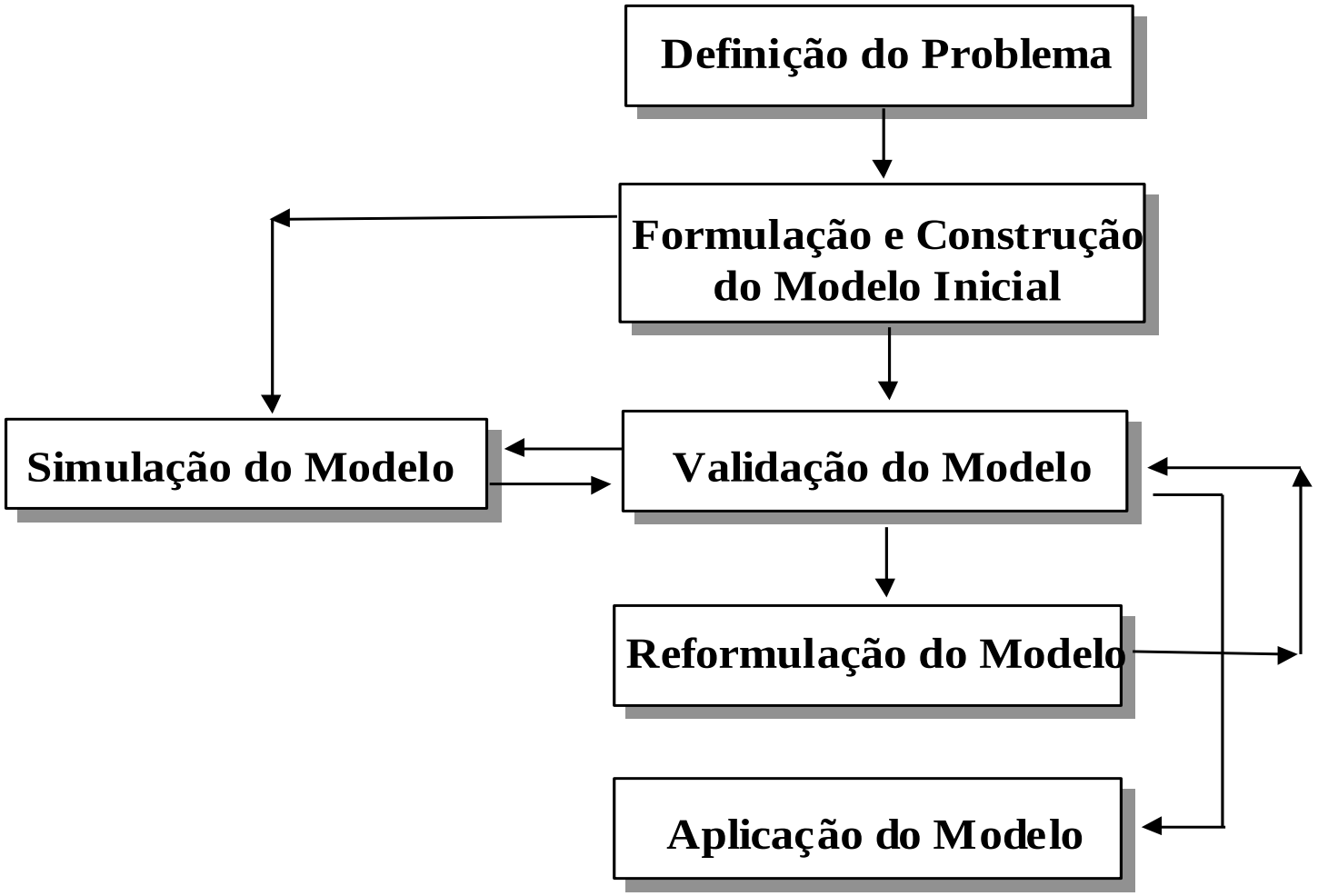
A dimensão da Complexidade



Classificação



Modelagem


Definição do Problema
Formulação e Construção do Modelo Inicial
Simulação do Modelo
Validação do Modelo
Reformulação do Modelo
Aplicação do Modelo

Teoria da Decisão

- ✓ Teoria de Utilidade
- ✓ Teoria de Probabilidade
- ✓ Pesquisa Operacional

Modelo de Otimização

$$\textit{Minimizar } f(x)$$

$$\textit{Sujeito a}:$$

$$h_i(x) = 0, \quad i = 1,\ldots,m_h$$

$$g_j(x) \leq 0 \quad j = 1,\ldots,m_g$$

$$x \in \Re^n$$

f:$\Re^n \rightarrow \Re$ g:$\Re^n \rightarrow \Re$ h:$\Re^n \rightarrow \Re$

Programação Matemática

- ✓ Programação Linear
- ✓ Programação Não-Linear
- ✓ Programação Inteira

Programação Matemática

- ✓ Melhorias Mensuráveis
- ✓ Automatização de Processos
- ✓ Análises Operacionais
- ✓ Identificação de Gargalos
- ✓ Determinação de Valores
- ✓ Projetos e Reengenharia

Modelos

Os modelos quantitativos não tomam as decisões, mas as tornam muito mais *claras e fáceis*

Exemplo

$$(BS)\ \text{Maximizar} z = x_1 + 3x_2$$

sujeitoa:

$$x_1 \leq 40$$

$$x_2 \leq 60$$

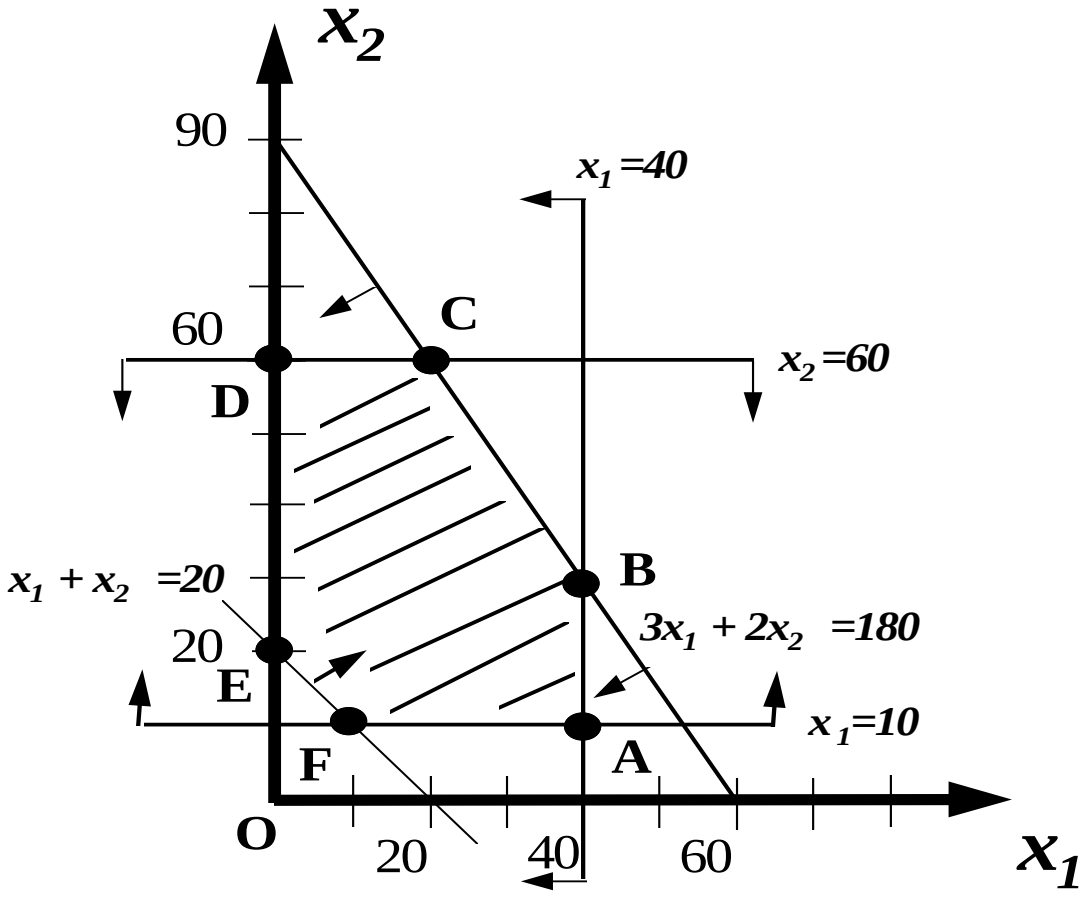
$$x_2 \geq 10$$

$$x_1 + x_2 \geq 20$$

$$3x_1 + 2x_2 \leq 180$$

$$x_1 \geq 0, x_2 \geq 0$$

Solução Gráfica



Solução Exaustiva

Pontos Examinados	*Coordenadas* (x_1, x_2)	*Valor da função* $z = x_1 + 3x_2$
A	*(40,10)*	*70*
B	*(40,30)*	*130*
C	*(20,60)*	*200*
D	*(0,60)*	*180*
E	*(0,20)*	*60*
F	*(10,10)*	*40*

Outro Exemplo

Maximizar $z = x_1 + 19x_2$

sujeito a:

$$x_1 + 20x_2 \leq 50$$

$$x_1 + x_2 \leq 20$$

x_1, x_2 variáveis inteiras

O Problema da Mochila (PK)

$$(PK) \quad \text{Maximizar } z = \sum_{j=1}^{n} c_j x_j$$

sujeito a :

$$\sum_{j=1}^{n} w_j x_j \leq b$$

$$x_j \geq 0 \text{ e inteiro}$$

O Problema da Mochila Unidimensional

$$(PKI) \quad \text{Maximizar } z = \sum_{j=1}^{n} c_j x_j$$

sujeito a :

$$\sum_{j=1}^{n} w_j x_j \leq b$$

$$x_j \in \{0,1\} \qquad j = 1,\ldots,n$$

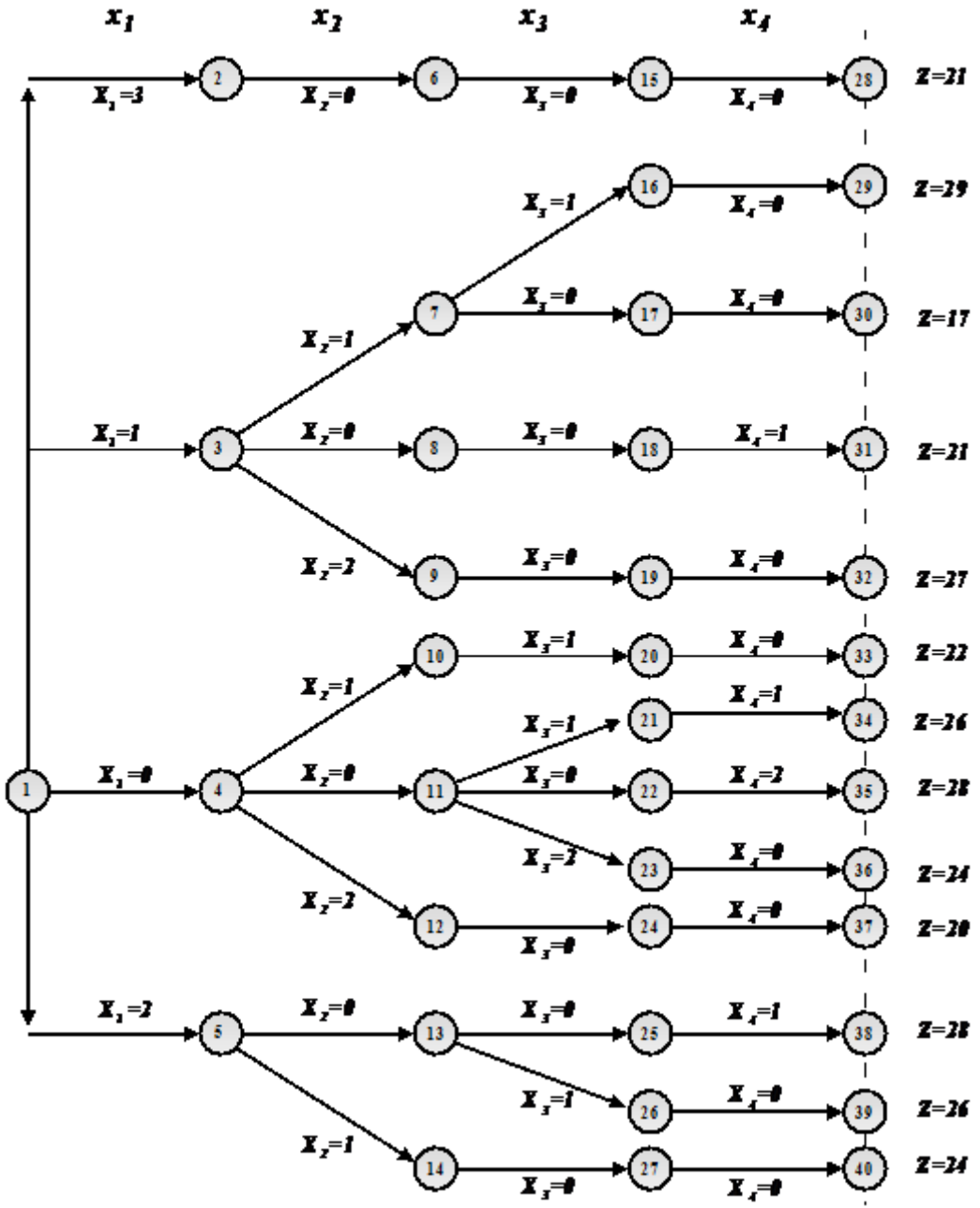
Exemplo

Minimizar $z = 7x_1 + 10x_2 + 12x_3 + 14x_4$

sujeito a :

$$41x_1 + 55x_2 + 60x_3 + 70x_4 \leq 160$$

Árvore de Enumeração



Branch-and-Bound

Maximizar $z = 5x_1 + 8x_2$

sujeito a :

$$x_1 + x_2 \leq 6$$

$$5x_1 + 9x_2 \leq 45$$

$$x_1, x_2 \in Z^+$$

Branch-and-Bound

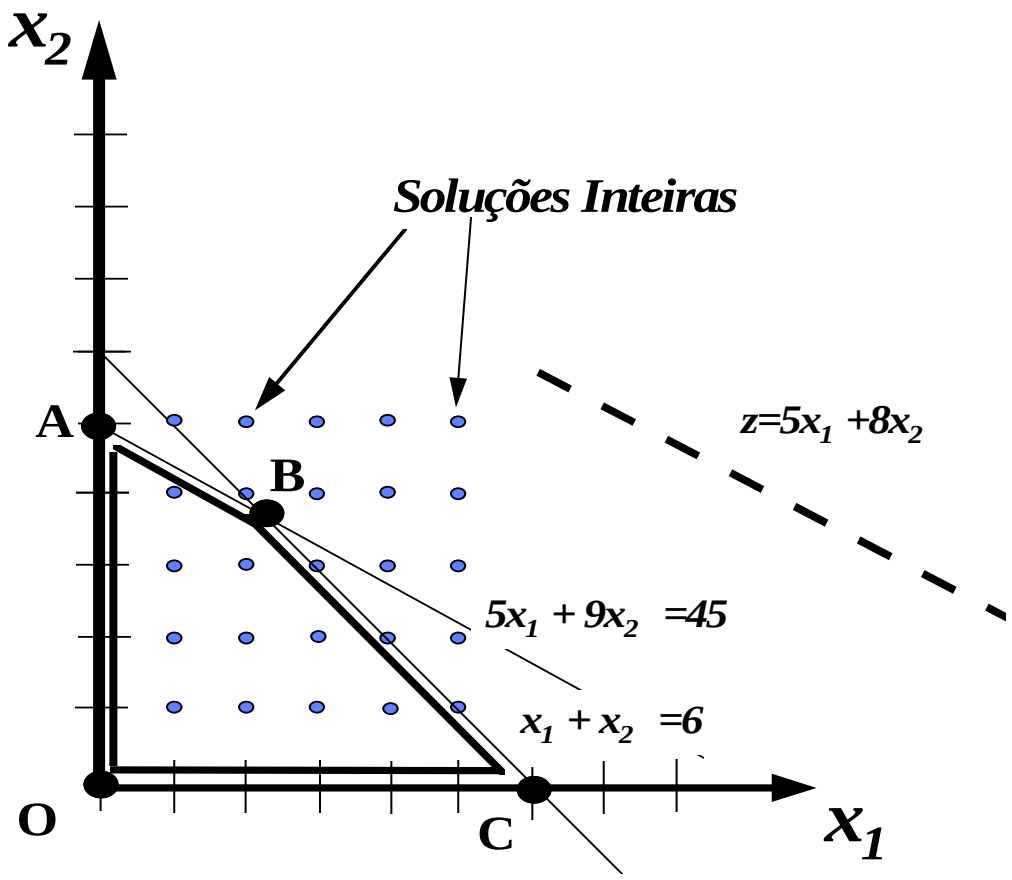
Solução Contínua

$x1 = \frac{9}{4}$ $x2 = \frac{15}{4}$ $Z = 41\frac{1}{4}$

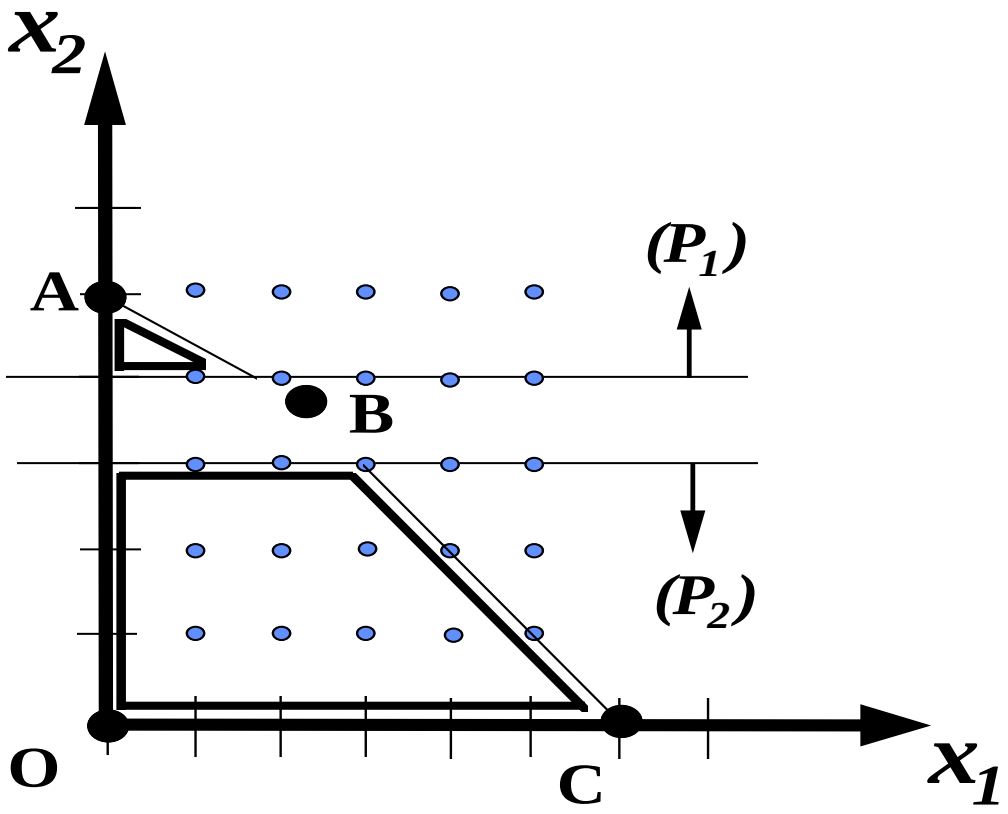
Disjuntiva

$$x_2 \geq \left\lfloor \frac{15}{4} \right\rfloor + 1 \geq 4 \quad \text{ou} \quad x_2 \leq \left\lfloor \frac{15}{4} \right\rfloor \leq 3$$

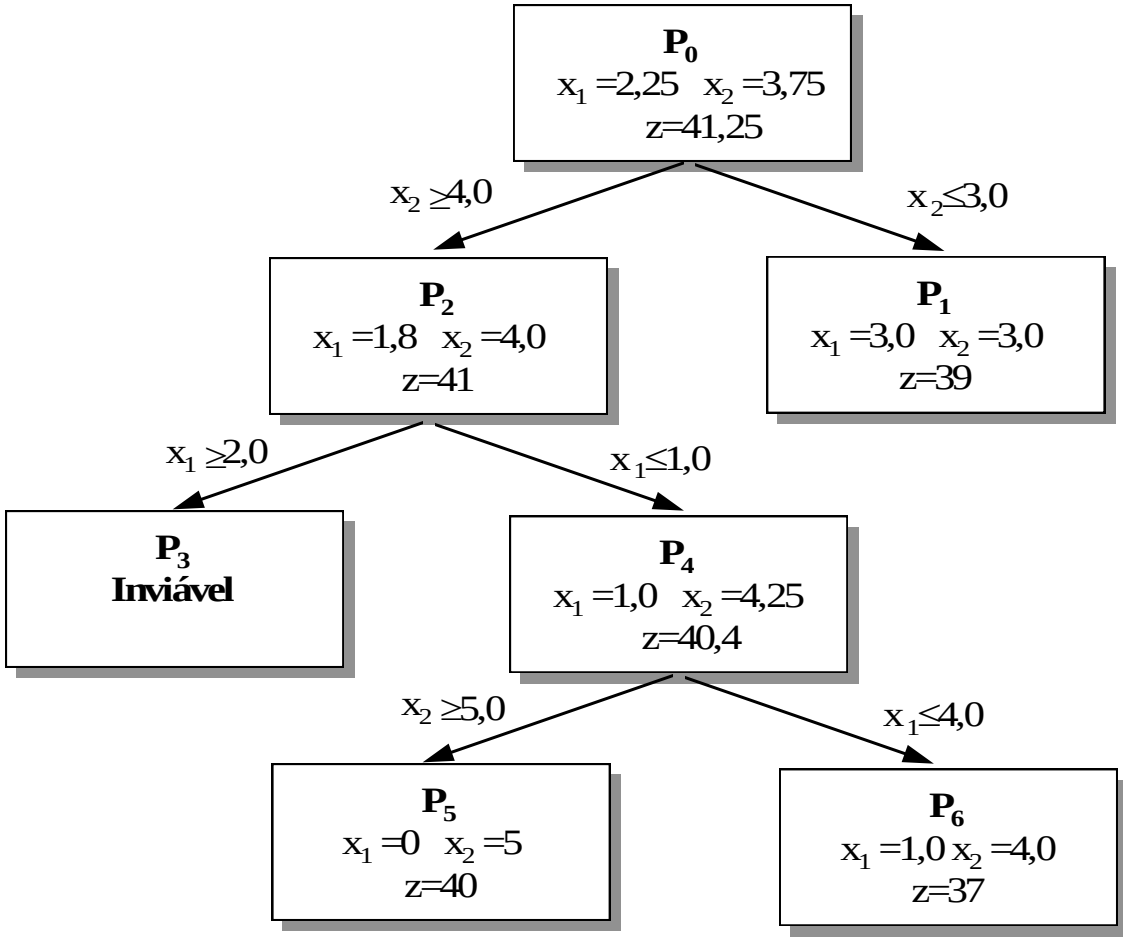
Solução Gráfica



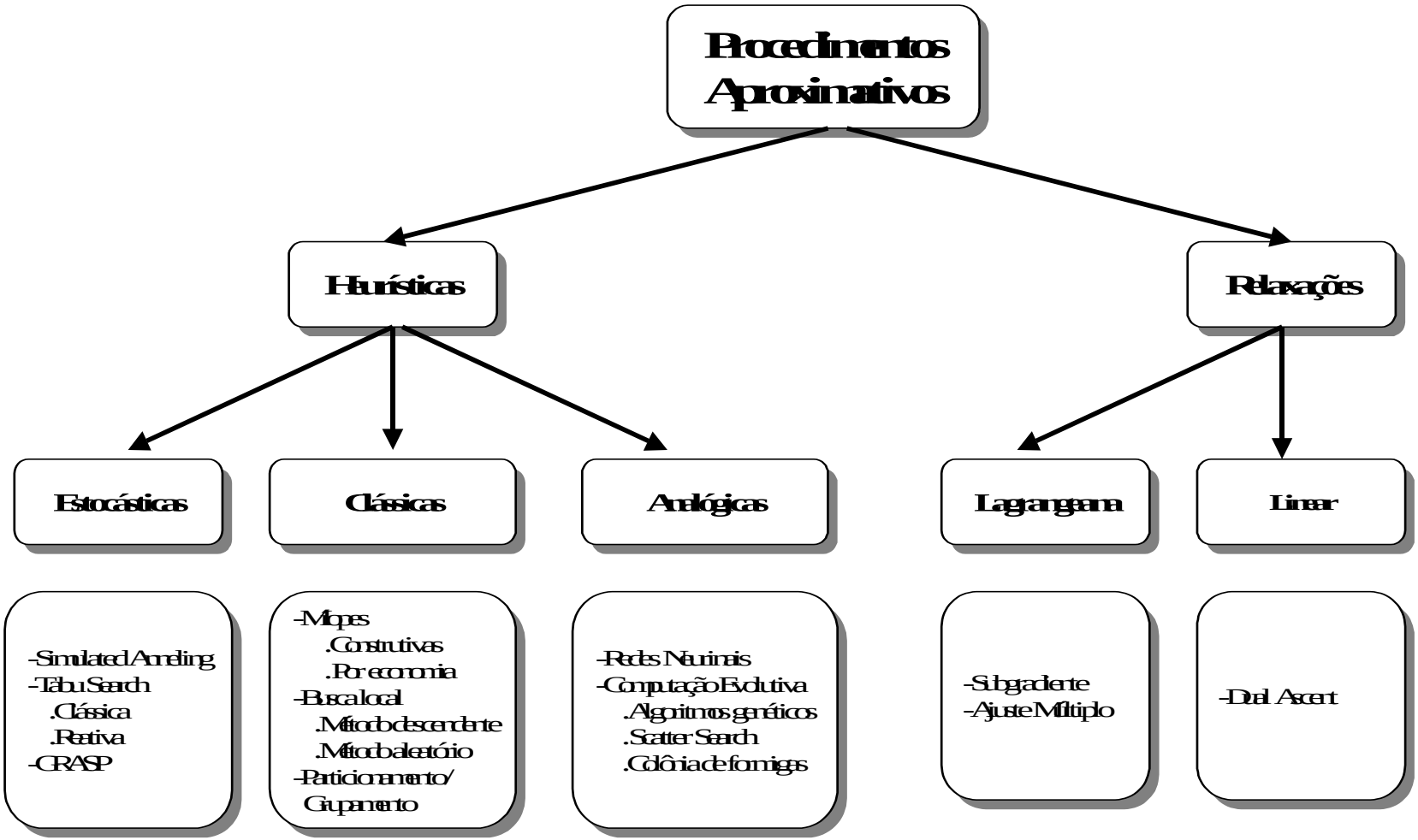
Resultado da Divisão (Branch)



Árvore de Branch



Heurísticas



Problemas de Interesse

O problema de Coloração de Grafos

- Pode ser definido sobre o conjunto dos vértices ou o conjunto das arestas de um grafo;
- Coloração própria: uma coloração própria para um grafo não direcionado G=(V,E) é um mapeamento c:V$\rightarrow\{1, \ldots, k\}$ tal que se $\{u, v\} \in$ E então $c(u) \neq c(v)$.
- Os elementos do conjunto $\{1, \ldots, k\}$ são chamados cores

Coloração de Grafos

- Duas versões usuais para o problema são:

— Determinar se é possível colorir um grafo com um número pré-determinado de cores ou

— Determinar o número cromático (ou o índice cromático) de um grafo G: o menor número de cores de $\{1, \ldots, k\}$ para colorir propriamente o conjunto de vértices (ou de arestas) de um dado grafo G

Exemplos de aplicação

• primeira versão ➡ programação de horários de grades escolares: alocação de n professores a m turmas nos h horários disponíveis na escola.

vértice	⟺	aula de um professor
aresta	⟺	as aulas ligadas pela aresta não podem ser realizadas no mesmo horário
cor	⟺	horário

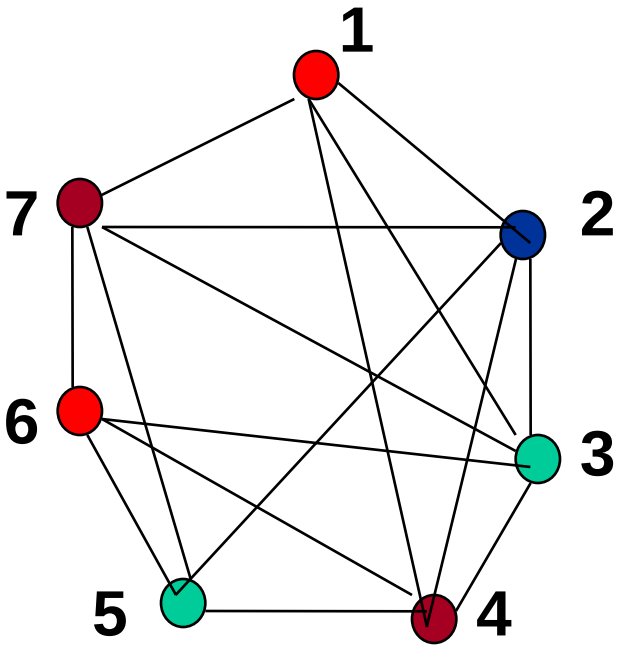
• segunda versão ➡ computação paralela: vértices de uma mesma cor representam processos que podem ser executados em paralelo, pois não possuem dependências.

poucas cores ⟺ processamento rápido

Exemplo

Existem 7 disciplinas. A tabela mostra a existência de alunos em comum: onde há * na célula ij, existe um aluno matriculado na disciplina i e na disciplina j.

	1	2	3	4	5	6	7
1	-	*	*	*	-	-	*
2		-	*	*	*	-	*
3			-	*	-	*	*
4				-	*	*	-
5					-	*	*
6						-	*
7							-

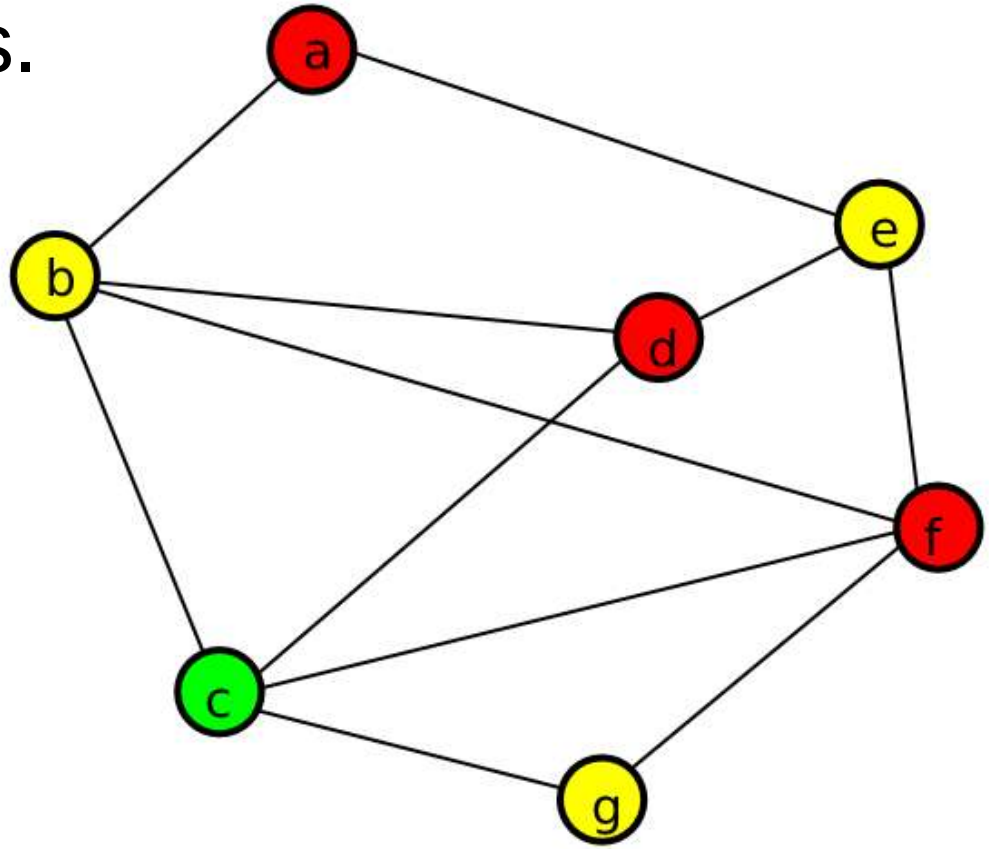


Horário	Disc.
1	1 e 6
2	7 e 4
3	3 e 5
4	2

A matriz é simétrica :a parte abaixo da diagonal principal não foi preenchida

Obter o número cromático

- Particionar o conjunto de vértices em k subconjuntos (mínimo possível) de vértices não adjacentes.



Como obter o número cromático?

NP-Completo

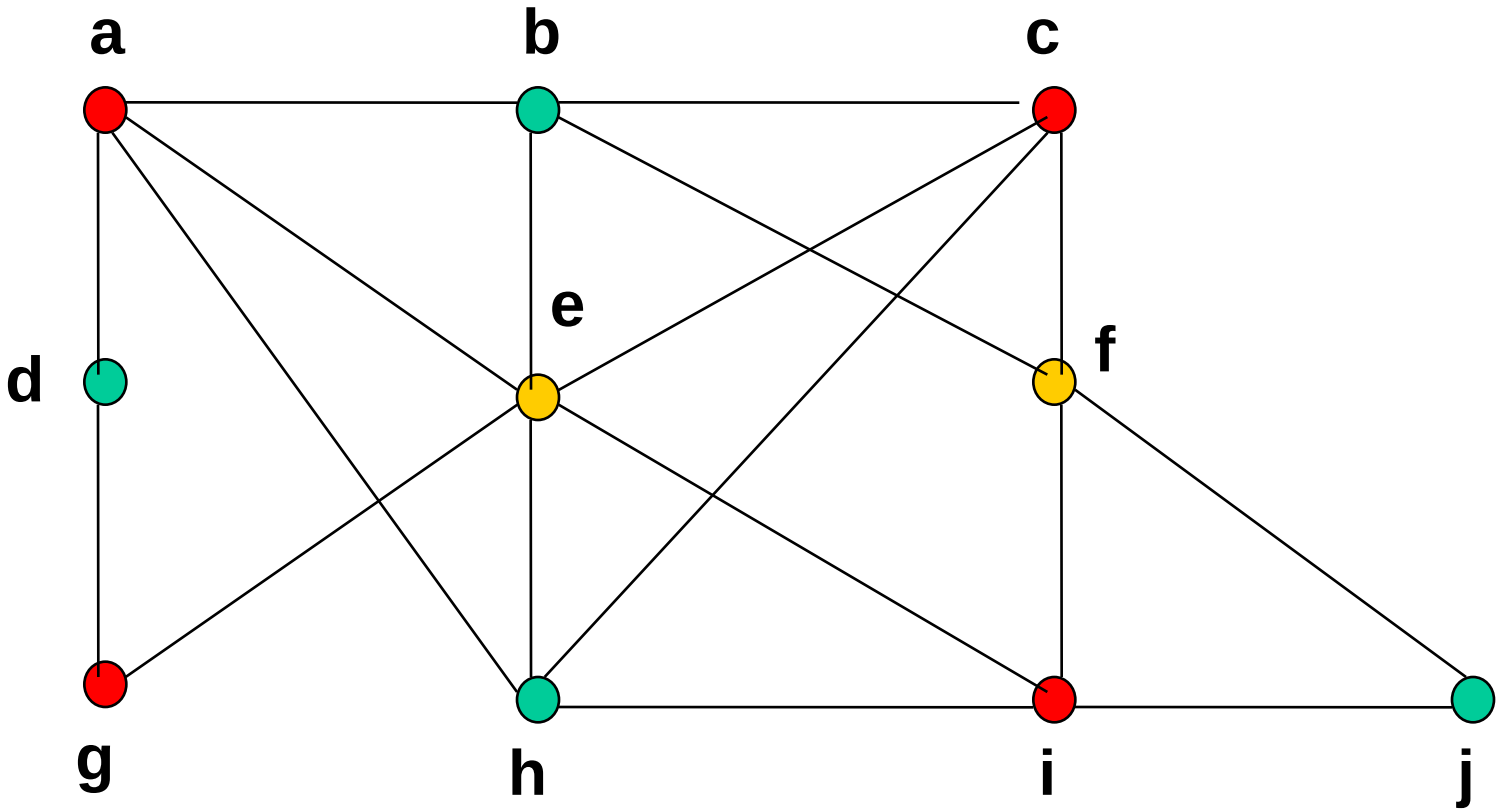
difícil de resolver

heurísticas

nem sempre se garante a obtenção do menor número de cores

Algoritmo de Coloração

```
k=0
Para j=0  até  n-1 faça D = {1, ... , k}
    Para  i=0  até  j-1  faça
       se  v[i]  é adjacente a  v[j] então
          D = D - {cor[v[i]]}
       fim se
    fim Para
    se D não é vazio então
        cor[v[j]] = min D
    senão
        k = k+1
        cor[v[j]] = k
    fim se
fim Para
```


a
b
c
d
e
f
g
h
i
j

O problema de Isomorfismo de Grafos

- O Problema de Isomorfismo de Grafos (PIG) tem sido amplamente estudado nas áreas de química, matemática e computação devido a sua aplicabilidade em vários problemas práticos

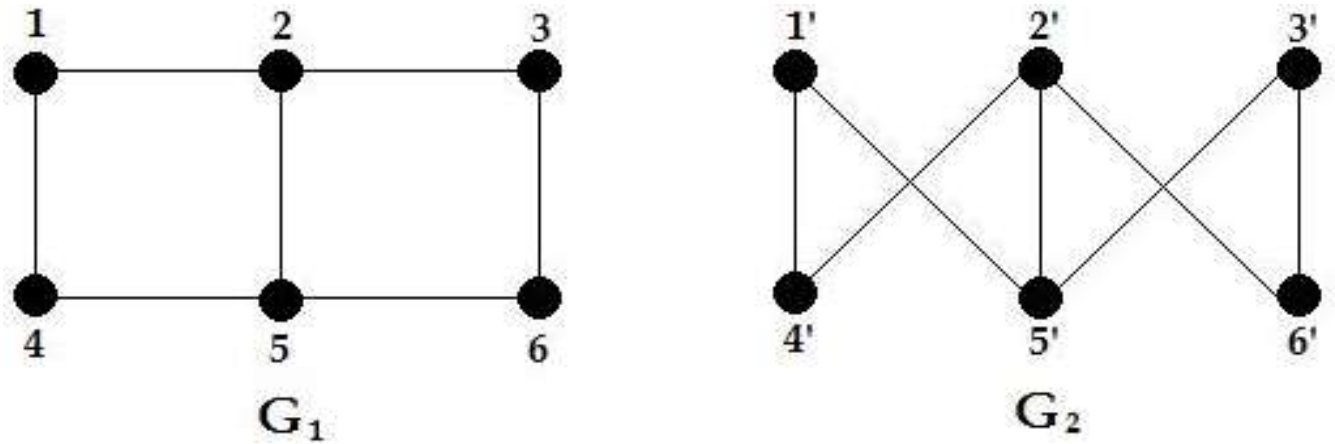
Definição

- Consiste em encontrar uma correspondência biunívoca dos vértices de dois grafos dados, obedecendo as adjacências existentes entre os vértices;

- Mais formalmente: Considere dois grafos $G_1=(V_1,E_1)$ e $G_2=(V_2,E_2)$. Estes grafos são ditos *isomorfos* se existir uma função bijetora $f : V_1 \rightarrow V_2$ onde as seguintes condições são satisfeitas:

  - ✓ Para cada aresta (a,b) de E_1, temos uma aresta (f(a),f(b)) em E_2;
  - ✓ Toda aresta de E_2 tem a forma (f(a),f(b)) para alguma aresta (a,b) de E_1.

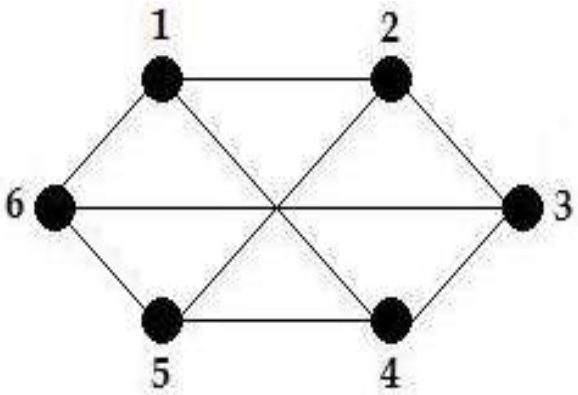
Exemplo de Grafos Isomorfos

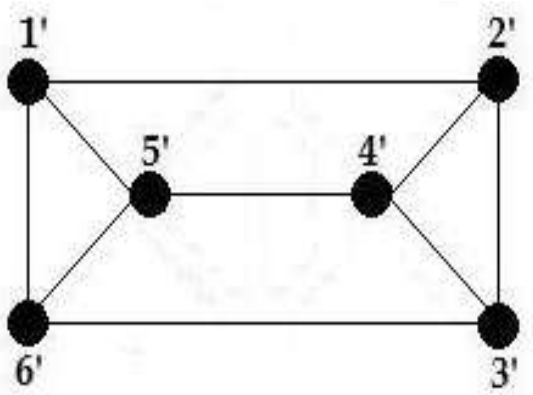


- É possível encontrar uma função bijetora f entre os vértices, f = {(1, 1'), (2, 5'), (3, 3'), (4, 4'), (5, 2'), (6, 6')}, que satisfaz as condições descritas anteriormente, isto é, mantém as características dos vértices em relação ao grau e a conectividade entre eles;

Condições de Isomorfismo

- Para que dois grafos sejam isomorfos, no mínimos as seguintes condições são necessárias:
  - ✓ Possuir o mesmo número de vértices;
  - ✓ Possuir o mesmo número de arestas;
  - ✓ Possuir a mesma seqüência de graus.





Infelizmente, estas condições não são suficientes para afirmar que dois Grafos são Isomorfos!!

Propriedade de Equivalência

- A relação de isomorfismo é uma relação de equivalência:

  - **Reflexiva**: Todo o grafo é isomorfo a si mesmo;

  - **Simétrica**: Se um grafo é isomorfo a um segundo grafo, então também o segundo é isomorfo ao primeiro;

  - **Transitiva**: Se um grafo é isomorfo a um segundo grafo, que por sua vez é isomorfo a um terceiro, então o primeiro é isomorfo ao terceiro.

Complexidade

- 2009: Ashay Dharwadker e John Tevet: o problema de isomorfismo de grafos pertence à classe P.
- Foi proposto um algoritmo polinomial para verificar se dois grafos são isomorfos.

Implementação disponível

http://www.geocities.com/dharwadker/tevet/isomorphism/

Exemplos de aplicação

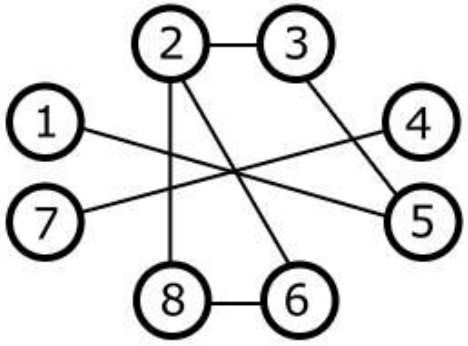
- Reconhecimento de imagens
- Problema de redução de banda de matrizes esparsas

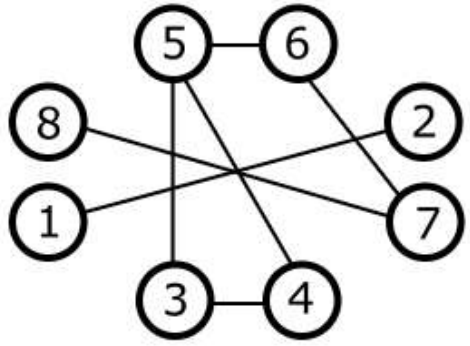
O problema de Redução de Banda de matrizes esparsas

- Para uma dada matriz esparsa simétrica M(nxn), o problema consiste em reduzir a largura de banda B, permutando linhas e colunas de maneira a mover todos os elementos não nulos o mais próximo possível da diagonal.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$





http://ciprian-zavoianu.blogspot.com/2009/01/project-bandwidth-reduction.html

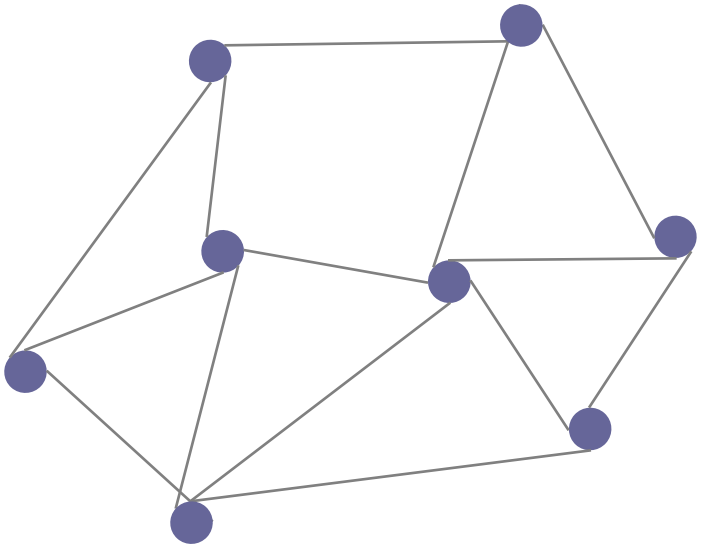
O problema de Particionamento de Grafos

- Sejam G = (V,E) um grafo valorado em vértices e arestas e k um inteiro tal que k >1.

- Deseja-se particionar o conjunto de vértices de um grafo em *k* subconjuntos disjuntos balanceados, de forma que o peso total das arestas com extremidades em diferentes subconjuntos seja minimizado.

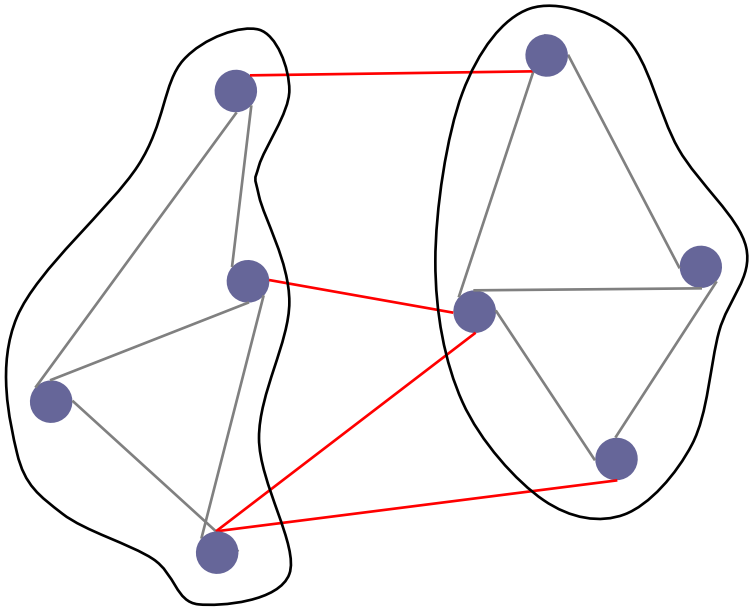
Exemplo

G = (V,E)



2 partições

V1 V2



Corte de arestas: conjunto de arestas cuja remoção de G torna G desconexo, desde que nenhum subconjunto próprio desse conjunto também desconecte

O Problema

- De modo geral, para grafos com pesos associados:
  - Dado um grafo *G = (V, E)* e um número k, deve-se encontrar k subconjuntos V_1, V_2, ... , V_k tal que:

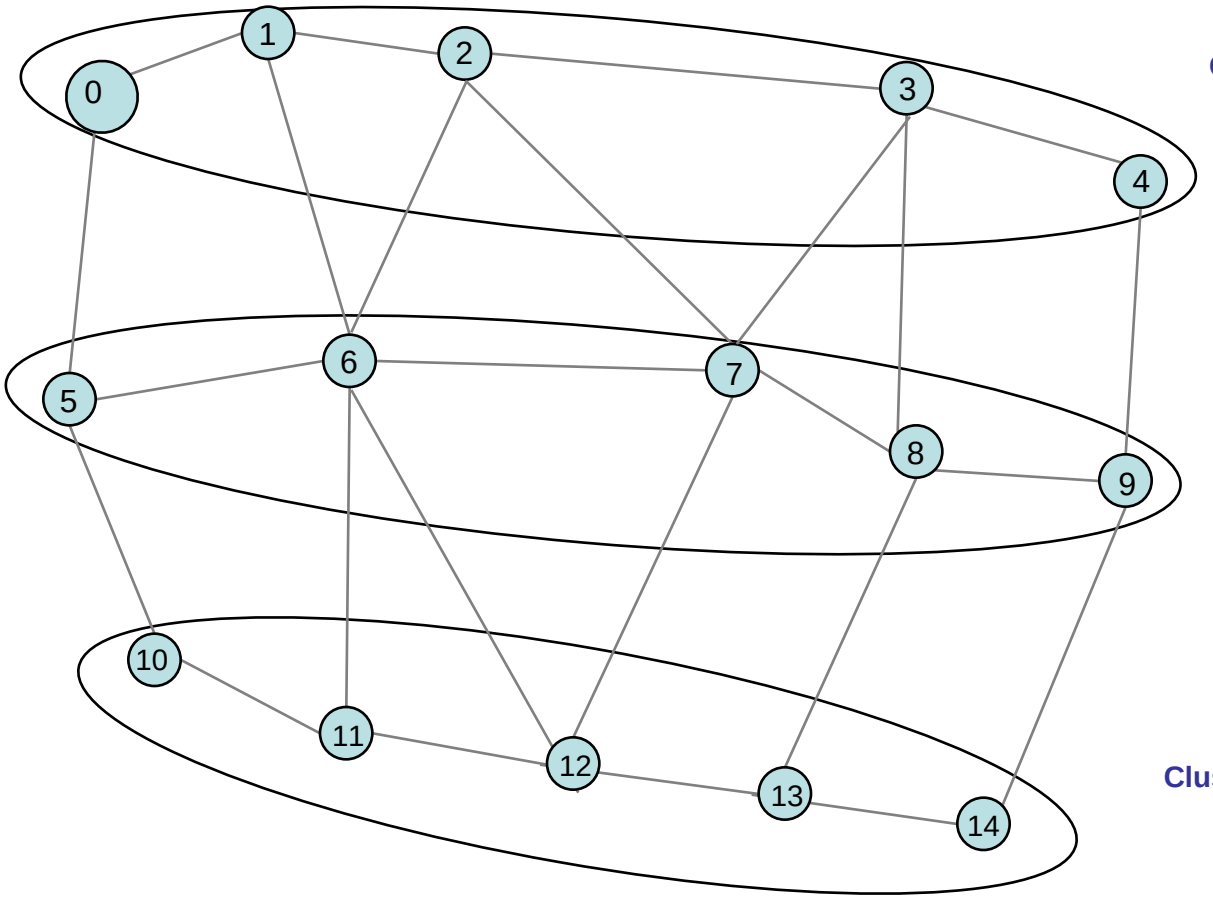
1) $\bigcup_{i=1}^{\bar{k}} V_i = V$ e $V_i \cap V_j = \varnothing$ para todo $i \neq j$;

2) $W(i) \approx W / \bar{k}, \forall i = 1,2,...,\bar{k}$ onde $W(i)$ e W representam, respectivamente, as somas dos pesos dos vértices pertencentes a V_i e V.

3) O corte de arestas, isto é, a soma dos pesos das arestas contidas no corte seja minimizada.

  - Quando k = 2, o problema é referido como Graph Bisection Problem. Para k > 2,o problema é referenciado como k-way Graph Partitioning Problem.

Exemplo



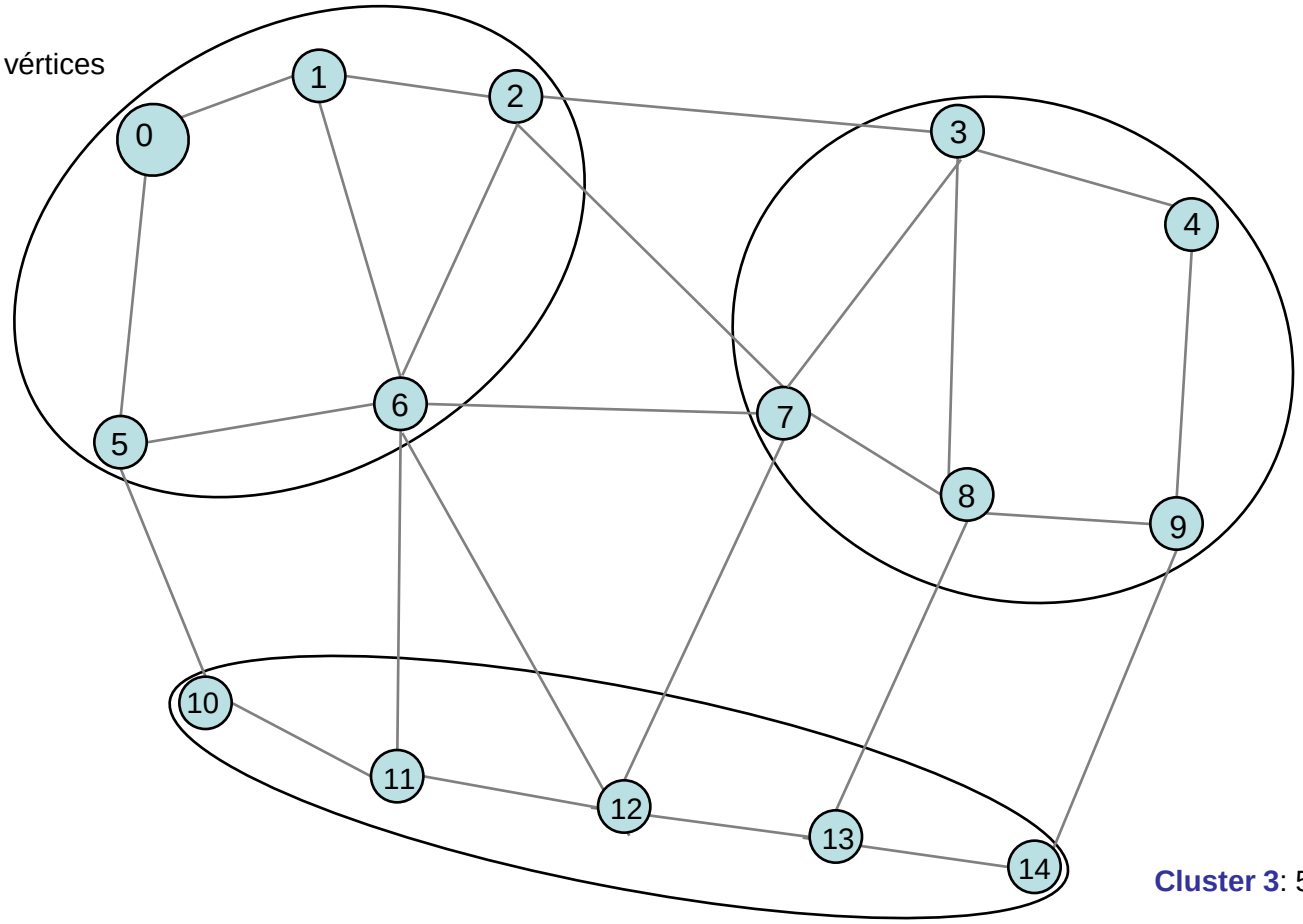
Cluster 1: 5 vértices

Cluster 2: 5 vértices

Cluster 3: 5 vértices

Corte: 13 arestas

Outro Exemplo



Corte: 9 arestas

Formulação Matemática

- Seja $G = (V, E)$ um grafo com um conjunto de vértices V e um conjunto de arestas E. Seja W_{ij} o peso da aresta (i, j) entre os vértices i e j, K o número máximo de clusters k e MaxCard o tamanho máximo de cada cluster ($|V| \leq K.MaxCard$).

- Formulação Padrão

Variáveis:

$v_{ik} = 1$ se o vértice i está no cluster k e 0 caso contrário.

$e_{ijk} = 1$ se a aresta (i,j) pertence ao cluster k e 0 caso contrário.

Maximizar:

$$\sum_{i,j,k} e_{ijk} W_{ij}$$

Sujeito à:

Cada vértice está em apenas um cluster (1)

$$\sum_{k} v_{ik} = 1 \qquad \forall\, i$$

A capacidade dos clusters deve ser obedecida (2)

$$\sum_{i} v_{ik} \leq MaxCard \qquad \forall\, k$$

Linearização da variável quadrática e_{ijk} $(e_{ijk} = v_{ik} * v_{jk})$ (3)

$$e_{ijk} \leq v_{ik}$$

$$e_{ijk} \leq v_{jk}$$

$$e_{ijk} \geq v_{ik} + v_{jk} - 1 \qquad \forall i, j, k$$

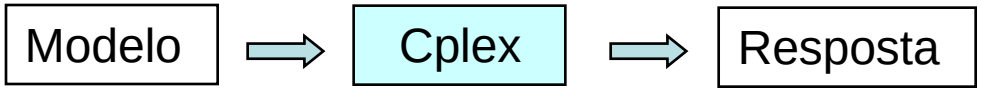
Variáveis binárias de decisão:

$$v_{ik}$$

Métodos de resolução

- Métodos Exatos
  - Branch and bound

Solver Cplex



Complexidade: NP-Difícil

Métodos de Resolução

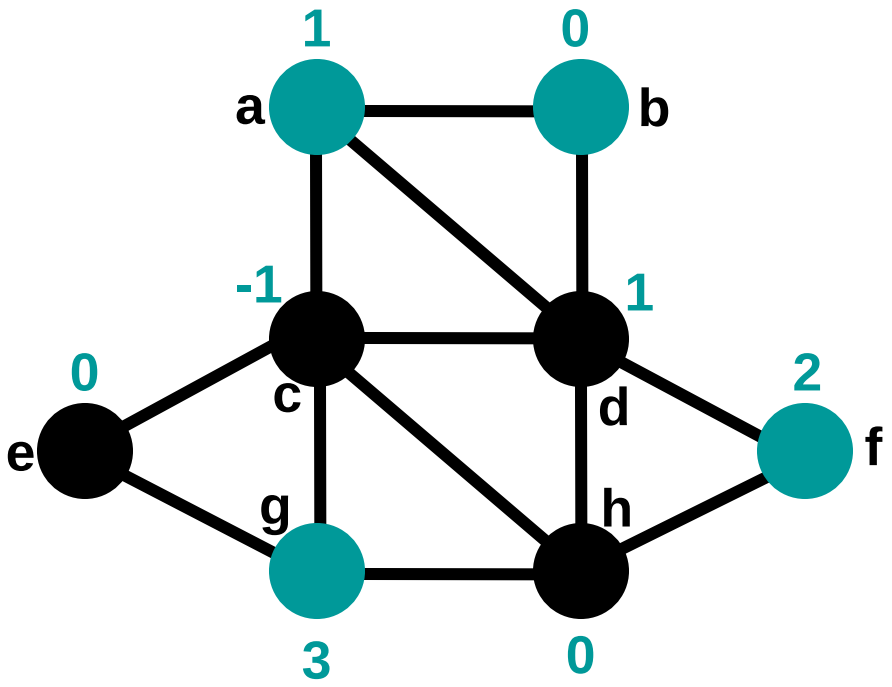
- Métodos heurísticos
  - Spectral Partitioning
  - Kernighan and Lin
  - Fiduccia-Mattheyses
  - Multilevel Graph Partitioning
  - Genetic Algorithms
  - Entre outros

Fiduccia-Mattheyses: Algoritmo

- passo 1
- computar os ganhos de todos os vértices
- passo 2
- **i** = 1
- achar um vértice para troca **c(i)**
- passo 3
- bloquear o vértice **c(i)** e atualizar os ganhos dos vizinhos
- passo 4
- se não existem mais vértices bloqueados,
- incrementar i
- achar um novo vértice para troca
- passo 5
- achar uma sequencia de movimentos que levem ao ganho máximo
- se não tem mais que uma sequencia de movimentos, escolher o particionamento que proporciona o melhor corte
- passo 6
- aplica os movimentos
- volta ao passo 2 até que um máximo seja atingido

Fiduccia-Mattheyses: Exemplo

- **Partição 1**: Vértices em azul
- **Partição 2**: Vértices em preto
- Tamanho inicial do corte: 8 arestas
- Ganho: redução do nº de arestas ao migrar um vértice de uma partição para outra
- O ganho inicial de cada vértice está mostrado em azul

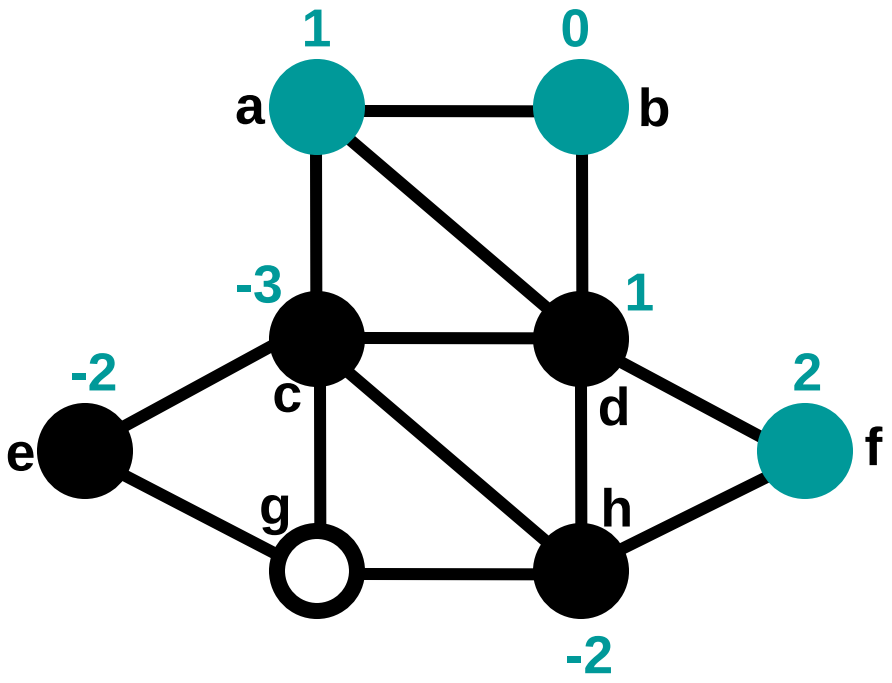


Nós movidos (e tamanho do corte depois) :

nenhum (8);

Fiduccia-Mattheyses: Exemplo

O nó na Partição 1 com maior ganho é g. Vamos tentar movê-lo para a Partição2 e recomputar o ganho de seus vizinhos.



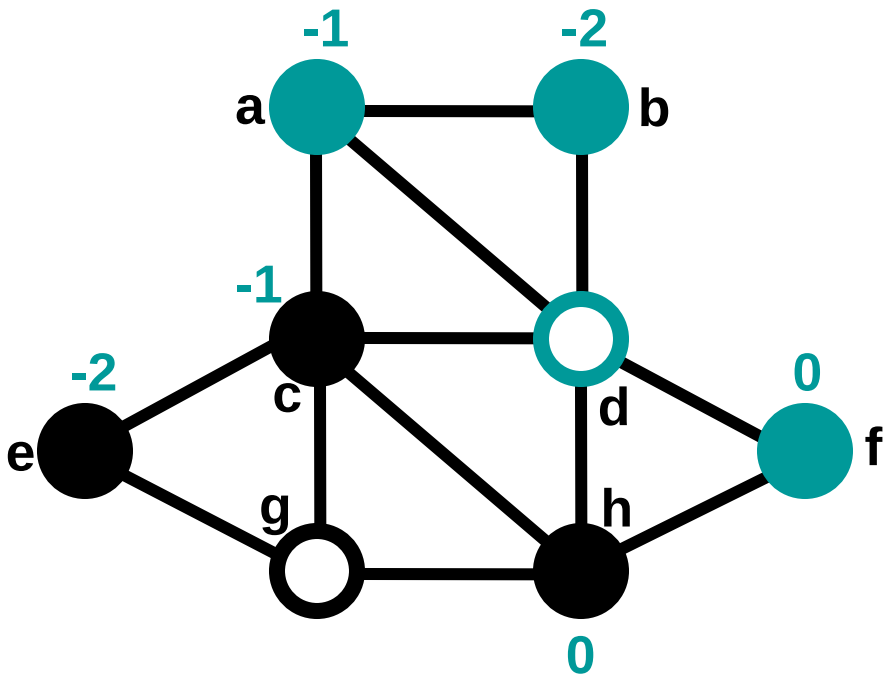
Nós movidos (e tamanho do corte depois) :

nenhum (8); g,

Fiduccia-Mattheyses: Exemplo

O nó na Partição2 com maior ganho é d. Tentamos movê-lo para a Partição1 e recomputar o ganho de seus vizinhos.

Depois da primeira tentativa de troca, o tamanho do corte é 4.

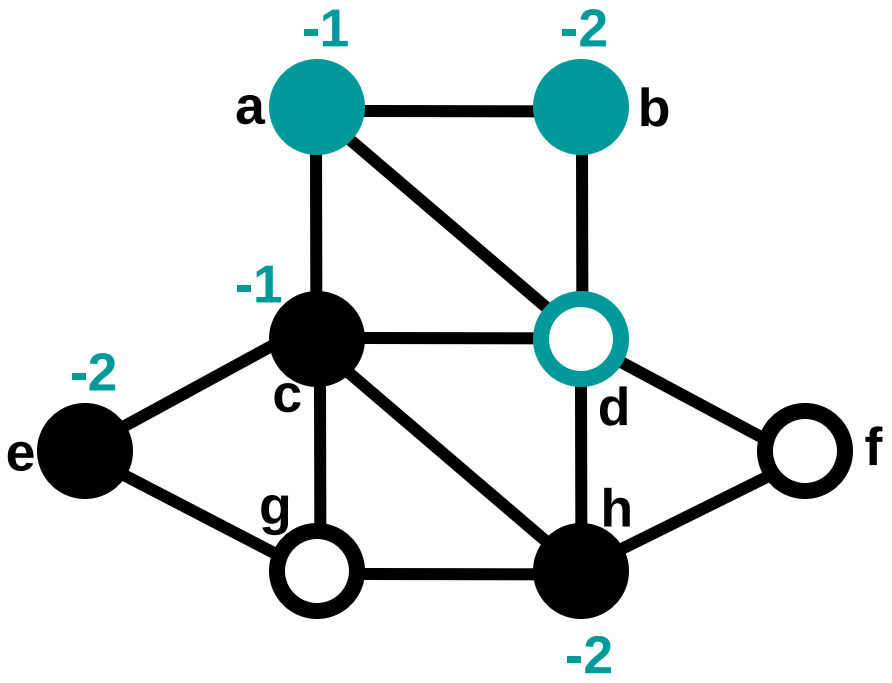


Nós movidos (e tamanho do corte depois) :

nenhum (8); g, d (4);

Fiduccia-Mattheyses: Exemplo

O nó ainda não movido na Partição1 com maior ganho é f.



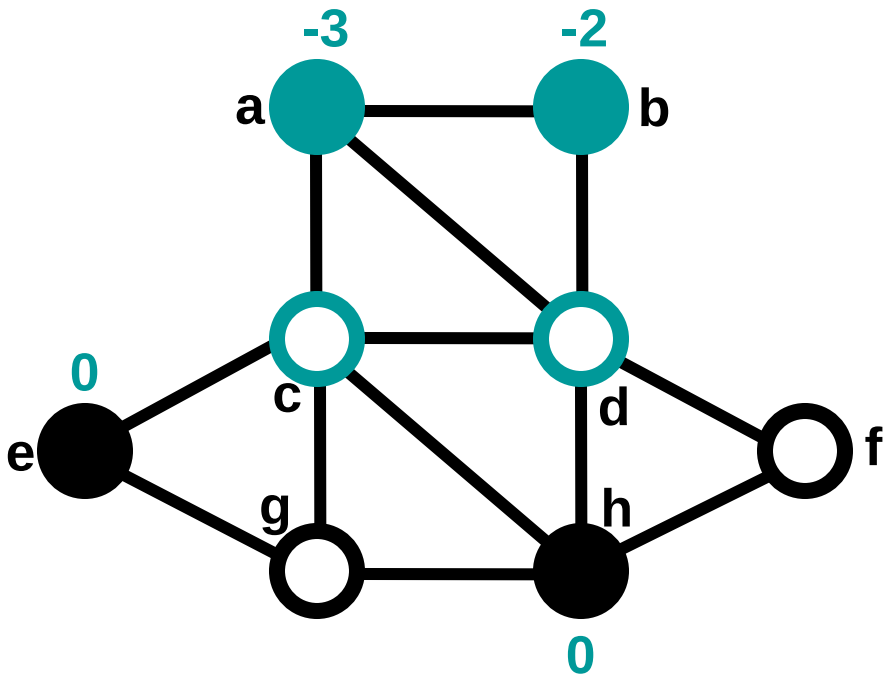
Nós movidos (e tamanho do corte depois) :

nenhum (8); g, d (4); f

Fiduccia-Mattheyses: Exemplo

O nó ainda não movido na Partição2 com maior ganho é c.

Após essa tentativa de troca, o tamanho do corte é 5.

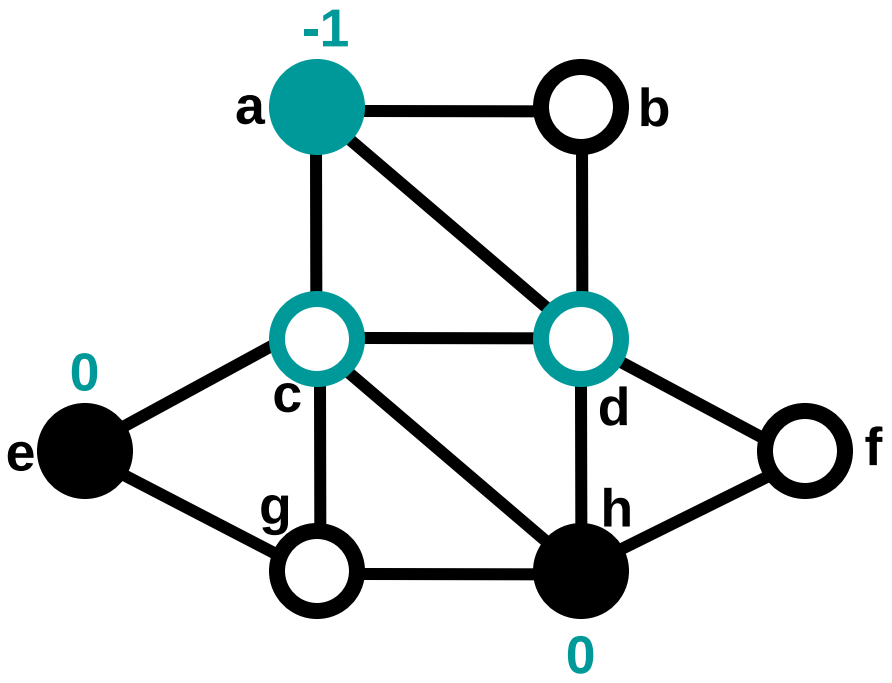


Nós movidos (e tamanho do corte depois) :

nenhum (8); g, d (4); f, c (5);

Fiduccia-Mattheyses: Exemplo

O nó ainda não movido na Partição1 com maior ganho é b.

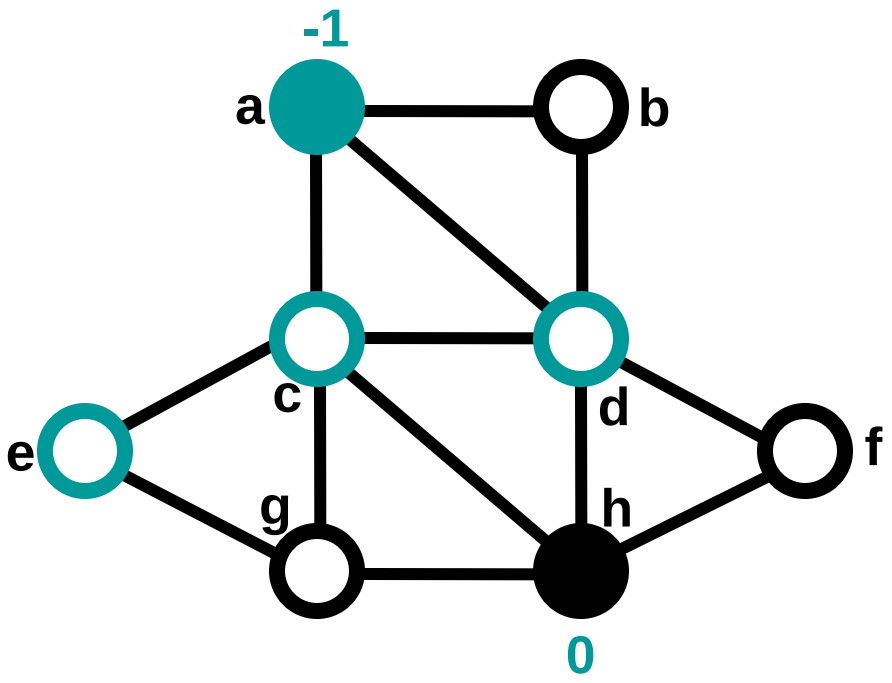


Nós movidos (e tamanho do corte depois) :

nenhum (8); g, d (4); f, c (5); b

Fiduccia-Mattheyses: Exemplo

Existe um empate entre nós na Partição2. Escolhemos um e tentamos movê-lo para a Partição1. Todos os seus vizinhos são nós movidos, portanto não precisa recomputar os ganhos.



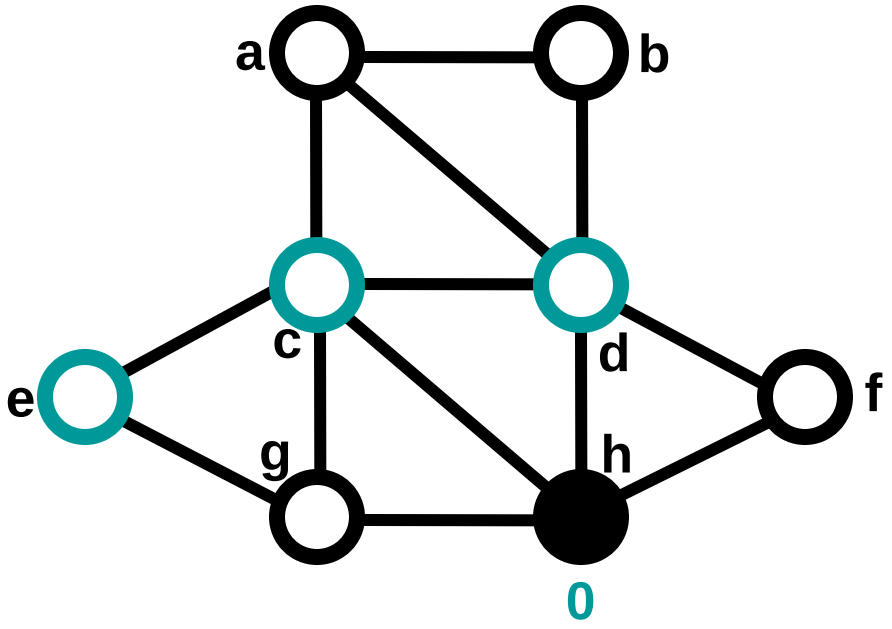
Após essa tentativa de troca, o tamanho do corte é 7.

Nós movidos (e tamanho do corte depois) :

nenhum (8); g, d (4); f, c (5); b, e (7);

Fiduccia-Mattheyses: Exemplo

O nó ainda não movido na Partição1 com maior ganho é a.



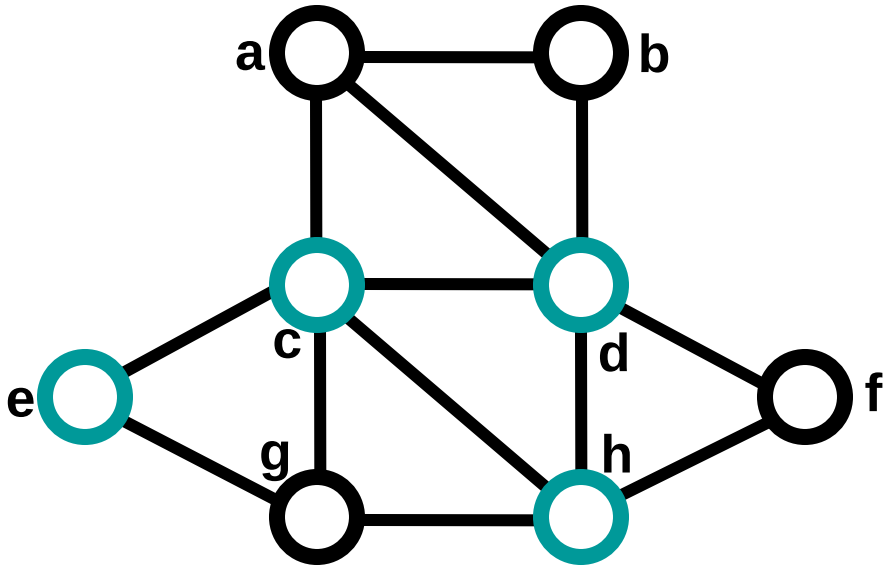
Nós movidos (e tamanho do corte depois) :

nenhum (8); g, d (4); f, c (5); b, e (7); a

Fiduccia-Mattheyses: Exemplo

O nó ainda não movido na Partição2 com maior ganho é h.

O tamanho de corte na ultima tentativa de troca é 8.



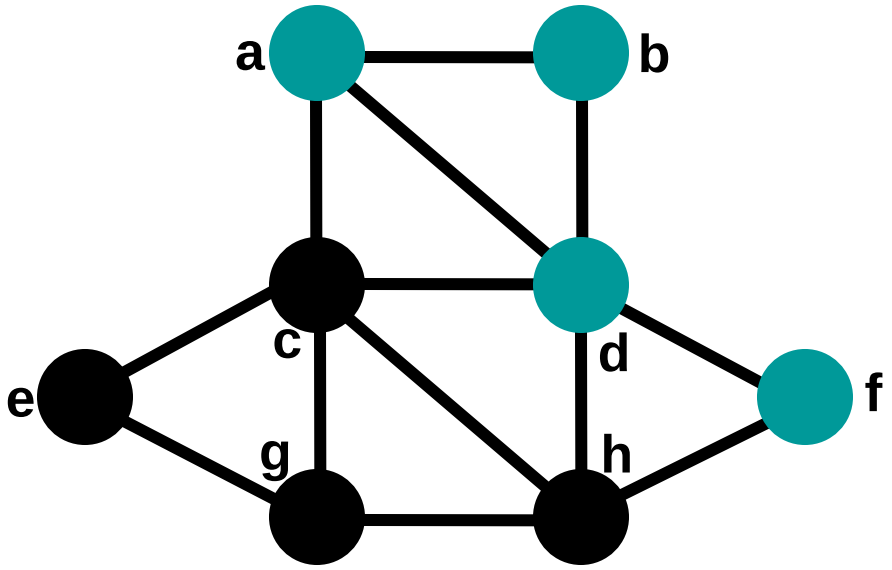
Nós movidos (e tamanho do corte depois) :

nenhum (8); g, d (4); f, c (5); b, e (7); a, h (8)

Fiduccia-Mattheyses: Exemplo

Depois que tentamos mover todos os nós, percorremos a sequência de trocas e fixamos a troca que resultou menor corte. Então, fazemos essa troca permanente e deletamos todas as tentativas posteriores.

Isso é o final do primeiro passo de melhoria.



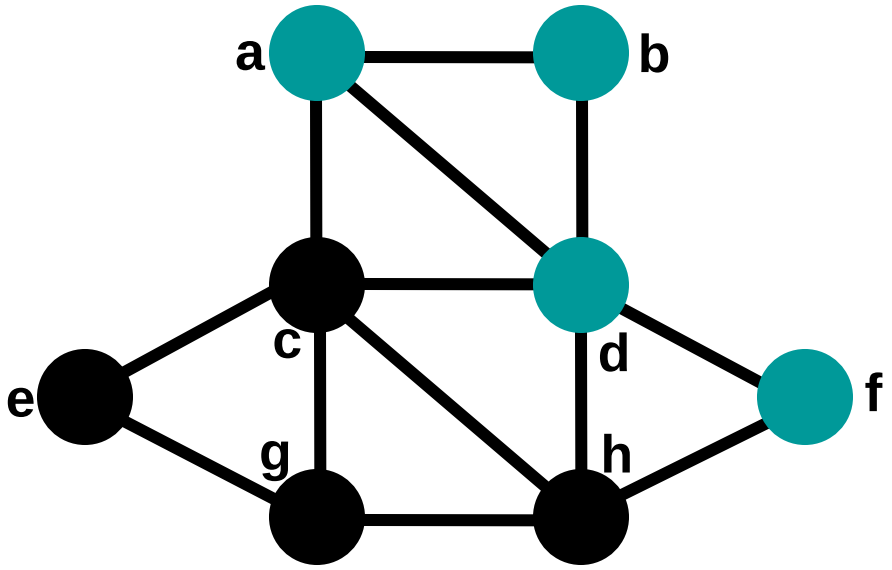
Nós movidos (e tamanho do corte depois) :

nenhum (8); **g, d (4);** f, c (5); b, e (7); a, h (8)

Fiduccia-Mattheyses: Exemplo

Fazemos o processo novamente começando com novo tamanho de corte igual a 4.

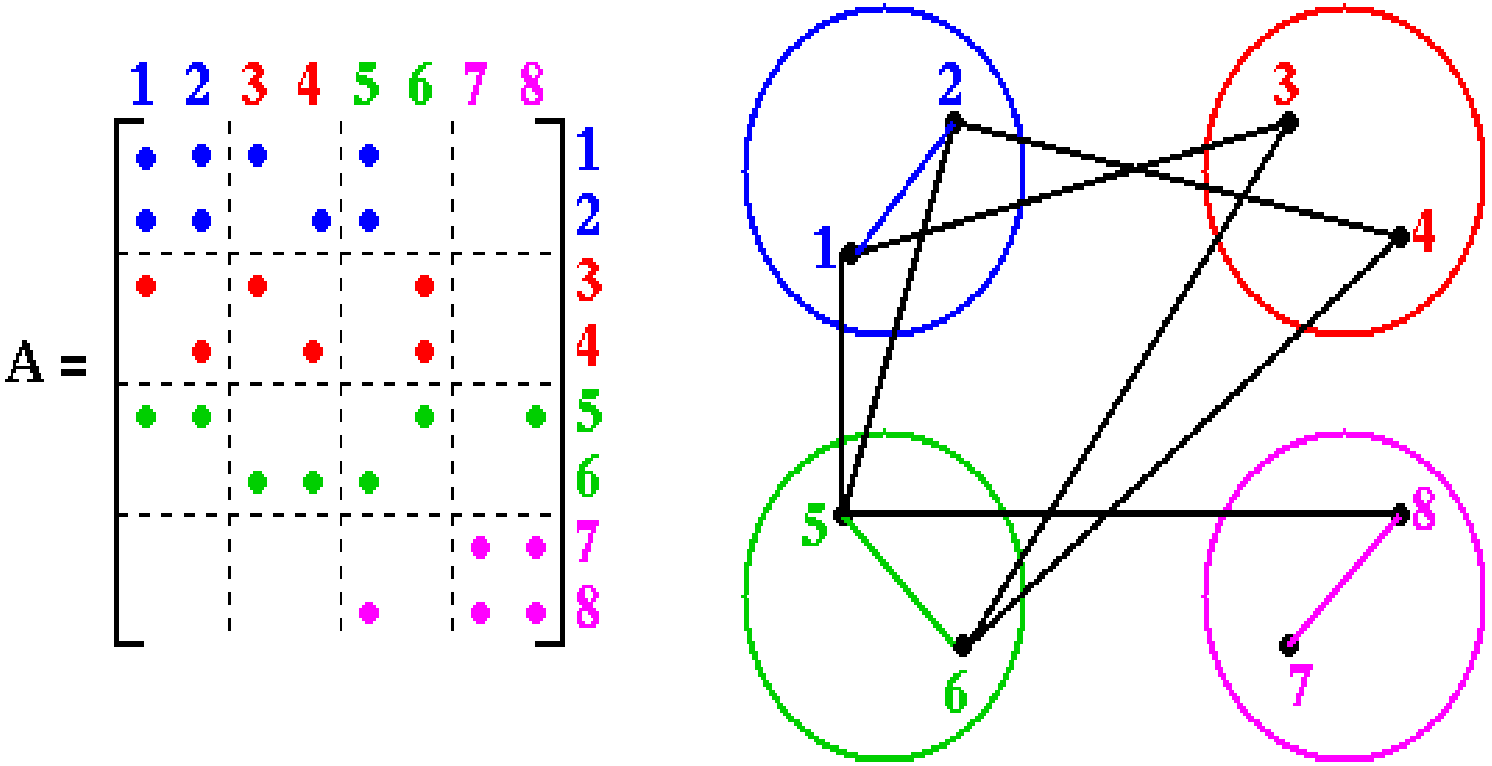
Neste caso, o segundo passo de melhoramento não melhora a solução e o algoritmo então para com tamanho de corte igual a 4.



Exemplos de aplicação

- Redes: dividir a rede em pequenos clusters para maximizar a quantidade de comunicações locais e minimizar a conectividade entre os clusters.
- Computação paralela: um problema de partição do conjunto de vértices de um grafo em p subconjuntos, onde o grafo representa uma malha de elementos finitos e p é o número de processadores disponíveis. Tal partição precisa levar em conta o balanceamento da carga de trabalho dos processadores bem como a a minimização dos custos de comunicação entre processadores.
- Localização de facilidades: localização de k hospitais em uma cidade de forma que ninguém more o mais longe que o necessário do hospital mais próximo

Partitioning a Sparse Symmetric Matrix



http://www.eecs.berkeley.edu/~demmel/cs267/lecture18/lecture18.html