Uma introdução à Otimização Combinatória

Slides adaptados de material didático dos professores Adriana Alvim (UniRio), Celso Ribeiro (UFF) e Cristina Rangel (UFES)

labotim
Laboratório de Otimização - UFES

UNIVERSIDADE FEDERAL DO ESPIRITO SANTO
DOCETE OMNES GENTES

Sumário

- Problemas de interesse
- Otimização contínua X otimização combinatória
- Problemas combinatórios
- Complexidade computacional
- Noções básicas sobre teoria da complexidade
- Como tratar problemas combinatórios difíceis?

Problemas de interesse...

Que tipo de problema estamos interessados em resolver?

Figura 1 - Esquema de otimização de abastecimento da pilha pelas minas, com os respectivos teores

Fonte – imagens: Google Imagens

Objetivos típicos...

- aproveitar melhor os materiais no processo de produção;
- otimizar o tempo para realização de ações e operações;
- transportar mais materiais por melhores rotas;
- diminuir chances de se obter prejuízos, aumentar lucros e diminuir gastos;
- entre outros...

Nestes problemas, não nos contentamos em ter apenas uma das soluções possíveis, mas aquela(s) que também otimize os objetivos de interesse...

Em um problema de otimização estamos interessados em encontrar a melhor solução dentre todas que satisfazem a um conjunto de condições estipuladas para o problema. Dependendo da natureza das variáveis, o problema é classificado como de Otimização Contínua (quando as variáveis são contínuas) ou de Otimização Combinatória (quando as variáveis são discretas). Em um problema de otimização combinatória, procuramos por um objeto de um conjunto finito (ou possivelmente enumerável).

Exemplo

- Otimização Contínua
- Encontrar o quadrilátero de menor perímetro com área de $1000m^2$

$$\text{Min } p = 2x + 2y$$

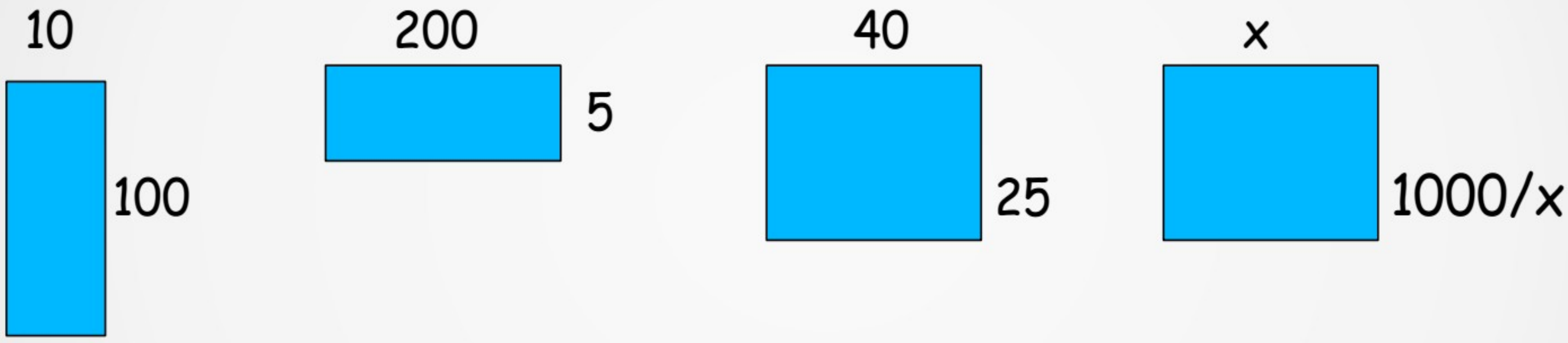
$$\text{sujeito à } xy = 1000$$

$$x >= 0 \text{ e } y >= 0$$

- Não é um problema de Programação Linear. Por quê?

Otimização Contínua X Otimização Combinatória

Soluções possíveis em valores reais:



p=220   p=410   p=130   p=2x+2000/x

Otimização Contínua X Otimização Combinatória

- Resolver analiticamente através da derivada
  - $y = 1000/x$
  - $p = 2x+2000/x$
  - $p' = 2 - 2000/x^2$
  - Ponto de mínimo quando $p' = 0$
  - $x^2 = 1000$, $x = \sqrt{1000}$
    - então $x \sim 31{,}623$ com $y \sim 31{,}623$
- perímetro $p \sim 126{,}5$

Otimização Contínua X Otimização Combinatória

- **Otimização Combinatória**
- Problema: Min $p = 2x+2y$

  sujeito à $xy = 1000$

  $x \geq 0$ e $y \geq 0$

Valores discretos possíveis.

Por que esses valores?

x	y	perímetro
1	1000	2002
2	500	1004
...		
20	50	140
25	40	130
40	25	130
....		
1000	1	2002

- Em uma grande quantidade de problemas, procura-se soluções que façam o melhor uso dos recursos disponíveis.
- Tipicamente não se busca apenas uma solução, mas a melhor que otimize os objetivos de interesse!

Problemas combinatórios

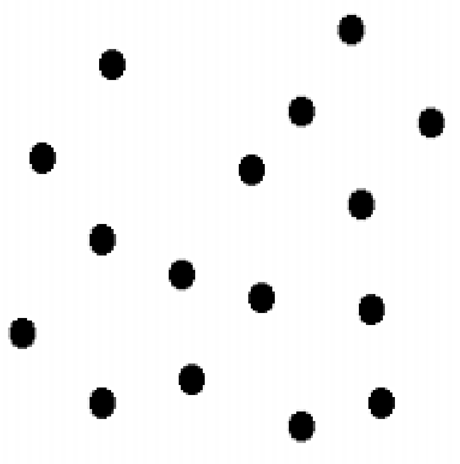
- Problemas combinatórios aparecem em diversas áreas da ciência da computação e em outras disciplinas onde métodos computacionais são aplicados
  - encontrar caminho mínimo de ida e volta (TSP)
  - roteamento
    - de pacotes de dados na internet
    - de veículos (VRP)
  - planejamento (planning)
  - escalonamento (scheduling)
    - programação das tarefas em uma linha de produção

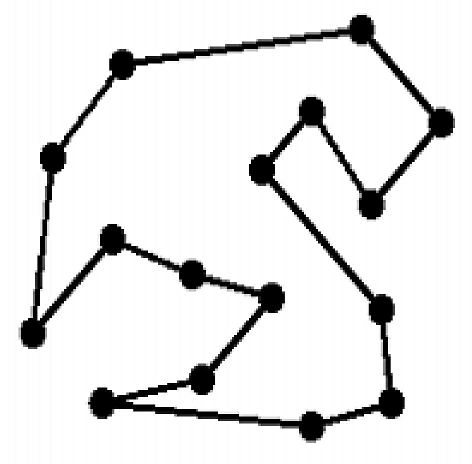
Problemas combinatórios

- Problemas combinatórios envolvem encontrar um agrupamento, uma ordenação, ou uma atribuição de um conjunto discreto e finito de objetos que satisfazem condições dadas
- Soluções candidatas são
  - combinações de componentes da solução
  - podem ser encontradas durante a tentativa de gerar uma solução
  - não precisam satisfazer todas as condições dadas

Problemas combinatórios

- Exemplo
  - Dados: conjunto de pontos no espaço Euclideano
  - Objetivo: encontrar o caminho de ida e volta de tamanho mínimo





Problemas combinatórios

- um caminho ligando todos os pontos corresponde a uma sequência de pontos

  (= atribuição dos pontos a uma sequência de posições)

  - componente da solução: segmento de caminho consistindo em dois pontos que são visitados um imediatamente depois do outro
  - solução candidata: um caminho de ida e volta
    - uma abordagem particular poderia considerar um caminho que visitasse uma cidade mais de uma vez
  - solução: um caminho de ida e volta de tamanho mínimo

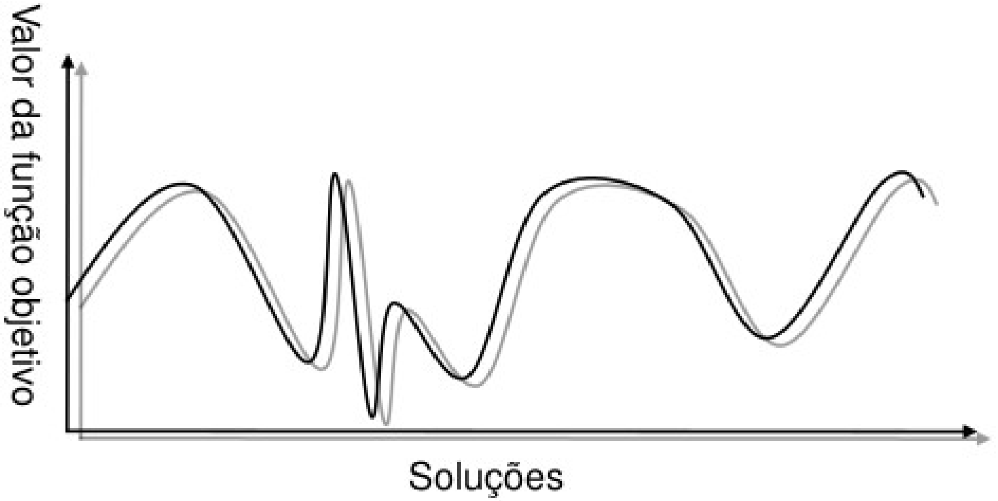
Problemas combinatórios

- Problema *versus* instância do problema
  - Problema: dados qualquer conjunto de pontos X, encontre um caminho de ida e volta de tamanho mínimo
  - Solução: algoritmo que encontra caminhos de ida e volta de tamanho mínimo para qualquer X
  - Instância do problema: dados um conjunto específico de pontos P, encontre um caminho de ida e volta de tamanho mínimo
  - Solução: caminho de ida e volta de tamanho mínimo para P
- Tecnicamente, problemas podem ser formulados como conjuntos de instâncias do problema

Problemas combinatórios

- Problema de escalonamento pode ser considerado um problema de atribuição onde
  - componentes da solução são os eventos a serem escalonados
  - os valores atribuídos aos eventos correspondem a hora em que eles ocorrem
- Tipicamente existe uma quantidade enorme de soluções candidatas
- Para a maioria dos problemas combinatórios, o espaço de potenciais soluções para uma dada instância do problema é pelo menos exponencial no tamanho da instância

Espaço de soluções de um problema



Complexidade computacional

- Questão fundamental
  - O quão difícil é resolver um dado problema computacional?
- Conceitos importantes
  - Complexidade de tempo de um problema Π
  - Complexidade de tempo de pior caso

Complexidade computacional

- Complexidade de tempo de um problema Π
  - tempo de computação necessário para resolver uma dada instância π de Π usando o algoritmo mais eficiente para Π
  - o tempo de execução de um algoritmo será definido em função dos dados de entrada
    - não dependerá da natureza dos dados de entrada mas sim do que se convenciona chamar de tamanho da instância
  - por exemplo, no processo de ordenar os elementos de um vetor
    - o fator determinante será o número de elementos a serem ordenados e não sua natureza

Complexidade computacional

- Complexidade de tempo de pior caso
  - complexidade de tempo no pior caso considerando todas as instâncias do problema de um dado tamanho
  - tipicamente medida como uma função no tamanho da instância
  - negligenciando constantes
  - notação O

Complexidade computacional

- Problemas para os quais existem algoritmos polinomiais são considerados tratáveis
  - função de complexidade é $O(p(n))$, onde $p(n)$ é um polinômio
- Enquanto que aqueles que comprovadamente não podem ser resolvidos por algoritmos polinomiais são considerados intratáveis
  - tipicamente os algoritmos propostos para esta classe de problemas são de natureza enumerativa (pesquisa exaustiva) e possuem complexidade exponencial
  - função de complexidade é $O(c^n)$, $c > 1$

Complexidade computacional

- Exemplos (complexidade polinomial): considere-se um vetor de n elementos inteiros
  - determinar o menor ou maior elemento: $O(n)$
  - busca sequencial por um elemento específico: $O(n)$
  - busca binária por um elemento (vetor ordenado): $O(\log n)$
  - ordenar o vetor por um método "simples": $O(n^2)$
  - ordenar o vetor por um método "elaborado": $O(n \log n)$

Complexidade computacional

- <u>Exemplos (complexidade polinomial)</u>: considera-se duas matrizes quadradas $n \times n$
  - calcular a soma das duas matrizes: $O(n^2)$
  - calcular o produto das duas matrizes: $O(n^3)$

Complexidade computacional

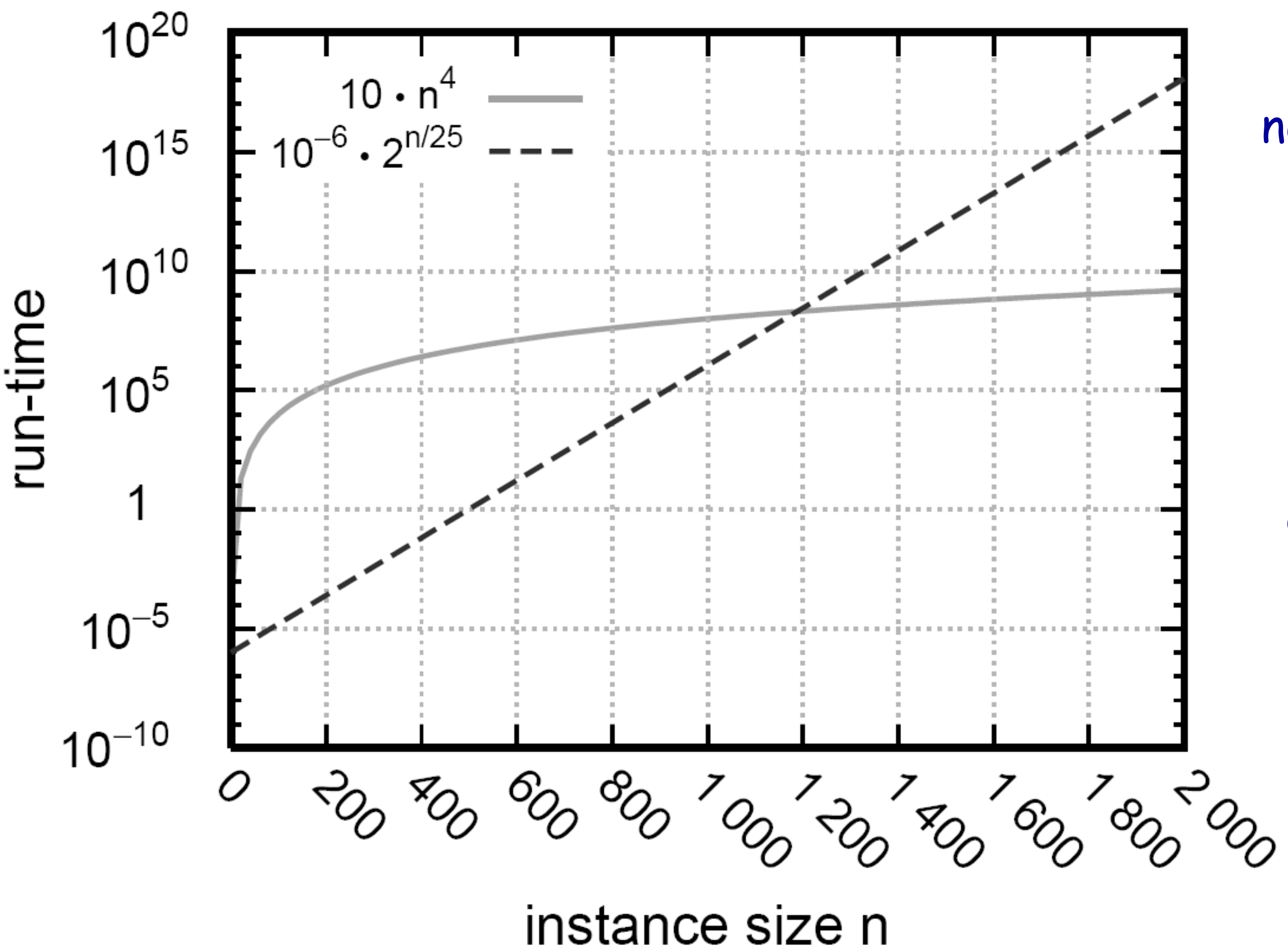
- Entretanto...
  - a distinção entre algoritmos polinomiais eficientes e algoritmos exponenciais ineficientes possui várias exceções
    - um algoritmo 2^n é mais rápido do que um algoritmo n^5 para $n \leq 20$
  - existem algoritmos exponenciais que são muito úteis na prática
    - o algoritmo Simplex para programação linear é exponencial, entretanto executa muito rápido na prática

Complexidade computacional

- Entretanto...
  - na prática os algoritmos polinomiais tendem a ter grau 2 ou 3 no máximo e não possuem coeficientes muito grandes
    - n^{100} ou $10^{99} n^2$ EM GERAL NÃO OCORREM

Complexidade computacional

- Exemplo: crescimento polinomial vs exponencial

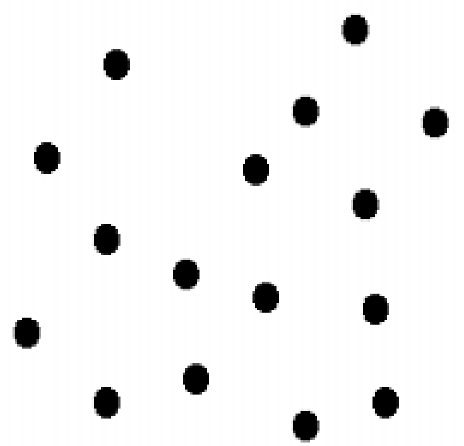


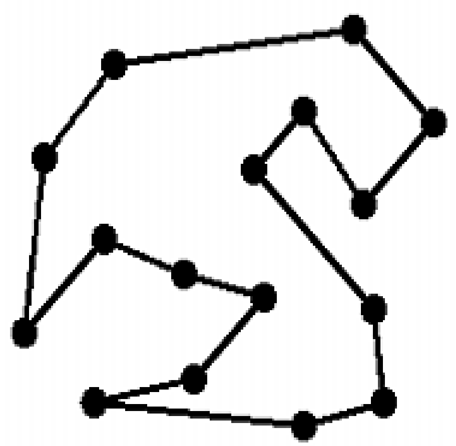
nem sempre algoritmo polinomial é melhor

neste exemplo, para instâncias de tamanho menor do que 1187, o algoritmo exponencial é melhor do que o polinomial

Complexidade computacional

- Exemplo: Problema do caixeiro viajante (PCV)
  - dados um conjunto de n cidades, e uma distância para cada par de cidades, encontre o caminho mais curto ligando todas as cidades, sem visitar nenhuma cidade duas vezes





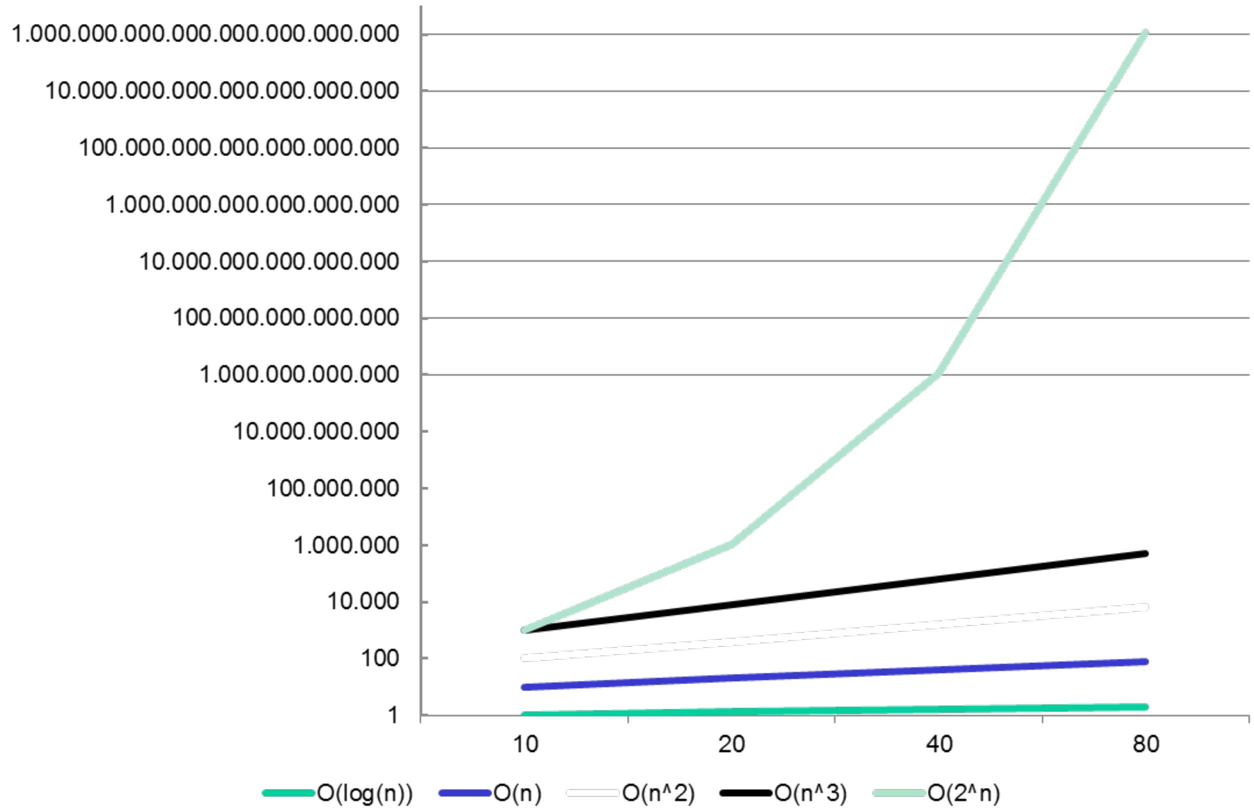
Complexidade computacional

- Qual é o número de soluções?
  - como a cidade de partida é arbitrária existem (n-1)! soluções viáveis,
    - conjunto de todas as permutações de (n-1) elementos
    - se considerarmos que a distância entre cada par de cidades é a mesma, não importa a direção, teremos (n-1) !/2

Complexidade computacional

- Há $(n-1)!$ rotas possíveis e a distância total percorrida em cada rota envolve n adições
  - logo o número total de adições é $n!$
- Suponha um problema com 50 cidades
  - o número de adições seria $50! \approx 10^{64}$
- Em um computador que executa 10^9 adições por segundo, o tempo total para resolver o problema com 50 cidades seria maior do que 10^{45} séculos só para executar as adições
- Por este motivo (tempo computacional cresce exponencialmente com o tamanho do problema)
  - enumeração completa é impossível!

Complexidade computacional



Complexidade de Problemas

- Classes de Problemas Combinatórios
  - Problemas de decisão
    - Existe uma determinada estrutura satisfazendo certa propriedade?
    - Resultado
      - "sim" ou "não"
  - Problemas de otimização
    - Dentre todas as estruturas satisfazendo determinada propriedade, obter aquela que otimiza certa função de custo
    - Resultado
      - uma solução viável ótima

Problema do Caixeiro viajante

- Problema de decisão
  - Entrada: Um conjunto de cidades $N = \{1, ..., n\}$, uma distância c_{ij} para cada par de cidades $(i,j) \in N$ e um inteiro k
  - Problema: Existe uma "rota" para todas as cidades em N, que inicie e termine no mesmo lugar, cujo comprimento seja menor ou igual a k?
- Problema de otimização
  - Entrada: Um conjunto de cidades $N = \{1, ..., n\}$, uma distância c_{ij} para cada par de cidades $(i,j) \in N$
  - Problema: Encontre uma rota de comprimento mínimo

Teoria da complexidade

- Algoritmos determinísticos
  - o resultado de cada operação é definido de forma única
  - dada uma configuração inicial, um algoritmo determinístico sempre chega a mesma configuração final

Teoria da complexidade

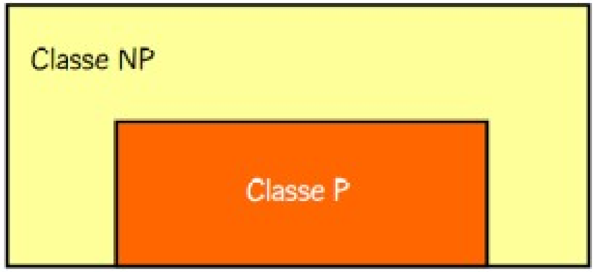
- Algoritmo não-determinístico
  - é capaz de escolher uma dentre as várias alternativas possíveis a cada passo
  - o algoritmo é capaz de adivinhar a alternativa que leva à solução
    - escolha correta (*reasonable guessing*)
  - resultado não é unicamente definido, ainda que limitado a um conjunto de soluções

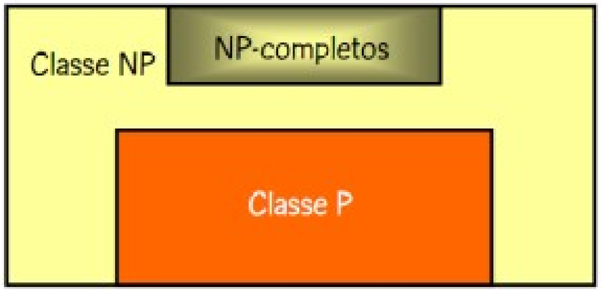
Teoria da complexidade

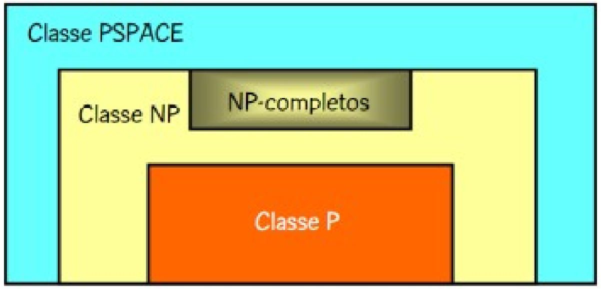
- Classe P
  - conjunto de todos os problemas de decisão que podem ser resolvidos por algoritmos determinísticos em tempo polinomial
- Classe NP
  - conjunto de todos os problemas de decisão que podem ser resolvidos por algoritmos não-determinísticos em tempo polinomial
- Classe EXP
  - problemas de decisão para os quais existe algoritmo exponencial

Teoria da complexidade

Visão simplificada do "mundo" dos problemas de decisão





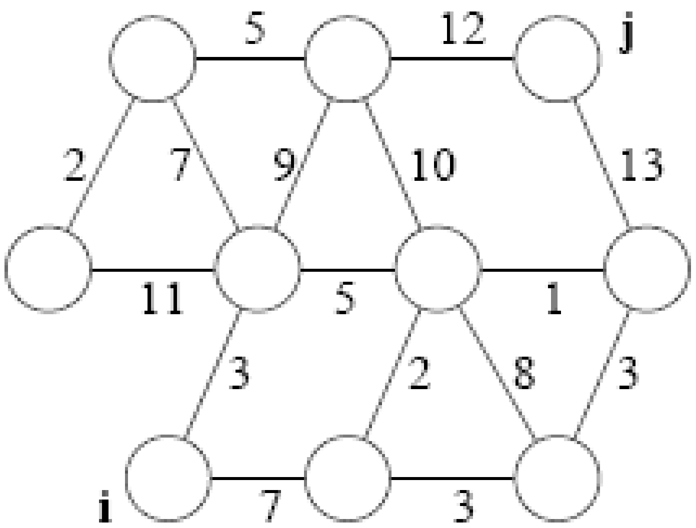


Teoria da complexidade

- Exemplos de problemas para os quais não se conhece um algoritmo de solução de complexidade polinomial
  - problema do caixeiro viajante
  - empacotamento de objetos em caixas (bin packing)
  - escalonamento de tarefas em processadores
  - coloração de grafos
  - projetos de redes de computadores
  - escalonamento de funcionários em turnos de trabalhos

Teoria da complexidade

- Alguns problemas combinatórios podem ser resolvidos de forma eficiente
  - Problema do Caminho Mínimo
    - dados um grafo com peso nas arestas e dois vértices i e j, encontre um caminho de i até j de peso mínimo



    - há um algoritmo eficiente com complexidade de tempo quadrática em relação ao número de vértices
    - algoritmo de Dijkstra

Como tratar problemas combinatórios difíceis?

- Como tratar na prática os problemas NP-completos?

1. Métodos exatos de complexidade super-polinomial (branch-and-bound, cortes, enumeração, programação dinâmica): se for necessário e se o porte dos problemas assim o permitir.
2. Processamento paralelo: clusters e grids permitem acelerações significativas na prática, utilizando recursos computacionais básicos e com custo reduzido.

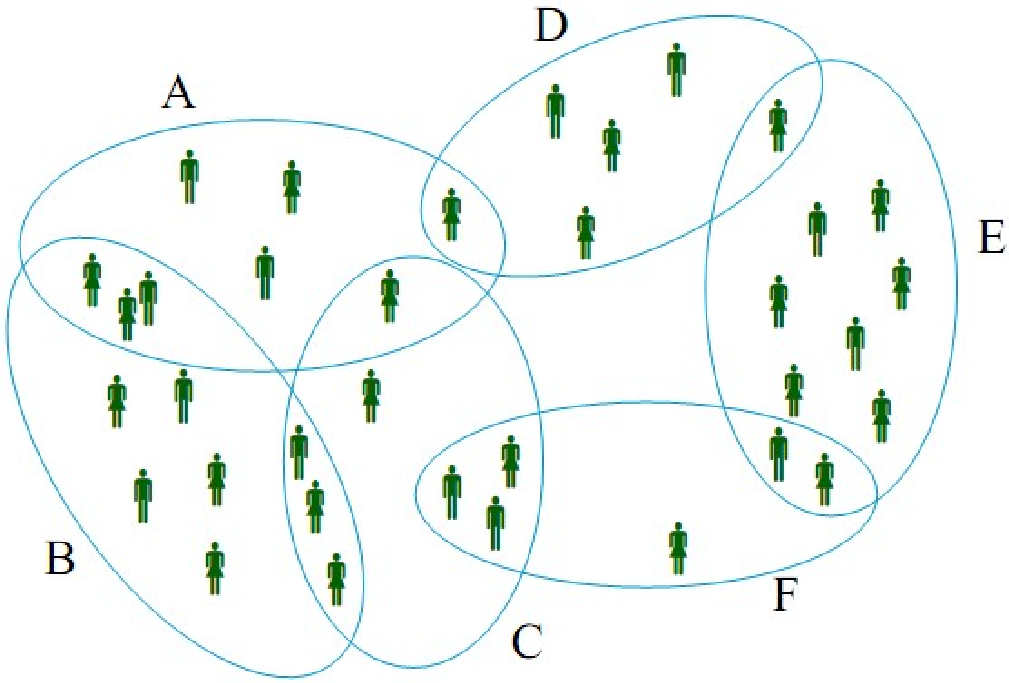
- Como tratar na prática os problemas NP-completos?

3. Heurísticas: qualquer método aproximado projetado com base nas propriedades estruturais ou nas características das soluções dos problemas, com complexidade reduzida em relação à dos algoritmos exatos e fornecendo, em geral, soluções viáveis de boa qualidade (sem garantia de qualidade)
    - Métodos construtivos
    - Busca local
    - Metaheurísticas

- Avanços no estudo e desenvolvimento de heurísticas buscam:
  - Resolver problemas maiores
  - Resolver problemas em tempos menores
  - Obter melhores soluções
- Heurísticas e metaheurísticas permitem resolver problemas de grande porte em tempos realistas, fornecendo sistematicamente soluções ótimas ou muito próximas da otimalidade:
  - Exemplo: problema do caixeiro viajante com milhões de cidades

- Problema:
  - Alunos cursam disciplinas.
  - Cada disciplina tem uma prova.
  - Há um único horário em que são marcadas provas em cada dia.
  - Provas de duas disciplinas diferentes não podem ser marcadas para o mesmo dia se há alunos que cursam as duas disciplinas.
- Montar um calendário de provas:
  - satisfazendo as restrições de conflito e ...
  - ... minimizando o número de dias necessários para realizar todas as provas.

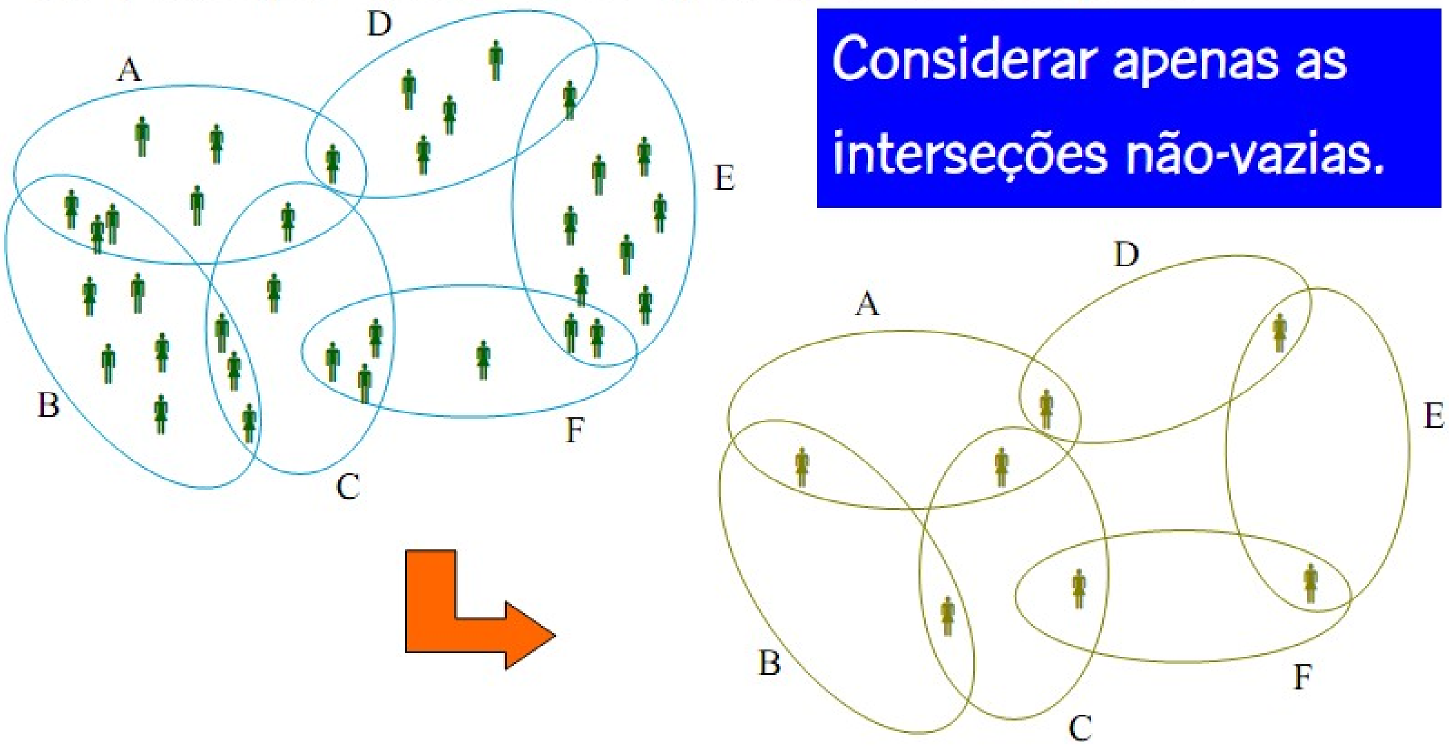
- Modelagem por conjuntos:



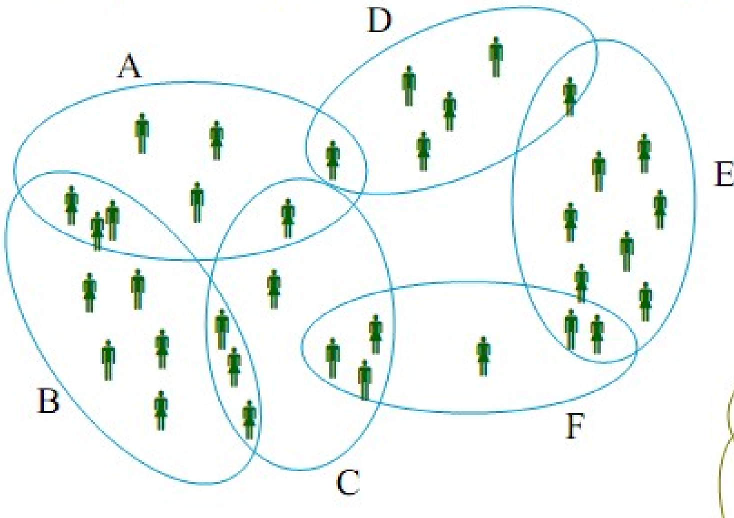
- Os alunos que fazem apenas uma disciplina influem na modelagem?
  - Não!
- O número de alunos que cursam simultaneamente um par de disciplinas influi na modelagem?
  - Não!
  - E se o objetivo fosse minimizar o número de alunos "sacrificados"?
  - Neste caso, sim!
- Este modelo pode ser simplificado?

- Simplificação tratando-se apenas as interseções:

Considerar apenas as interseções não-vazias.

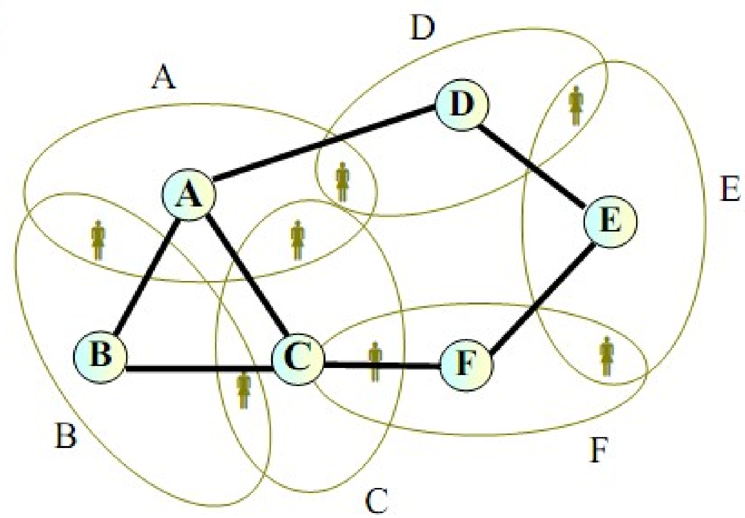


- Simplificação com a utilização de um grafo de conflitos:

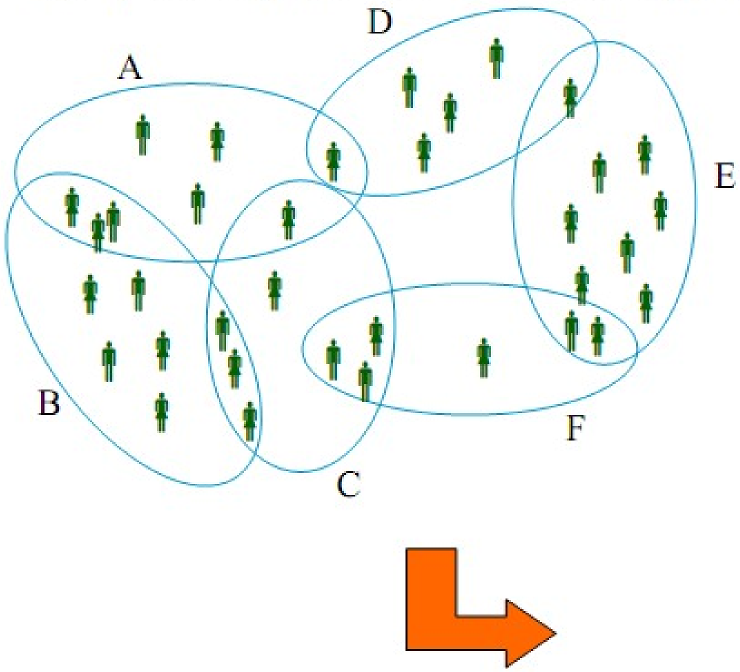


disciplinas → nós

conflitos → arestas

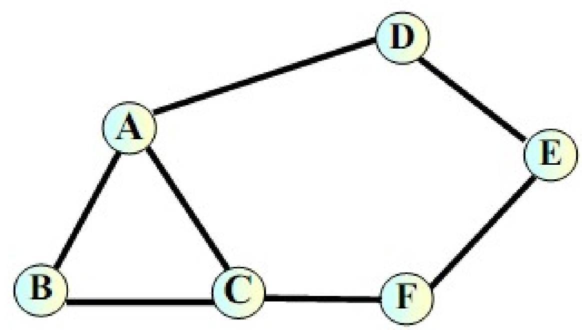


- Simplificação com a utilização de um grafo de conflitos:

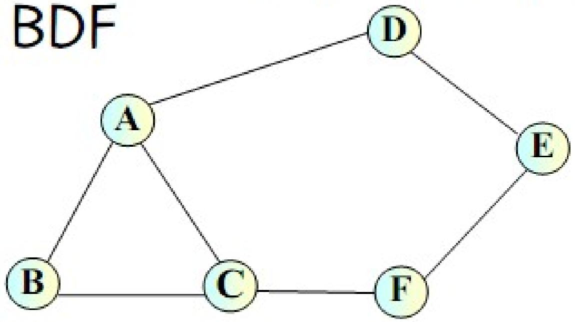


disciplinas ⟶ nós

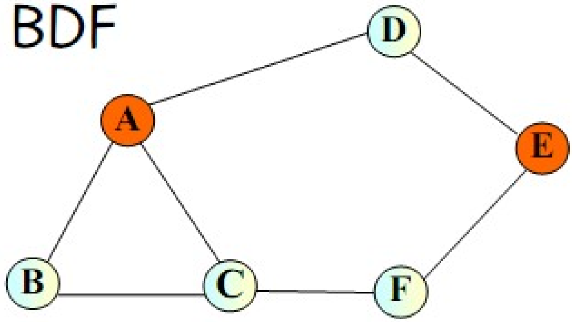
conflitos ⟶ arestas



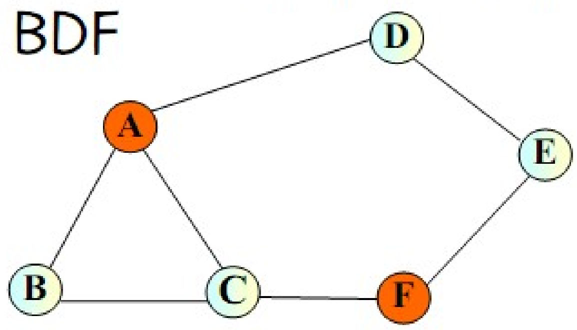
- Como montar um calendário satisfazendo as restrições de conflito?
- Marcar para o primeiro dia qualquer combinação de disciplinas sem conflitos: A, B, C, D, E, F, AE, AF, BD, BE, BF, CD, CE, DF, BDF



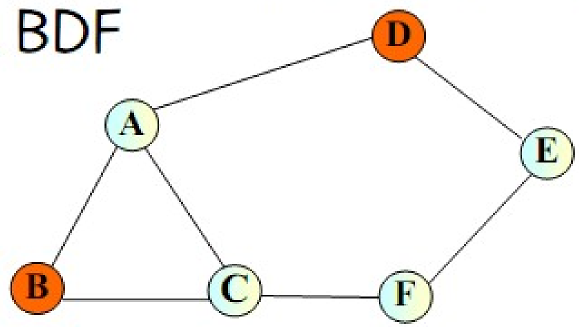
- Como montar um calendário satisfazendo as restrições de conflito?
- Marcar para o primeiro dia qualquer combinação de disciplinas sem conflitos: A, B, C, D, E, F, AE, AF, BD, BE, BF, CD, CE, DF, BDF



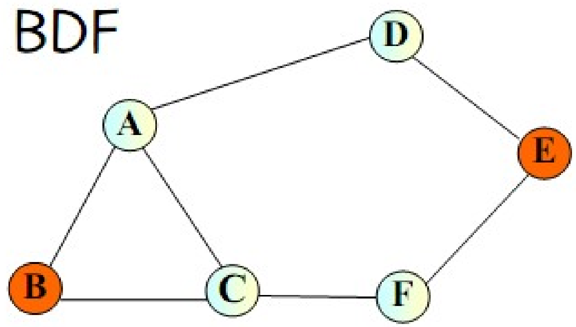
- Como montar um calendário satisfazendo as restrições de conflito?
- Marcar para o primeiro dia qualquer combinação de disciplinas sem conflitos: A, B, C, D, E, F, AE, AF, BD, BE, BF, CD, CE, DF, BDF



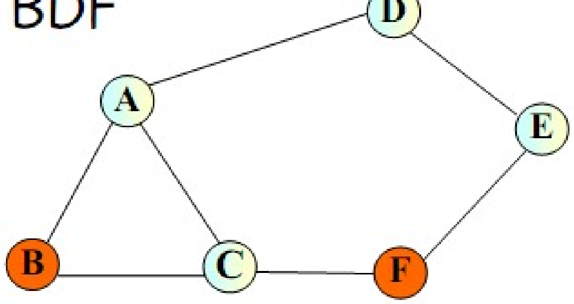
- Como montar um calendário satisfazendo as restrições de conflito?
- Marcar para o primeiro dia qualquer combinação de disciplinas sem conflitos: A, B, C, D, E, F, AE, AF, BD, BE, BF, CD, CE, DF, BDF



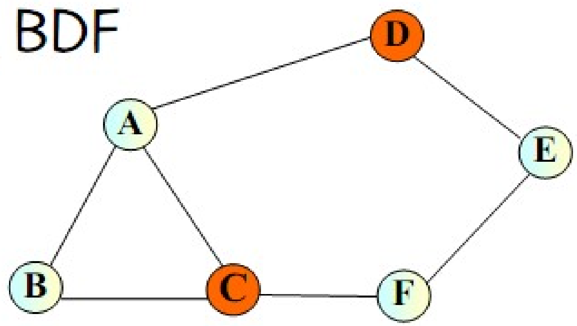
- Como montar um calendário satisfazendo as restrições de conflito?
- Marcar para o primeiro dia qualquer combinação de disciplinas sem conflitos: A, B, C, D, E, F, AE, AF, BD, BE, BF, CD, CE, DF, BDF



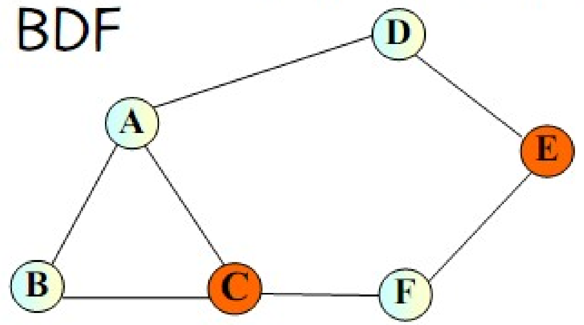
- Como montar um calendário satisfazendo as restrições de conflito?
- Marcar para o primeiro dia qualquer combinação de disciplinas sem conflitos: A, B, C, D, E, F, AE, AF, BD, BE, BF, CD, CE, DF, BDF



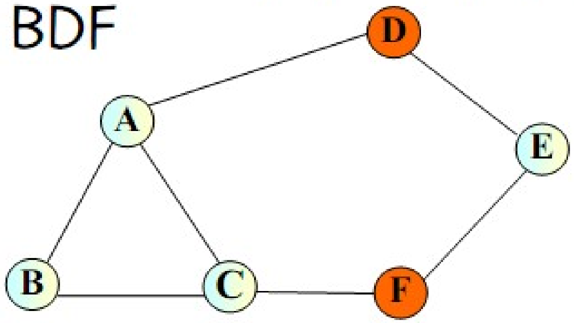
- Como montar um calendário satisfazendo as restrições de conflito?
- Marcar para o primeiro dia qualquer combinação de disciplinas sem conflitos: A, B, C, D, E, F, AE, AF, BD, BE, BF, CD, CE, DF, BDF



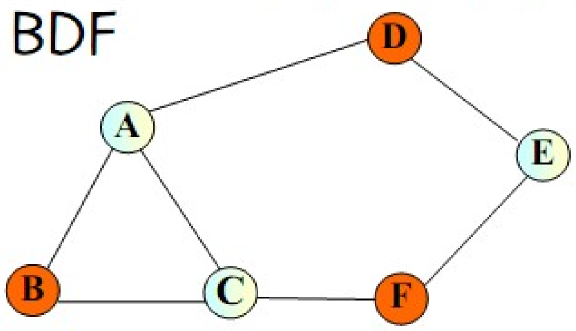
- Como montar um calendário satisfazendo as restrições de conflito?
- Marcar para o primeiro dia qualquer combinação de disciplinas sem conflitos: A, B, C, D, E, F, AE, AF, BD, BE, BF, CD, CE, DF, BDF



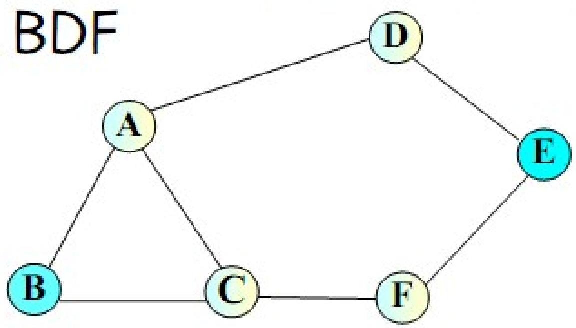
- Como montar um calendário satisfazendo as restrições de conflito?
- Marcar para o primeiro dia qualquer combinação de disciplinas sem conflitos: A, B, C, D, E, F, AE, AF, BD, BE, BF, CD, CE, DF, BDF



- Como montar um calendário satisfazendo as restrições de conflito?
- Marcar para o primeiro dia qualquer combinação de disciplinas sem conflitos: A, B, C, D, E, F, AE, AF, BD, BE, BF, CD, CE, DF, BDF

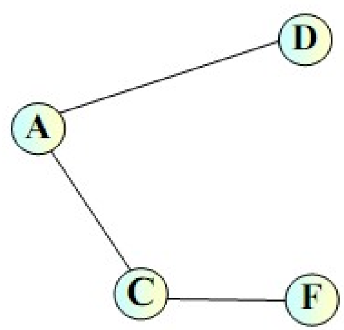


- Como montar um calendário satisfazendo as restrições de conflito?
- Marcar para o primeiro dia qualquer combinação de disciplinas sem conflitos: A, B, C, D, E, F, AE, AF, BD, BE, BF, CD, CE, DF, BDF

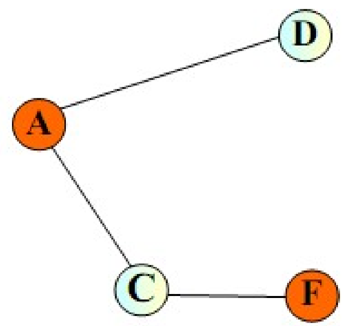


- Escolher por exemplo o par de disciplinas B e E para o primeiro dia e retirá-las do grafo.

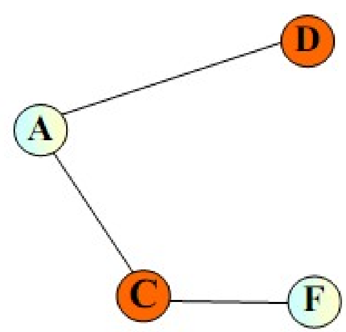
- Marcar para o segundo dia qualquer combinação de disciplinas sem conflitos: A, C, D, F, AF, CD, DF



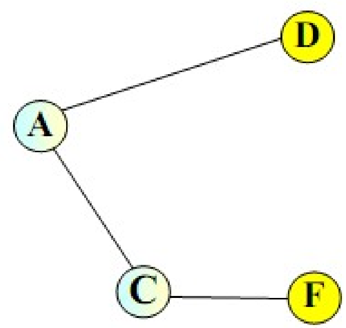
- Marcar para o segundo dia qualquer combinação de disciplinas sem conflitos: A, C, D, F, AF, CD, DF



- Marcar para o segundo dia qualquer combinação de disciplinas sem conflitos: A, C, D, F, AF, CD, DF

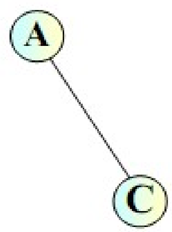


- Marcar para o segundo dia qualquer combinação de disciplinas sem conflitos: A, C, D, F, AF, CD, DF



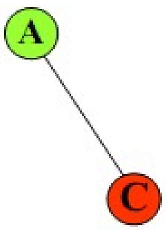
- Escolher por exemplo o par de disciplinas D e F para o segundo dia e retirá-las do grafo.

- Marcar para o segundo dia qualquer combinação de disciplinas sem conflitos: A, C, D, F, AF, CD, DF

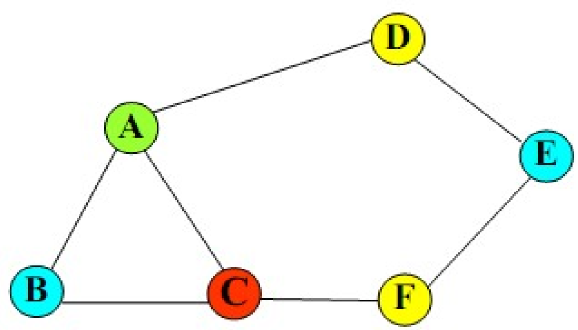


- Escolher por exemplo o par de disciplinas D e F para o segundo dia e retirá-las do grafo.

- Marcar A ou C para o terceiro dia e a que sobrar para o quarto dia:



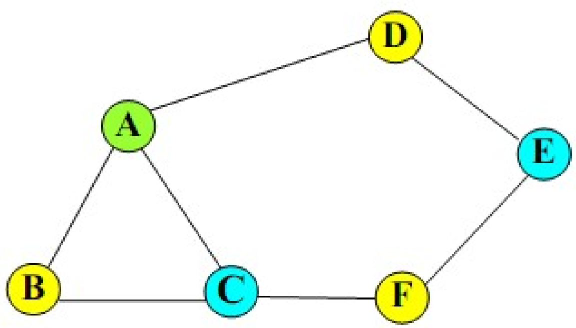
- A solução assim construída utiliza quatro dias: B-E no primeiro dia, D-F no segundo dia, A no terceiro e C no quarto:



- Uma solução com quatro dias corresponde a colorir os nós do grafo de conflitos com quatro cores!

- Esta solução utiliza o menor número possível de dias? ou ...
- É possível montar uma solução com menos dias? ou ...
- É possível colorir o grafo de conflitos com três cores?

Sim!



- É impossível colorir o grafo de conflitos com menos de três cores! (por que?)

- O algoritmo proposto nem sempre encontra a solução ótima (isto é, uma solução com o número mínimo de cores).
- Este algoritmo é então um algoritmo aproximado ou uma heurística para o problema de coloração de grafos.
- Algoritmos exatos vs. algoritmos aproximados:
  - qualidade da solução
  - tempo de processamento

Heurísticas

- No contexto de otimização combinatória, o termo heurística é usado em oposição aos métodos que garantem uma solução ótima

- Avanços no estudo e desenvolvimento de heurísticas
  - resolver problemas maiores
  - resolver problemas em tempos menores
  - obter melhores soluções