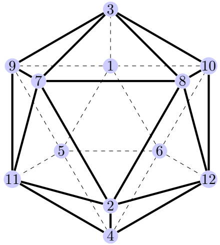
Metaheuristicas - Métodos de Resolução

Alexandre Checoli Choueiri

27/02/2023

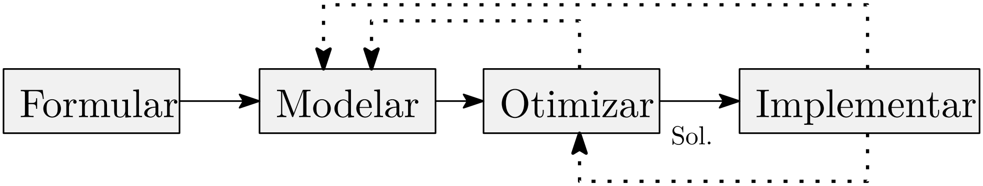


Processo de tomada de decisão

O processo de tomada de decisão

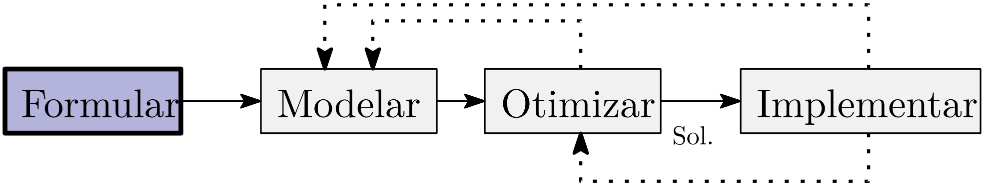
O processo para tomada de decisão baseado na ciência deve ser executado de forma sistemática. Conseguir **identificar um problema**, **desenvolver** algum tipo de modelo que o representa, encontrar uma solução (**otimizar**) e por fim **implementar** a solução no mundo real são algumas das etapas mais importantes deste processo.

Processo de tomada de decisão



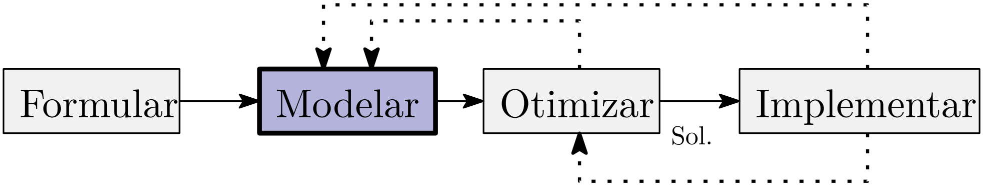
De forma geral, as seguintes etapas sempre são tomadas, seja de forma sistemática ou não.

Processo de tomada de decisão



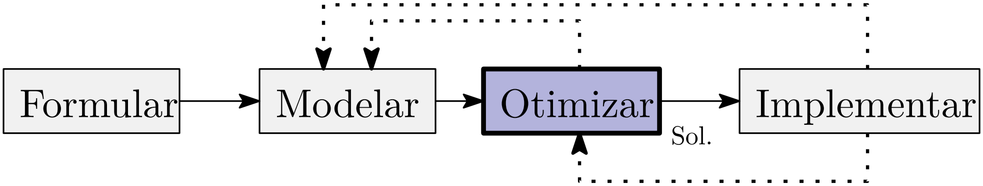
Formular: nesta etapa o problema é identificado de maneira informal, ou seja, os dados são levantados, requisições e preferências, e mais importante; se é um problema já conhecido (pela comunidade acadêmica) ou não. Sendo um problema conhecido o caminho é muito facilitado.

Processo de tomada de decisão



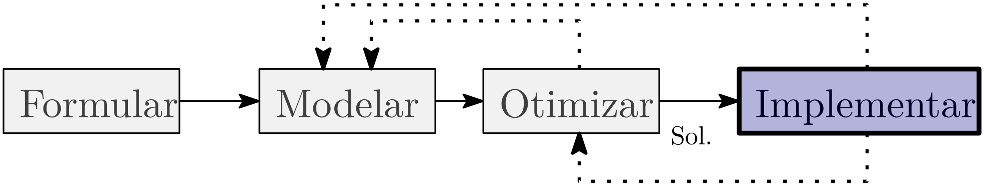
Modelar: nesta etapa um modelo matemático é desenvolvido para o problema. O modelo *sempre é uma simplificação da realidade*. Nesta etapa também fazemos a *modelagem computacional* do problema: como representamos o problema computacionalmente? suas entradas/saídas, estruturas de dados, restrições, etc...

Processo de tomada de decisão



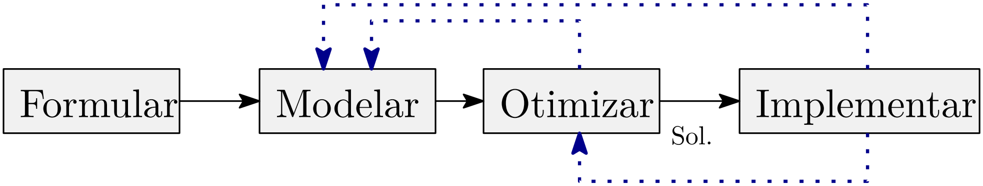
Otimizar: Nesta etapa deve-se encontrar uma solução para o problema. Diversos métodos podem ser usados nessa etapa, e é a maior parte do escopo deste curso. (DICA PARA O MERCADO: A solução não precisa ser a melhor do mundo...somente ser melhor e mais prática do que aquela gerada atualmente pelos gestores.)

Processo de tomada de decisão



Implementar: Com um método de solução satisfatório encontrado em "laboratório", nesta etapa a solução é adequada para o uso, ou seja, uma interface gráfica é criada, bancos de dados, etc...

Processo de tomada de decisão



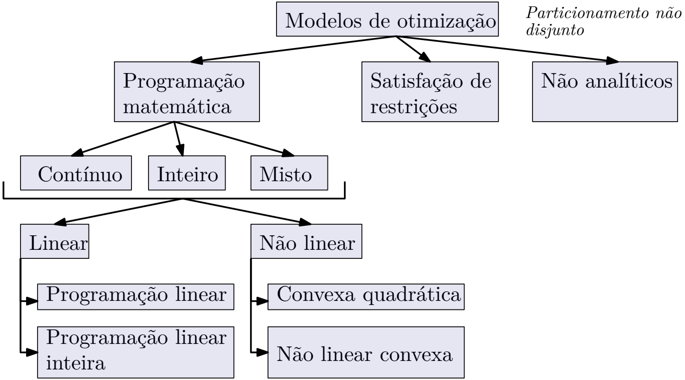
Retomar: Este processo nunca é linear como na imagem, pode ser necessário retomar etapas já executadas.

Modelos clássicos de otimização

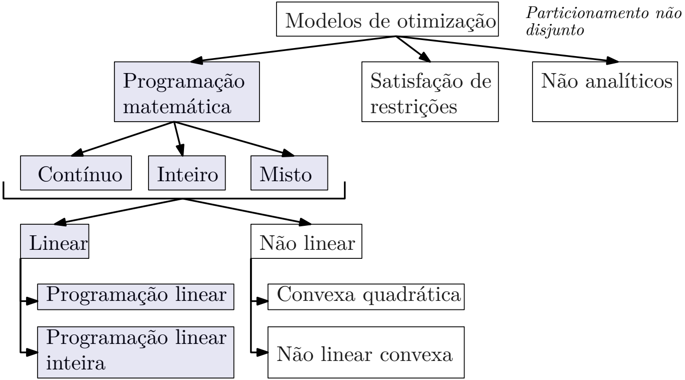
Modelagem

A etapa de modelagem é muito importante, é nela que "cercamos"o problema de forma matemática. Existem diversas técnicas para modelarmos um problema.

Modelos clássicos de otimização



Modelos clássicos de otimização



No nosso curso nos concentraremos em modelos de programação matemática **lineares** e **inteiros**.

Um modelo de **programação linear (PL)** admite valores reais não negativos para as suas variáveis, e assume a forma genérica:

$$\begin{aligned} \min \mathrm{z} = \mathbf{c}^T \mathbf{x} \\ \mathbf{A}\mathbf{x} = \mathbf{b} \\ \mathbf{x} \in R^n_+ \end{aligned}$$

Em que:

1. **A** é uma matriz mxn chamada matriz dos coeficientes.
2. $\mathbf{c}^T$ é o vetor dos custos.
3. $\mathbf{b}^T$ é o vetor dos termos independentes ou de recursos.

Um modelo de **programação inteira (PI)** admite valores inteiros não negativos para as suas variáveis, e assume a forma genérica:

$$\begin{aligned} \min \text{ z} = \mathbf{c}^T\mathbf{x} \\ \mathbf{A}\mathbf{x} = \mathbf{b} \\ \mathbf{x} \in Z_+^n \end{aligned}$$

Em que:

1. **A** é uma matriz mxn chamada matriz dos coeficientes.
2. $\mathbf{c}^T$ é o vetor dos custos.
3. $\mathbf{b}^T$ é o vetor dos termos independentes ou de recursos.

Modelos clássicos de otimização

Programação matemática linear

Um modelo de **programação inteira mista (PIM)** admite tanto valores inteiros quanto reais não negativos para as suas variáveis, e assume a forma genérica:

$$\begin{aligned} \text{min z} = \mathbf{c}^T\mathbf{x} + \mathbf{d}^T\mathbf{y} \\ \mathbf{Ax} + \mathbf{Dy} = \mathbf{b} \\ \mathbf{x} \in R_+^n, \mathbf{y} \in Z_+^p \end{aligned}$$

Em que:

1. **A** é uma matriz mxn e **D** uma matriz mxp.
2. $\mathbf{c}^T$ é um vetor $1xn$ e $\mathbf{y}^T$ um vetor $1xp$.
3. $\mathbf{b}^T$ é o vetor dos termos independentes ou de recursos.

Um modelo de **programação binaria (PB)** admite somente valores 0 ou 1 para suas variáveis.

$$
\begin{aligned}
\min \text{ z} = \mathbf{c}^T\mathbf{x} \\
\mathbf{A}\mathbf{x} = \mathbf{b} \\
\mathbf{x} \in B_+^n
\end{aligned}
$$

Em que:

1. **A** é uma matriz mxn chamada matriz dos coeficientes.
2. $\mathbf{c}^T$ é o vetor dos custos.
3. $\mathbf{b}^T$ é o vetor dos termos independentes ou de recursos.
4. B^n representa o espaço dos vetores com n componentes binárias (0,1).

EXERCÍCIO: Escreva o modelo de programação matemática para o problema da mochila visto na última aula (capacidade de 150). O problema possui os seguintes parâmetros:

	Ipod	**Abobrinha**	H_2O	**Canivete**	**Carne**	**Arroz**	**Aveia**	**PS4**
Valor	10	8	5	15	25	17	8	30
Peso	50	55	60	45	15	25	35	25

Modelos clássicos de otimização

Programação matemática: modelo da mochila

Sejam as variáveis:

$$\begin{cases} x_i : & \text{se o item } i \text{ é levado (1) na mochila ou não (0)}, i \in \{1, ..., 8\} \end{cases}$$

$$\begin{array}{ll} \text{maximizar} & 10x_1 + 8x_2 + 5x_3 + 15x_4 + 25x_5 + 17x_6 + 8x_7 + 30x_8 \\ \text{Sujeito à} & 50x_1 + 55x_2 + 60x_3 + 45x_4 + 15x_5 + 25x_6 + 35x_7 + 25x_8 \leq 150 \\ & x_i \in \{0, 1\}, i \in \{1, ..., 8\} \end{array}$$

Definição de problemas de otimização

OK..Já vimos diversos de **problemas de otimização**, e diversos métodos para modelarmos **problemas de otimização**. Mas afinal,

Definição de problemas de otimização

OK..Já vimos diversos de **problemas de otimização**, e diversos métodos para modelarmos **problemas de otimização**. Mas afinal,

O QUE SÃO PROBLEMAS DE OTIMIZAÇÃO?

Definição de problemas de otimização

Definição

Problema de otimização: Um problema de otimização pode ser definido como um par $(\mathbb{S}, f)$ em que $\mathbb{S}$ representa o conjunto de soluções factíveis e $f : \mathbb{S} \to \mathbb{R}$ a função objetivo a ser otimizada. A função objetivo atribui um valor a cada solução $s \in \mathbb{S}$ indicando a sua importância.

OBSERVAÇÕES:

1. O conjunto $\mathbb{S}$ de soluções factíveis é chamado de **espaço de busca** (esta definição está ainda incompleta...).
2. Não se confunda com as terminologias! Vimos **problemas** e **modelos** de otimização, porém não falamos nada sobre **métodos** (algoritmos) que resolvem estes problemas.

Definição de problemas de otimização

EXERCÍCIO: Considere um TSP com 4 pontos, e a matriz de distâncias abaixo. Este é um problema de otimização? Mostre escrevendo o espaço de busca $\mathbb{S}$ e a função objetivo f.

$$
\begin{array}{cc}
 & \begin{array}{cccc} p_1 & p_2 & p_3 & p_4 \end{array} \\
D = & \begin{bmatrix} 0 & 1 & 3 & 2 \\ 1 & 0 & 4 & 1 \\ 3 & 4 & 0 & 3 \\ 2 & 1 & 3 & 0 \end{bmatrix} \begin{array}{c} p_1 \\ p_2 \\ p_3 \\ p_4 \end{array}
\end{array}
$$

OBSERVAÇÕES: Os elementos da matriz indicam a distância entre os pontos, ou seja: d_{ij} = distância entre o ponto i e o ponto j.

Definição de problemas de otimização

Definição de problemas de otimização

Se representarmos uma rota por um vetor numérico em que cada índice representa um ponto (considerando que o último ponto volta ao primeiro), obtemos o seguinte espaço de busca (conjunto $\mathbb{S}$):

$$\mathbb{S} = \{\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 4 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 & 2 & 4 \end{bmatrix}, \begin{bmatrix} 1 & 3 & 4 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 4 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 1 & 4 & 3 & 2 \end{bmatrix}\}$$

Definição de problemas de otimização

Se representarmos uma rota por um vetor numérico em que cada índice representa um ponto (considerando que o último ponto volta ao primeiro), obtemos o seguinte espaço de busca (conjunto $\mathbb{S}$):

$$\mathbb{S} = \{\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 4 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 & 2 & 4 \end{bmatrix}, \begin{bmatrix} 1 & 3 & 4 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 4 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 1 & 4 & 3 & 2 \end{bmatrix}\}$$

A função objetivo para algumas soluções fica então:

$$f(s_1) = f(\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}) = 1 + 3 + 2 + 2 = 9$$

$$f(s_2) = f(\begin{bmatrix} 1 & 2 & 4 & 3 \end{bmatrix}) = 1 + 1 + 3 + 3 = 8$$

$$\ldots$$

Definição de problemas de otimização

Definimos formalmente o que é um **problema de otimização**. Mas qual é o nosso objetivo com esse problema? O que queremos encontrar?

Definição de problemas de otimização

Definimos formalmente o que é um **problema de otimização**. Mas qual é o nosso objetivo com esse problema? O que queremos encontrar?

Definição

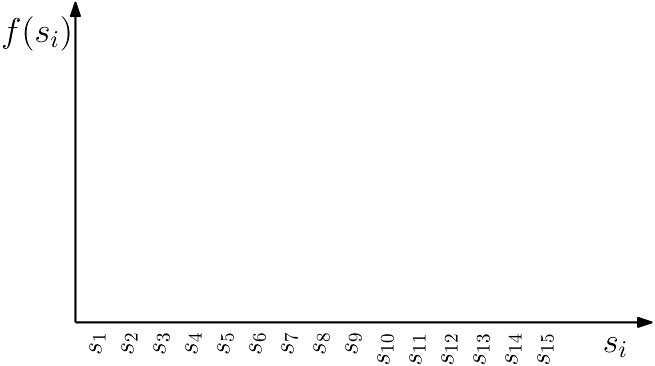
Ótimo global: Uma solução $s^* \in \mathbb{S}$ é um *ótimo global* se possui uma função objetivo melhor do que todas as outras soluções do espaço de busca, ou seja, $f(s^*) \leq f(s) \forall s \in \mathbb{S}$ para um problema de minimização.

De forma que o **nosso objetivo** é então encontrar a solução que é um ótimo global para o problema de otimização.

Definição de problemas de otimização

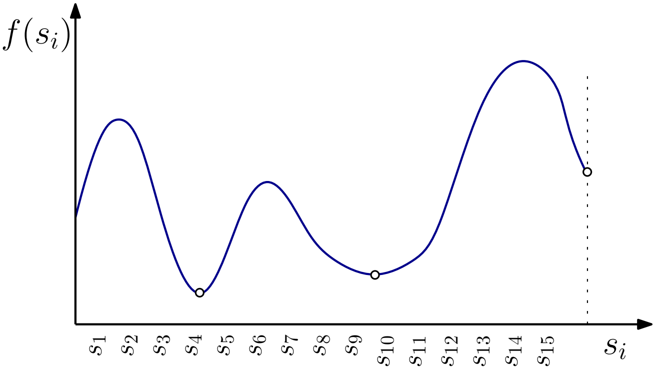
Bom...isso é simples, podemos gerar todo o *espaço de busca* ($\mathbb{S}$) de um problema de otimização e avaliar os valores das funções objetivo (f), selecionando o menor de todos.

Definição de problemas de otimização



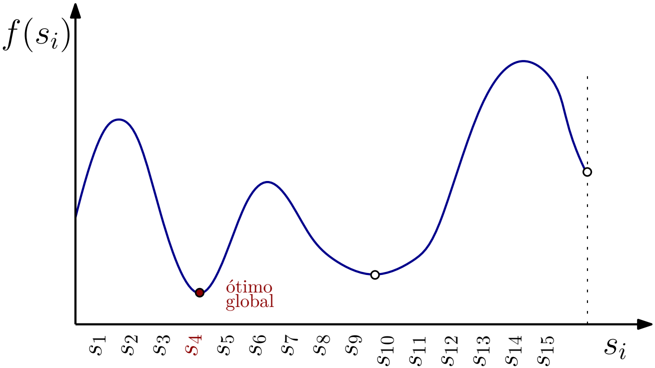
Imagine que escrevemos todo o espaço de busca no eixo x e calculamos os valores de f em y. Temos um gráfico como mostrado na Figura acima.

Definição de problemas de otimização



Imagine que escrevemos todo o espaço de busca no eixo x e calculamos os valores de f em y. Temos um gráfico como mostrado na Figura acima.

Definição de problemas de otimização



Claramente s_4 é um ótimo global! Pronto, encontramos a melhor solução para o nosso problema.

Definição de problemas de otimização

Problemática

Pergunta

Por quê então não usamos esta técnica? Gerar todo o espaço de busca e verificar qual a melhor solução? (OBS: esse tipo de estratégia é conhecida como *força-bruta*, justamente por gerar todo o espaço).

Pergunta

Por quê então não usamos esta técnica? Gerar todo o espaço de busca e verificar qual a melhor solução? (OBS: esse tipo de estratégia é conhecida como *força-bruta*, justamente por gerar todo o espaço).

Vamos pensar um pouco...

Definição de problemas de otimização

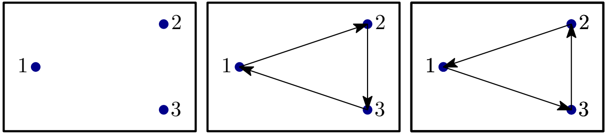
Problemática

Quantas rotas são possíveis de serem feitas em um problema do caixeiro-viajante com 3 pontos?

Definição de problemas de otimização

Problemática

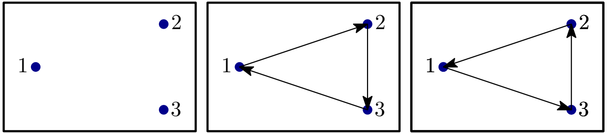
Quantas rotas são possíveis de serem feitas em um problema do caixeiro-viajante com 3 pontos?



Definição de problemas de otimização

Problemática

Quantas rotas são possíveis de serem feitas em um problema do caixeiro-viajante com 3 pontos?

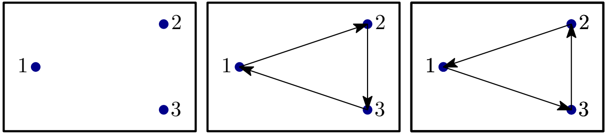


E se nosso problema tivesse 4 pontos?

Definição de problemas de otimização

Problemática

Quantas rotas são possíveis de serem feitas em um problema do caixeiro-viajante com 3 pontos?

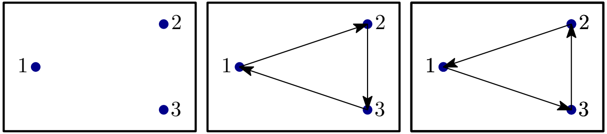


E se nosso problema tivesse 4 pontos? Como no exemplo passado, vimos que 4 pontos geraram um espaço de busca $\mathbb{S}$ com 6 soluções. Qual a relação que define o número de rotas em função do número de pontos?

Definição de problemas de otimização

Problemática

Quantas rotas são possíveis de serem feitas em um problema do caixeiro-viajante com 3 pontos?



E se nosso problema tivesse 4 pontos? Como no exemplo passado, vimos que 4 pontos geraram um espaço de busca $\mathbb{S}$ com 6 soluções. Qual a relação que define o número de rotas em função do número de pontos?

Para n cidades em um problema do caixeiro viajante, existem $(n-1)!$ rotas possíveis.

Definição de problemas de otimização

Problemática

Se considerarmos um computador capaz de realizar **1 bilhão de somas/segundo**. Um problema de 20 cidades precisa de **20 operações de soma** para obter o custo de cada rota. Assim, o computador é capaz de analisar:

Definição de problemas de otimização

Problemática

Se considerarmos um computador capaz de realizar **1 bilhão de somas/segundo**. Um problema de 20 cidades precisa de **20 operações de soma** para obter o custo de cada rota. Assim, o computador é capaz de analisar:

$$\frac{10^9}{20} = 53 \text{ milhões de rotas/segundo} \tag{1}$$

Definição de problemas de otimização

Problemática

Se considerarmos um computador capaz de realizar **1 bilhão de somas/segundo**. Um problema de 20 cidades precisa de **20 operações de soma** para obter o custo de cada rota. Assim, o computador é capaz de analisar:

$$\frac{10^9}{20} = 53 \text{ milhões de rotas/segundo} \tag{1}$$

O número de rotas é:

$$19! = 1.2x10^{17} \tag{2}$$

Definição de problemas de otimização

Problemática

Se considerarmos um computador capaz de realizar **1 bilhão de somas/segundo**. Um problema de 20 cidades precisa de **20 operações de soma** para obter o custo de cada rota. Assim, o computador é capaz de analisar:

$$\frac{10^9}{20} = 53 \text{ milhões de rotas/segundo} \tag{1}$$

O número de rotas é:

$$19! = 1.2x10^{17} \tag{2}$$

O tempo necessário para avaliar essas rotas é então:

$$\boxed{\frac{1.2x10^{17}}{53 \text{ milhões}} = \textbf{73 anos!}} \tag{3}$$

Definição de problemas de otimização

Problemática

A Tabela abaixo mostra os tempos de processamento utilizando o algoritmo de força-bruta, em função do número de cidades:

n	**Tempo**
5	insignificante
10	0.003s
15	20 min
20	73 anos
25	**470 milhões de anos**

Definição de problemas de otimização

Problemática

A Tabela abaixo mostra os tempos de processamento utilizando o algoritmo de força-bruta, em função do número de cidades:

n	**Tempo**
5	insignificante
10	0.003s
15	20 min
20	73 anos
25	**470 milhões de anos**

Isso para o problema do caixeiro viajante...imagine os problemas mais complexos de otimização que vimos nas últimas aulas!

Definição de problemas de otimização

Problemática

Pergunta

Por quê então não usamos esta técnica? Gerar todo o espaço de busca e verificar qual a melhor solução? (OBS: esse tipo de estratégia é conhecida como *força-bruta*, justamente por gerar todo o espaço).

Definição de problemas de otimização

Problemática

Pergunta

Por quê então não usamos esta técnica? Gerar todo o espaço de busca e verificar qual a melhor solução? (OBS: esse tipo de estratégia é conhecida como *força-bruta*, justamente por gerar todo o espaço).

RESPOSTA: Porquê existe uma explosão combinatória nos problemas, e mesmo com o uso de computadores é inviável usar força-bruta!

Definição de problemas de otimização

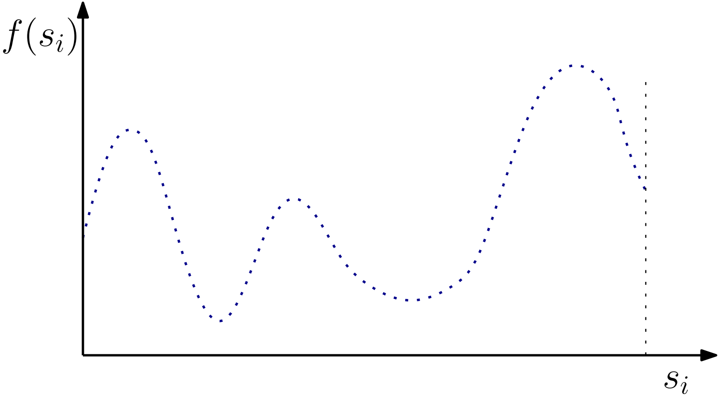
Problemática

Pergunta

Por quê então não usamos esta técnica? Gerar todo o espaço de busca e verificar qual a melhor solução? (OBS: esse tipo de estratégia é conhecida como *força-bruta*, justamente por gerar todo o espaço).
RESPOSTA: Porquê existe uma explosão combinatória nos problemas, e mesmo com o uso de computadores é inviável usar força-bruta!

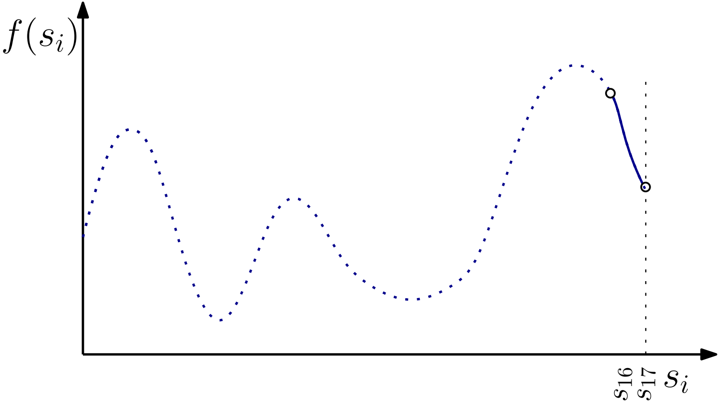
Na prática, os algoritmos que resolvem problemas de otimização não geram todo o espaço de busca, **apenas um pequeno subconjunto deste**. A forma de geração deste subconjunto é o que diferencia os vários métodos de resolução.

Métodos de resolução



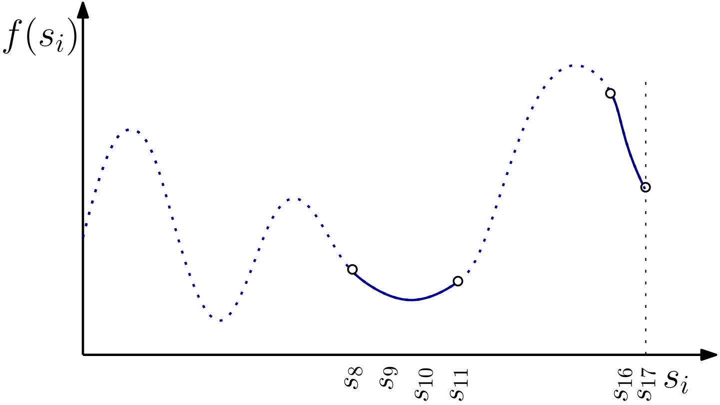
Considerando novamente o espaço de busca.

Métodos de resolução



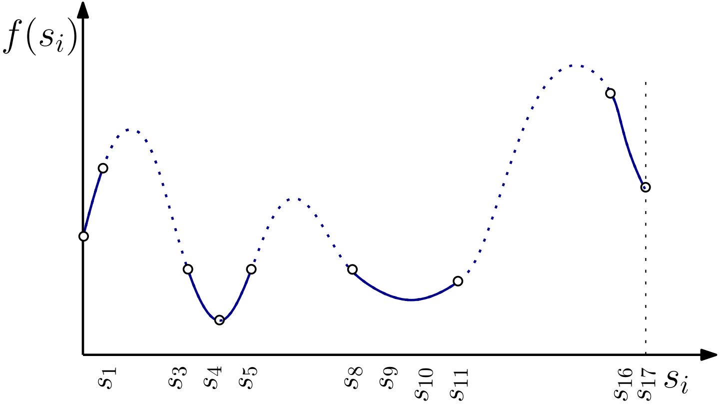
Os métodos de resolução geram um subconjunto do espaço total.

Métodos de resolução



De uma forma inteligente, para tentar encontrar o ótimo global.

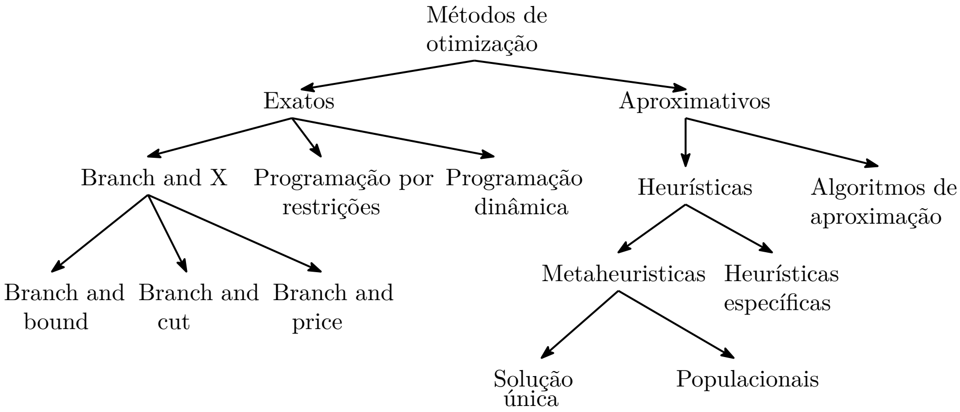
Métodos de resolução



Evitando assim a explosão combinatória gerada pelos algoritmos de força-bruta.

Métodos de resolução

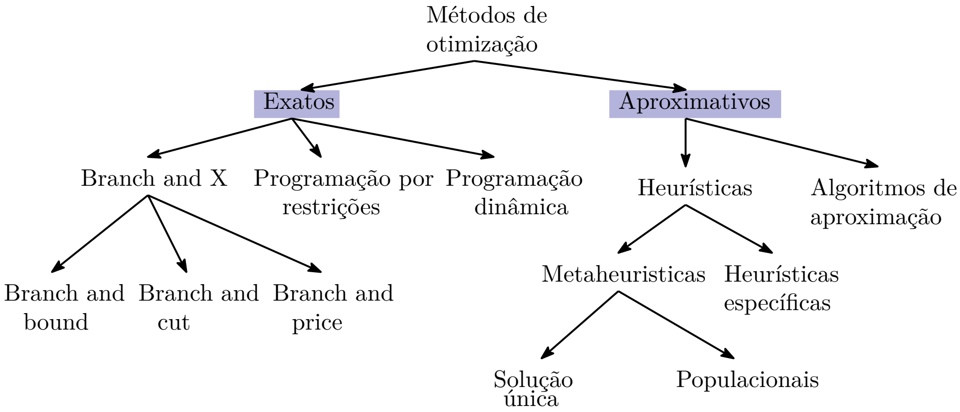
Árvore de métodos



Os métodos de resolução de problemas de otimização são resumidos na **árvore acima**.

Métodos de resolução

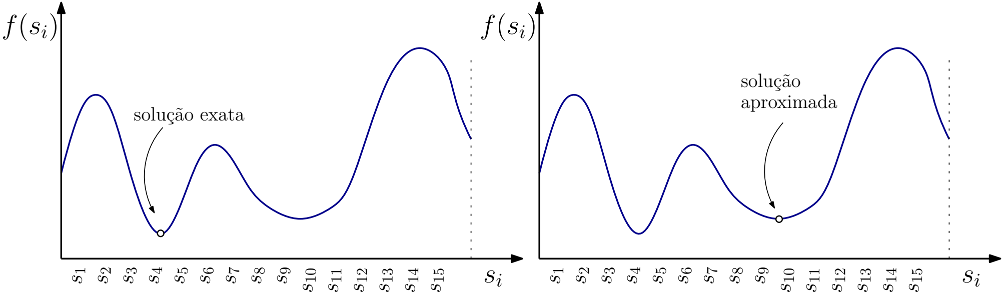
Exatos vs Aproximativos



A primeira grande separação nos métodos é em relação ao tipo de solução que eles encontram: o valor da função objetivo é o ótimo global ou não?

Métodos de resolução

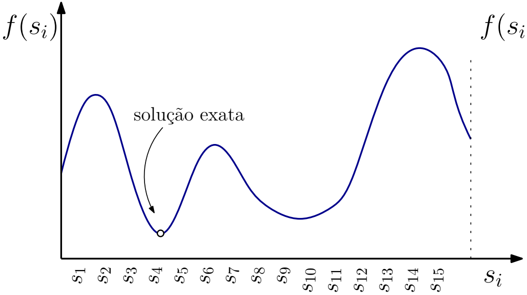
Exatos vs Aproximativos

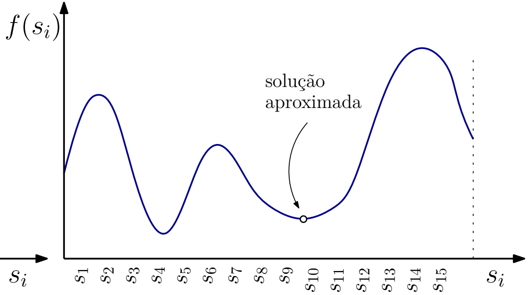


Se o algoritmo garante que o ótimo global é encontrado, ele é considerado **exato**. Se o ótimo não pode ser garantido ele é considerado **aproximativo**.

Métodos de resolução

Exatos vs Aproximativos

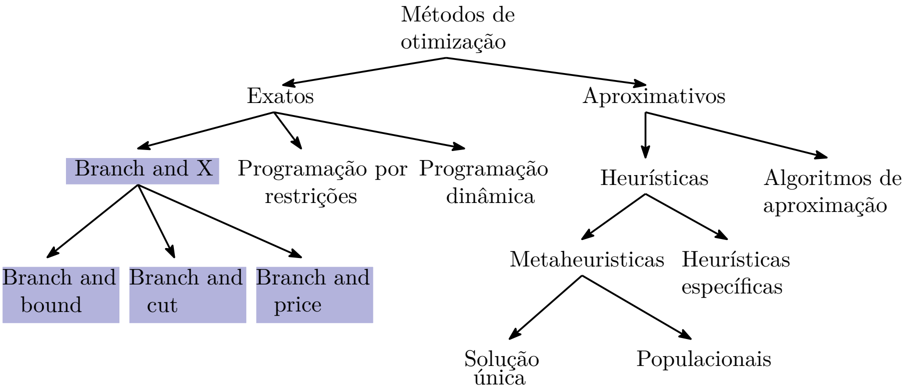




Atenção: um algoritmo aproximativo pode encontrar o ótimo. Só não é garantido matematicamente que ele o fará.

Métodos de resolução

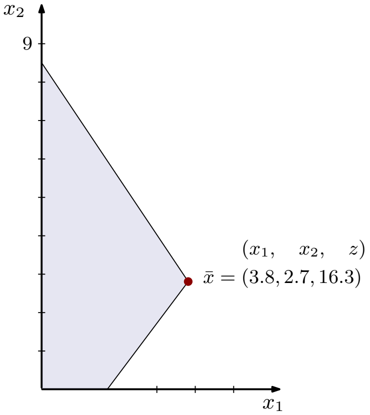
Branch and Bound

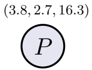


Os métodos exatos baseados em *Branching* utilizam o algoritmo Simplex como seu motor de otimização.

Métodos de resolução

Branch and Bound

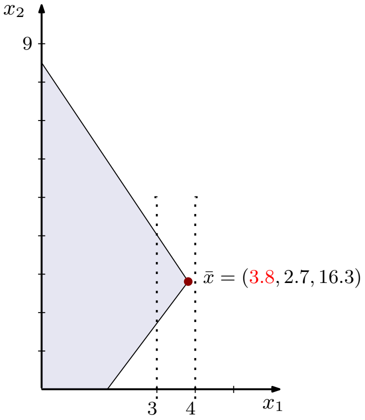


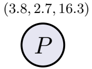


O Simplex é resolvido diversas vezes, e novas restrições são inseridas no modelo.

Métodos de resolução

Branch and Bound

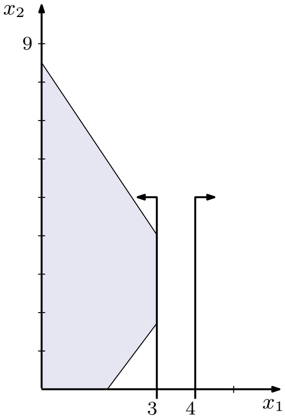


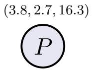


Esse processo é repetido até que uma solução inteira seja encontrada.

Métodos de resolução

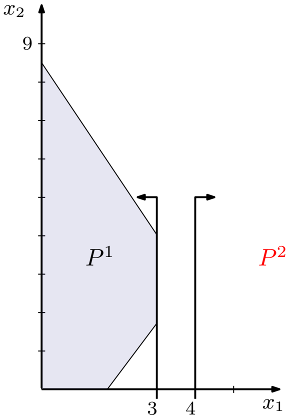
Branch and Bound

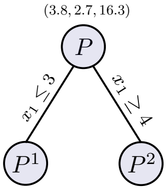




Métodos de resolução

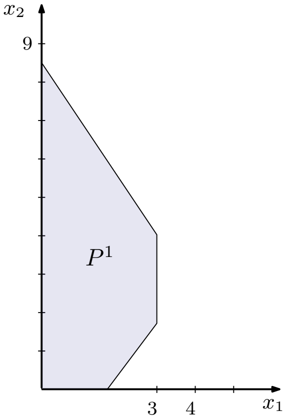
Branch and Bound

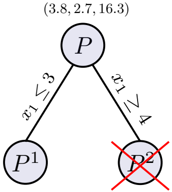




Métodos de resolução

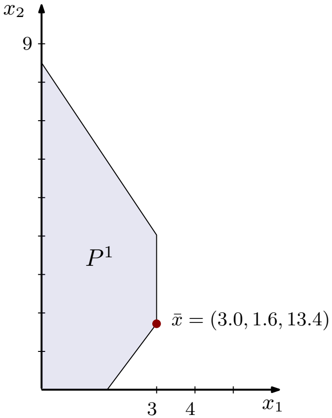
Branch and Bound

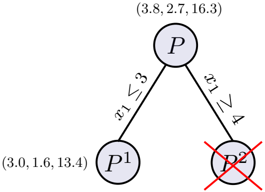




Métodos de resolução

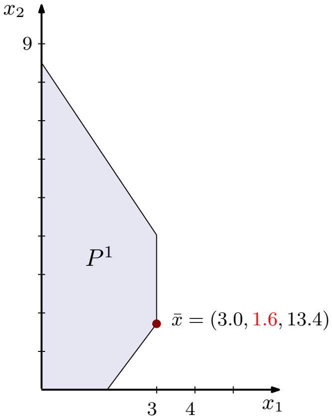
Branch and Bound

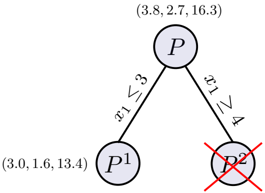




Métodos de resolução

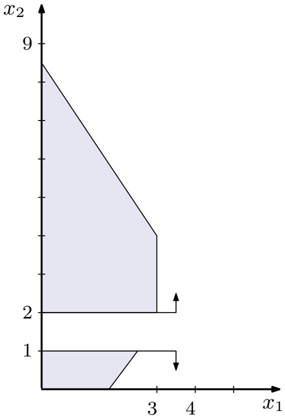
Branch and Bound

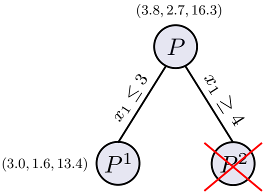




Métodos de resolução

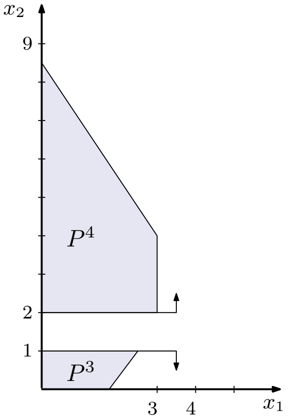
Branch and Bound

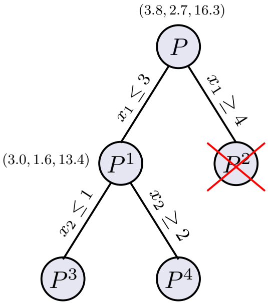




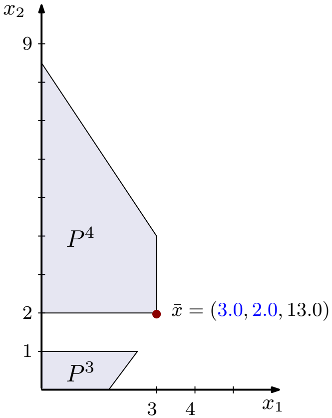
Métodos de resolução

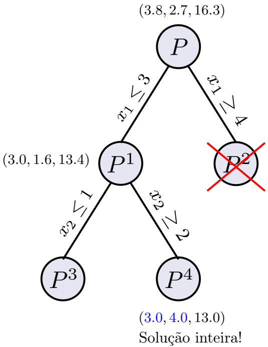
Branch and Bound





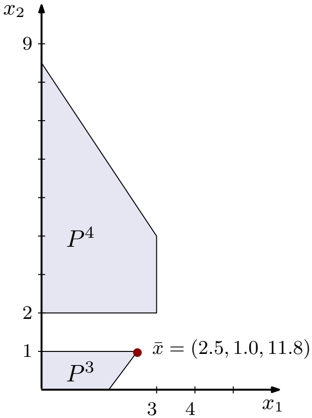
Métodos de resolução
Branch and Bound

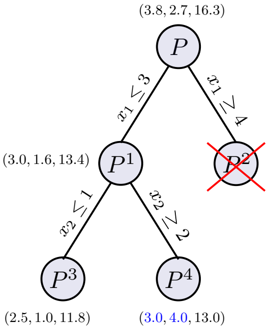




Métodos de resolução

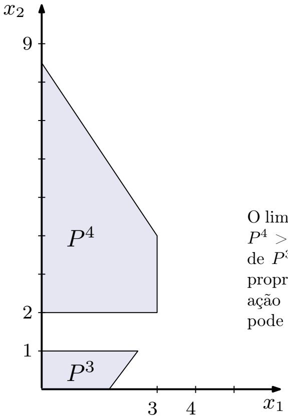
Branch and Bound



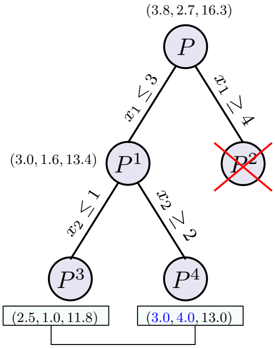


Métodos de resolução

Branch and Bound

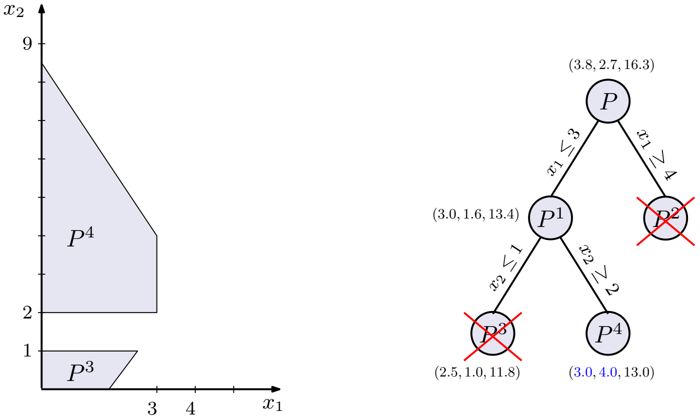


O limitante inferior de $P^4 >$ relaxação linear de P^3, portanto, pela propriedade da relaxação linear, o nó P^3 pode ser eliminado.



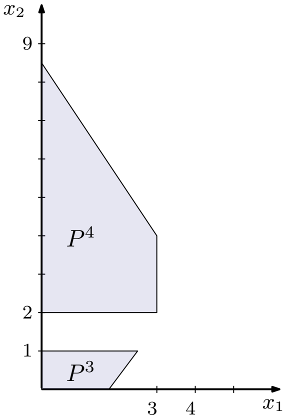
Métodos de resolução

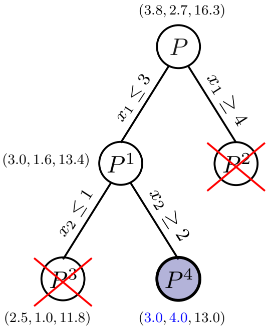
Branch and Bound



Métodos de resolução

Branch and Bound





EXERCÍCIO: Encontre a solução ótima do problema da mochila visto nos exercícios anteriores, usando o *solver open source* GLPK **G**eneral **L**inear **P**rogramming **K**it, por meio da interface GUSEK.

OBSERVAÇÕES:

1. Os softwares que implementam algoritmos de otimização (como o Simplex e B&B) são chamados de *solvers*, como o GLPK. O GLPK não possui interface, e na maioria das vezes é usado como uma biblioteca (API) por meio de alguma linguagem de programação.
2. No entanto, o professor **Leandro Magatão** (UTFPR - Curitiba) implementou uma solução com interface gráfica utilizando o glpk, chamada GUSEK. Com ele podemos escrever modelos de PL em arquivos de texto e encontrar a solução em um mesmo ambiente.
3. Para uma referência sobre a escrita do formato do arquivo .lp, veja o site do solver GUROBI

Métodos de resolução

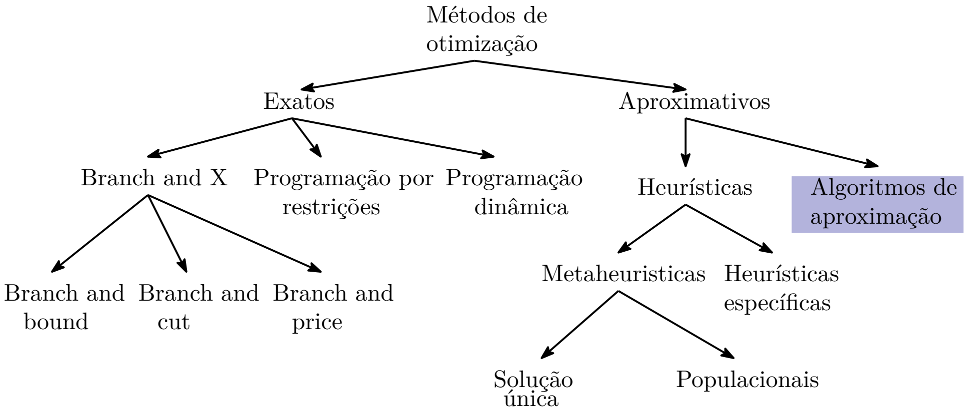
Branch and Bound

Atenção

Mesmo realizando uma busca inteligente no espaço de soluções, usualmente os **algoritmos exatos também não são viáveis para problemas práticos**. O tempo computacional necessário para encontrar uma solução ótima ainda é muito grande (claro que menor do que a força bruta). Dessa forma, lança-se mão dos algoritmos aproximativos.

Métodos de resolução

Aproximativos



Dentre os algoritmos aproximativos existe uma separação entre as heurísticas e os algoritmos de aproximação.

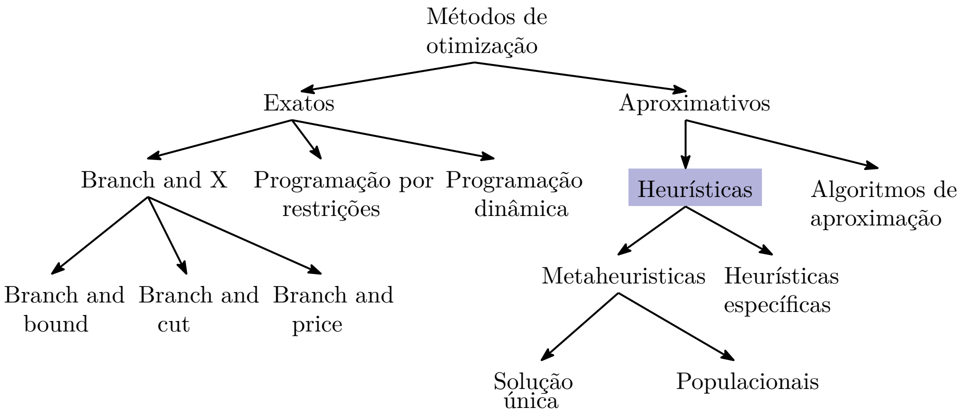
Definição

Algoritmos de aproximação: Em algoritmos de aproximação, existe uma garantia no limitante (bound) da solução encontrada em relação ao ótimo global.

EXEMPLO: O algoritmo de *Christofides–Serdyukov* para o problema do caixeiro viajante garante que o custo da solução gerada não ultrapassa $\frac{3}{2}$ do custo da solução ótima. Ou seja, em uma instância em que a rota ótima tem um custo de 10, no pior cenário possível o algoritmo gera uma solução de custo $\frac{3}{2}10 = 15$.

Métodos de resolução

Aproximativos



Finalmente vamos entender o nome da nossa disciplina!

Introdução

Origens

Heurística

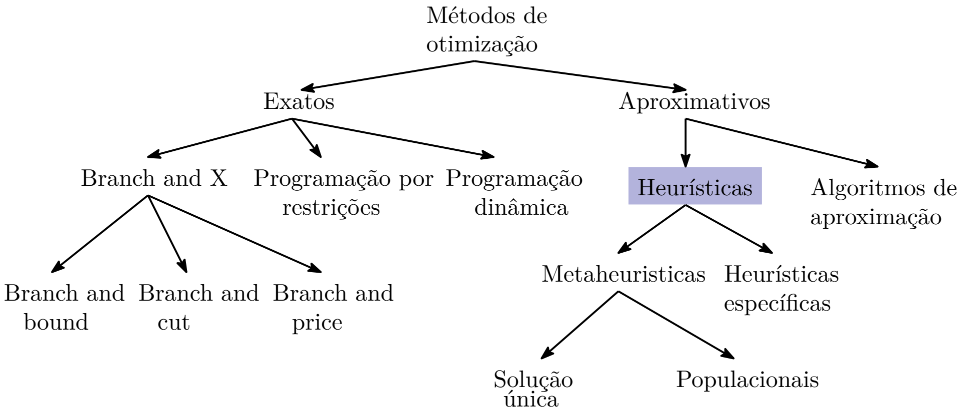
Heuriskein: Antiga palavra de origem Grega com significado "a arte de descobrir novas estratégias (regras) para resolver problemas".

Meta

Meta: O sufixo *meta*, também de origem grega significa, "metodologia de nível superior"(meta linguagem, metafísica, Meta).

Métodos de resolução

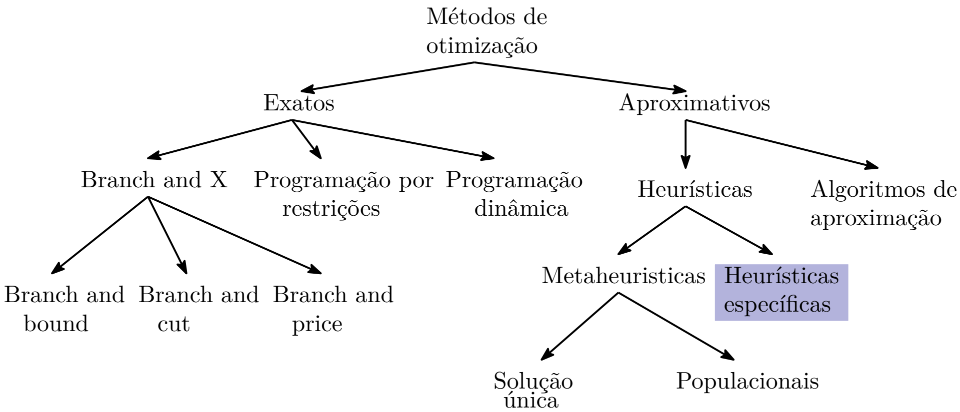
Heurísticas



Por exclusão, sabemos que as Heurísticas não garantem o ótimo, e nem mesmo um limitante em relação a solução ótima.

Métodos de resolução

Heurísticas

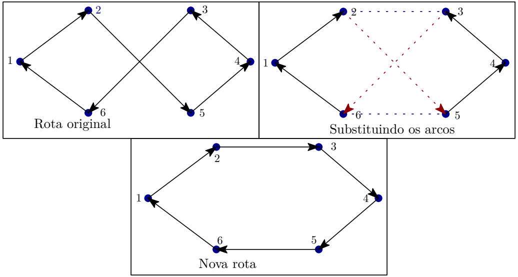


As heurísticas específicas são talhadas para resolver um tipo específico de problema.

Métodos de resolução

Heurísticas

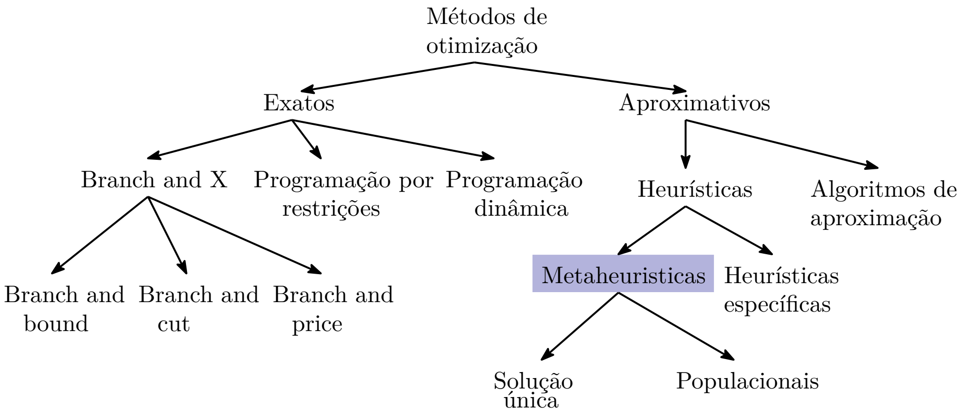
EXEMPLO: Um algoritmo de otimização do problema do caixeiro viajante é chamado de 2-OPT. Ele consiste em destruir dois arcos de uma rota e substituir por outros dois arcos, como mostrado na Figura abaixo:



Note que é impossível aplicar um 2-OPT em um problema de carregamento de contêineres, por exemplo. Não faria sentido, de forma que o 2-OPT é uma heurística específica para o TSP.

Métodos de resolução

Metaheuristicas



As Metaheurísticas, por outro lado, são procedimentos que servem para qualquer problema de otimização. Elas "guiam" as heurísticas específicas (Meta-acima).

Algorithm 1 Template genérico busca local

$s = s_0$ ▷ Gerar uma solução inicial
while Critério de parada não for satisfeito **do**
  Altere a solução s por s'
  if s' for melhor do que s **then**
    $s = s'$
  end if
end while
return s.

EXEMPLO: A metaheuristica mais simples é chamada de busca local (*local search* - LS). Nesse algoritmo, uma melhoria na solução é feita a cada iteração, até que um critério de parada seja satisfeito.

Métodos de resolução

Metaheuristicas

Algorithm 2 Template genérico busca local

$s = s_0$ ▷ Gerar uma solução inicial
while Critério de parada não for satisfeito **do**
  Altere a solução s por s' ▷ Colocar o 2-OPT aqui
  if s' for melhor do que s **then**
    $s = s'$
  end if
end while
return s.

Note que o algoritmo pode ser aplicado para qualquer problema! O que é alterado é o método de alteração de s. No caso do TSP, poderíamos colocar o 2-OPT nesta etapa.

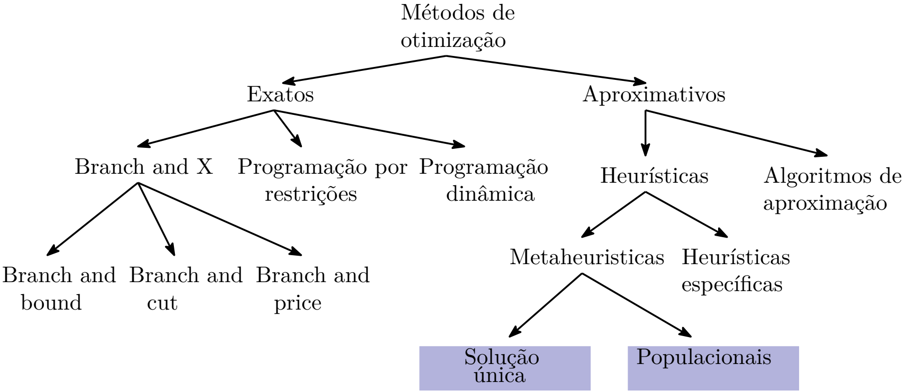
Métodos de resolução

Metaheuristicas

Percebe-se que a **Meta**heurística age **sobre** a heurística específica (2-OPT). Se escolhermos outro problema de otimização, o LS poderia ser utilizado, a única coisa que seria diferente é o método de alteração da soluções, que seria uma heurística específica para o outro problema.

Métodos de resolução

Solução única x populacionais



As metaheurísticas, por sua vez, podem ser separadas em dois grupos: aquelas que desenvolvem soluções únicas (como o LS), e as que geram mais de uma solução (como algoritmos genéticos).

Atividade

1. O que é um problema de otimização? Crie uma instancia pequena (4 itens) para o problema da mochila, e mostre que ele é um problema de otimização.
2. Escreva o modelo de programação inteira para o problema de designação visto na última apresentação.
3. Resolva o problema usando o software GUSEK, e mostre a solução ótima (designação e custo total).
4. O que são algoritmos de força-bruta? Por que não os utilizamos em problemas de otimização?
5. Qual a diferença entre algoritmos aproximativos e exatos? E entre heurísticas e algoritmos de aproximação?

Referências I

Marcos Nereu Arenales, Vinícius Amaral Armentano, Reinaldo Morabito Neto, and Horacio Hideki Yanasse.
Pesquisa operacional para cursos de engenharia. 2007.