Métodos Heurísticos: Introdução e Construção

Caps. 1, 2, 4
Sec. 3.3

Cap. 2

- Problema
  - Modelo → Programação Matemática → Solução Ótima! → Inviável para problemas complexos! (e.g., Problema mochila, NP-difíceis)
  - Código → Métodos Heurísticos → Solução Boa! → Não garantem otimalidade, porém são muito eficientes (tempo, espaço)

Métodos Heurísticos: Métodos gerais, independentes de problema, que definem estratégias para guiar a busca de soluções.

Visão Geral:

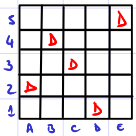
- Otimização
  - Métodos Exatos
    - Simplex
      - Ex: Dieta, mix produtos
    - Branch & Bound
      - Ex: Cobertura vértices, Mochila
  - Métodos Heurísticos (→ Mais liberdade (criatividade))
    - Construção
      - Aleatória,
      - Gulosa,
      - Semi-gulosa
    - Refinamento
      - Buscas locais (simples, iterada, ...)

Representação de Soluções

- Específica para cada problema
- Define o tamanho do espaço de soluções
- " a complexidade das operações
- Permite excluir (algumas) soluções inviáveis

EXEMPLOS:

- **n-queens**

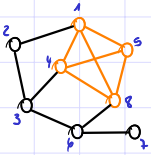


PL: $X_{ij} \in \{0,1\}$ → $\begin{bmatrix} 0&0&0&0&1 \\ 0&1&0&0&0 \\ 0&0&1&0&0 \\ 1&0&0&0&0 \\ 0&0&0&1&0 \end{bmatrix}$

$R = (2,4,3,1,5)$ (A B C D E) → vs. $[2,4,3,1,5]$

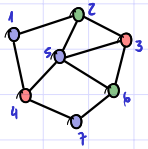
↳ + Trocitão repetição!

- **Clique máximo (max clique)**



$C = (1,0,0,1,1,0,0,1)$ (1 2 3 4 5 6 7 8)

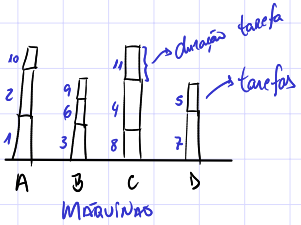
- **Coloração de vértices**



Seja $\{1,2,\dots,c\}$ o conjunto de cores

$C = (1,3,2,2,1,3,1)$ (1 2 3 4 5 6 7)

- **Pm || C_{max}**



$T = (A,A,B,C,D,B,D,C,B,A,C)$ (1 2 3 4 5 6 7 8 9 10 11)

Heurística Construtiva: Inicia com uma solução vazia e iterativamente adiciona componentes/elementos até que uma solução completa é produzida.

↳ Ou seja, constrói uma solução "do zero"!

Construção (G):

```
S ← ∅
C ← elementos candidatos
Enquanto C ≠ ∅ e solução não completa Faça
    c ← Seleciona c ∈ C conforme estratégia G
    Se c pode ser inserido em S então
        S ← S ∪ {c}
    C ← C \ {c}
Retorna S
```

Estratégia (G)	Descrição	Algoritmo
Aleatória	Seleciona um elemento aleatoriamente	Construção aleatória
Gulosa	Seleciona o melhor elemento	Construção gulosa
Semi-Gulosa	Seleciona um dos K melhores elementos	Construção semi-gulosa

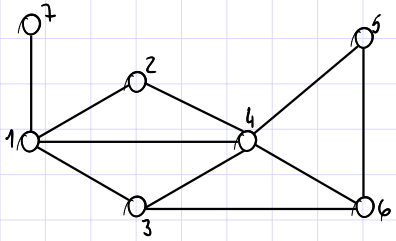
↳ Parâmetro K: 1) Constante (ex: K = 10)
2) Percentual da solução (ex: K = 10% elementos)

Para a construção (semi) gulosa, os elementos devem ser ranqueados conforme uma função heurística f.

↳ Se f não depende da solução parcial → f é estática,
↳ Caso contrário → f é dinâmica / adaptativa.

↳ Neste caso, f é recalculada a cada iteração.

EXEMPLO: Cobertura de Vértices



Representação: Vetor binário
Elementos: {1, 2, 3, 4, 5, 6, 7}
Estratégia (G): gulosa
Função Heurística (f): grau do vértice

↳

Vértice	grau
1	4
2	2
3	3
4	5
5	2
6	3
7	1

Execução:

Iteração	Solução	f.o.	Completa?
1	{4}	1	x
2	{4, 1}	2	x
3	{4, 1, 3}	3	x
4	{4, 1, 3, 6}	4	✓

↳ Desempate?
- Escolha aleatória
- Segunda função heurística

↳ Executar construção gulosa com f = número de arestas não cobertas ligadas ao vértice
↳ função dinâmica/adaptativa!

Construção para o Problema da Mochila

Instância: Capacidade = 113

Item	Weight	Profit
1	32 lbs.	727 $
2	40 lbs.	763 $
3	44 lbs.	60 $
4	20 lbs.	606 $
5	1 lbs.	45 $
6	29 lbs.	370 $
7	3 lbs.	414 $
8	13 lbs.	880 $
9	6 lbs.	133 $
10	39 lbs.	820 $

Sol. ótima

$(1,0,0,1,0,0,1,1,1,1)$

$z = 3580$

Peso total: 113

Heurística 1: Seleciona o item de maior profit que cabe na mochila;

Repete até não haver mais itens que caibam na mochila.

↳ Resultado: $(0,1,0,1,1,0,0,1,0,1)$, $z = 3114$

Ordenando pelo Profit:

Heurística 1

Item	Weight	Profit		Σ_7	Σ_w
8	13 lbs.	880 $	✓	880	13
10	39 lbs.	820 $	✓	1700	52
2	40 lbs.	763 $	✓	2463	92
1	32 lbs.	727 $	✗	–	–
4	20 lbs.	606 $	✓	3069	112
7	3 lbs.	414 $	✗	–	–
6	29 lbs.	370 $	✗	–	–
9	6 lbs.	133 $	✗	–	–
3	44 lbs.	60 $	✗	–	–
5	1 lbs.	45 $	✓	3114	113

Resultado:

Step	Item	Add ?	Total weight	Total profit
1	8	yes	13 lbs.	880 $
2	10	yes	52 lbs.	1700 $
3	2	yes	92 lbs.	2463 $
4	1	no		
5	4	yes	112 lbs.	3069 $
6	7	no		
7	6	no		
8	9	no		
9	3	no		
10	5	yes	113 lbs.	3114 $

$\delta = (1,1,1,0,1,0,0,0,0,1)$

$z = 3114$

Peso total = 113

Heurística 2: Igual à anterior, mas seleciona os itens pela sua eficiência, i.e. pelo custo-benefício $\frac{\text{Profit}}{\text{Weight}}$.

Ordenando pela eficiência:

Item	Weight	Profit	Efficiency
7	3 lbs.	414 $	138.00 $ per pound
8	13 lbs.	880 $	67.69 $ per pound
5	1 lbs.	45 $	45.00 $ per pound
4	20 lbs.	606 $	30.30 $ per pound
1	32 lbs.	727 $	22.72 $ per pound
9	6 lbs.	133 $	22.17 $ per pound
10	39 lbs.	820 $	21.03 $ per pound
2	40 lbs.	763 $	19.08 $ per pound
6	29 lbs.	370 $	12.76 $ per pound
3	44 lbs.	60 $	1.36 $ per pound

Heurística 2

	$\sum v$	$\sum w$
✓	414	3
✓	1294	16
✓	1339	17
✓	1945	37
✓	2672	69
✓	2805	75
✗	—	—
✗	—	—
✓	3175	104
✗	—	—

Resultado:

Step	Item	Add ?	Total weight	Total profit
1	7	yes	3 lbs.	414 $
2	8	yes	16 lbs.	1294 $
3	5	yes	17 lbs.	1339 $
4	4	yes	37 lbs.	1945 $
5	1	yes	69 lbs.	2672 $
6	9	yes	75 lbs.	2805 $
7	10	no		
8	2	no		
9	6	yes	104 lbs.	3175 $
10	3	no		

$s = (1, 1, 1, 1, 1, 1, 0, 0, 1, 0)$

$z = 3175$

Peso total = 104

Construção Repetida: Constrói N soluções usando uma heurística construtiva não-determinística (semi-gulosa). Ao final, retorna a melhor solução. (incumbente)

```
Construção Repetida (N, Construção):
    s* ← Construção
    Para i ← 1, ..., N Para
        s ← Construção
        Se s é melhor que s* então
            s* ← s
    Retorna s*  → solução incumbente
```

Em vez de um número de iterações N, pode-se usar um tempo limite de execução, ou outro critério de terminação, como número máximo de iterações em estagnação (i.e., sem melhoria).

Guloso Iterado (Iterated Greedy): Iterativamente aplica uma destruição à solução atual, seguida de uma etapa de Reconstrução. O parâmetro D define o tamanho da destruição.

```
Iterated Greedy (D, Destruição, Construção, N):
    s* ← Construção
    s ← s*
    PARA i ← 1, ..., N FAÇA
        s ← Destruição (s, D)
        s ← Construção (s)
        SE s é melhor que s* ENTÃO
            s* ← s
    RETORNA s*
```

→ Ou outro critério

→ Reconstrução a partir de uma solução parcial. (destruída parcialmente)

→ Melhor solução encontrada (incumbente)

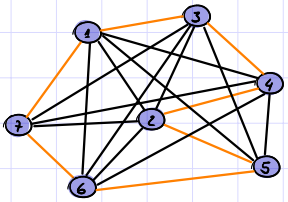
Destruição: Remove D elementos da solução

↳ Ex: Remove a cor de D vértices → Destruição
e atribui novas cores → Reconstrução
(Coloração de vértices)

Ex: Seta D itens para 0,5 → Destruição
e atribui novos valores → Reconstrução
(problema da mochila)

Resumo:
Construção repetida → Construção independente
Iterated greedy → Usa informações da solução construída antes → Construção dependente.

EXTRA: HEURÍSTICA CONSTRUTIVA PARA O PROBLEMA DO CAIXEIRO VIAJANTE (PCV)



Dado um grafo $G = (V, A)$, geralmente completo, encontra o ciclo hamiltoniano de menor custo. Ou seja, um caminho que parte de um vértice e retorna a ele, visitando cada vértice exatamente uma vez.

↳ PCV é NP-Completo!

HEURÍSTICA PARA PCV

Vizinho mais próximo: (Nearest Neighbor)

Complexidade? $O(n^2)$

Atual ← vértice aleatório
Enquanto há vértices não selecionados:
- Próximo ← vértice mais próximo do atual
- Insere arco (Atual, próximo) na solução
- Atual ← Próximo

Retorna solução

↳ Apesar de não seguir o pseudocódigo discutido, o algoritmo possui seus elementos:
- ↳ Elementos candidatos: vértices a serem incluídos
- ↳ Função heurística: distância do último vértice incluído

↳ Logo, trata-se de uma heurística construtiva!