



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
FLUMINENSE

Sistemas de Informação

Inteligência Computacional

Eduardo Rodrigues

eduardo.rodrigues@iff.edu.br

Objetivos

- ▶ Entender o conceito dos algoritmos de otimização;
- ▶ Diferenciar heurística vs metaheurística;
- ▶ Implementar problema de maximização de lucro.

Algoritmos de otimização

Algoritmos de Otimização

► Objetivo dos algoritmos de otimização

- Busca-se maximizar ou minimizar alguma variável do problema;
- Exemplo: minimizar custos e maximizar lucros de uma empresa.

► Heurísticas x metaheurísticas

- As heurísticas são informações específicas de um determinado problema que busca fazer uma aproximação que simplifique a solução. (ex: distância em linha reta entre duas cidades);
- As **metaheurísticas** são métodos heurísticos para resolver de forma genérica problemas de otimização.

Algoritmos de Otimização

► Metaheurística

- As metaheurísticas são procedimentos que guiam outras **heurísticas**, ou seja, **procedimentos computacionais**, usualmente de busca local, explorando o espaço de soluções além do **ótimo local**.
- As metaheurísticas consideram boas características das soluções encontradas para explorar novas regiões promissoras.

Algoritmos de Otimização

► Metaheurística

- Partem de uma ou mais soluções iniciais e tentam melhorar a(s) solução(ões) por meio de modificações realizada(s) na(s) mesma(s).
- São necessários os elementos:
 - 1. Procedimento para gerar a solução inicial;
 - 2. Função objetivo ou função de avaliação;
 - 3. Método de modificação da(s) solução(ões) corrente(s) para encontrar uma solução que pode ser melhor do que a melhor solução conhecida.

Algoritmos de Otimização

► Metaheurística

- Problema de maximização de lucro para uma empresa que trabalha com transporte de objetos;
- Um caminhão possui capacidade de transporte $t = 2000\text{kg}$ e há uma lista de 5 possíveis produtos com os respectivos custos (R\$) e massas (Kg).

Objeto(j)	1	2	3	4	5
Peso (Kg)	400	500	700	900	600
Lucro (R\$)	200	200	300	400	400

- **QUESTÃO:** Como escolher, baseado no custo e peso dos produtos, a melhor combinação de objetivos para maximizar o lucro da empresa?

Algoritmos de Otimização

► Modelagem do problema

- **SOLUÇÃO:** adicionemos um objeto no caminhão a cada passo, o objeto mais valioso por unidade de peso e que não ultrapasse a capacidade de transporte do caminhão.

Objeto(j)	1	2	3	4	5
Peso (Kg)	400	500	700	900	600
Lucro (R\$)	200	200	300	400	400
Lucro / Peso	0.5	0.4	0.43	0.44	0.66

Algoritmos de Otimização

► Modelagem do problema

- **SOLUÇÃO:** Reordenando os objetos nós teremos, baseado na relação $\text{LUCRO(R\$)} / \text{PESO(Kg)}$.

Objeto(j)	5	1	4	3	2
Peso (Kg)	600	400	900	700	500
Lucro (R\$)	400	200	400	300	200
Lucro / Peso	0.66	0.50	0.44	0.43	0.40

- Representamos uma solução s por um vetor binário de n posições.

Algoritmos de Otimização

► Executando o algoritmo - Sequência de passos

- **PASSO 1:** Adicionemos, o objeto 5, que tem a maior relação lucro/peso.

- $s: 0\ 0\ 0\ 0\ 1$

- $f(s) = 400$

- Peso atual do caminhão = $600 < t=2000$

- **PASSO 2:** Adicionemos, o objeto 1, que tem a maior relação lucro/peso na sequência.

- $s: 1\ 0\ 0\ 0\ 1$

- $f(s) = 600$

- Peso atual do caminhão = $1000 < t=2000$

- **PASSO 3:** Adicionemos, o objeto 4, que tem a maior relação lucro/peso na sequência.

- $s: 1\ 0\ 0\ 1\ 1$

- $f(s) = 1000$

- Peso atual do caminhão = $1900 < t=2000$

- **PASSO 4:** O próximo objeto a ser adicionado deveria ser o 3. No entanto, esta alocação faria superar a capacidade do caminhão. Neste caso, devemos tentar alocar o próximo objeto na sequência com a maior relação de benefício, que é o objeto 2. Como também ultrapassaríamos a capacidade do caminhão e não há mais objetos a serem inseridos, concluímos que a solução* é: $s(10011)$ com $f(s^*) = 1000$.

Implementação

Algoritmos de Otimização - Implementação

▶ Classe: Objeto

▶ Atributos

- ▶ **Id:** identificador único do objeto;
- ▶ **Peso:** massa medida em Kg;
- ▶ **Lucro:** valor em R\$ que o produto proporciona de lucro.

▶ Métodos

- ▶ **Construtores:** padrão e com argumentos;
- ▶ **Get's e Set's;**
- ▶ **Comparator:** compara dois objetos a partir do seu peso. Usado com o *Collection sort* para ordenar *Array* de objetos baseado em um parâmetro específico.

Algoritmos de Otimização - Implementação

▶ Classe: Veículo

▶ Atributos

- ▶ **Capacidade:** Kg total que o veículo suporta;
- ▶ **Objetos:** *ArrayList* para representar os Objetos que serão transportados.

▶ Métodos

- ▶ **Construtores:** padrão e com argumentos;
- ▶ **Get's e Set's;**
- ▶ **Massa total dos objetos:** soma os Kg's de todos os objetos que estiverem na lista do veículo;
- ▶ **Inserir Objeto:** adiciona o Objeto na lista do veículo;
- ▶ **Imprime Lista de Objetos:** exibe informações de todos os objetos dentro do veículo.

Algoritmos de Otimização - Implementação

► Classe: **MaximizaLucro**

► Atributos

- **Lista de objetos:** *ArrayList* de objetos que podem ser transportados;
- **Solução:** vetor de inteiros para armazenar 1's e 0's indicando objetos que foram inseridos no veículo;
- **Veículo:** Objeto caminhão que irá transportar os produtos.

► Métodos

- **Construtor:** declaramos e inicializamos: os objetos, inserimos na lista de objetos, o caminhão e o vetor solução.
- **Função de otimização:** ordena os objetos de acordo com a taxa lucro/peso, insere os elementos da lista de objetos do veículo enquanto a capacidade não for superada.
- **Imprime lista de objetos:** exibe informações sobre os objetos disponíveis para ser transportados;
- **Imprime vetor solução:** exibe zeros e uns para representar objetos selecionados para o transporte;
- **Calcula o lucro total:** a partir do vetor solução, calcula o lucro total para os objetos selecionados.

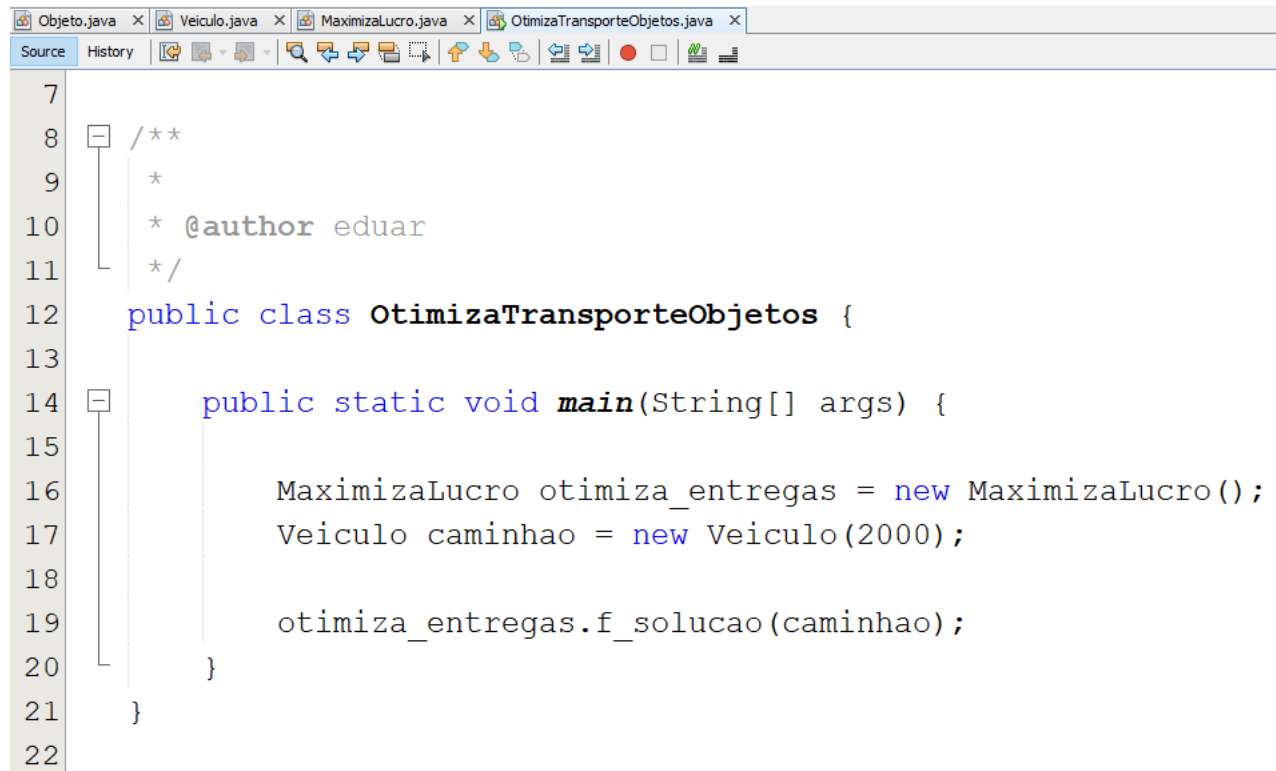
Algoritmos de Otimização - Implementação

► Construtor - MaximizaLucro

```
Objeto.java x Veiculo.java x MaximizaLucro.java x OtimizaTransporteObjetos.java x
Source History
21 public MaximizaLucro() {
22     // Criamos veículo com capacidade de 2000kg e uma lista vazia de objetos
23     caminhao = new Veiculo(2000);
24
25     // Criamos lista de objetos com seus respectivos parâmetros
26     Objeto obj1 = new Objeto(1, 400, 200);
27     Objeto obj2 = new Objeto(2, 500, 200);
28     Objeto obj3 = new Objeto(3, 700, 300);
29     Objeto obj4 = new Objeto(4, 900, 400);
30     Objeto obj5 = new Objeto(5, 600, 400);
31
32     // Alocamos uma lista de objetos não ordenada
33     lista_de_objetos = new ArrayList<>();
34     lista_de_objetos.add(obj1);
35     lista_de_objetos.add(obj2);
36     lista_de_objetos.add(obj3);
37     lista_de_objetos.add(obj4);
38     lista_de_objetos.add(obj5);
39
40     // Alocamos o vetor binário para a solução do problema
41     this.solucao = new int[lista_de_objetos.size()];
42     imprimeListaObjetos();
43 }
```

Algoritmos de Otimização - Implementação

► Classe - Testes



```
Objeto.java x Veiculo.java x MaximizaLucro.java x OtimizaTransporteObjetos.java x
Source History
7
8  /**
9   *
10  * @author eduar
11  */
12  public class OtimizaTransporteObjetos {
13
14      public static void main(String[] args) {
15
16          MaximizaLucro otimiza_entregas = new MaximizaLucro();
17          Veiculo caminhao = new Veiculo(2000);
18
19          otimiza_entregas.f_solucao(caminhao);
20      }
21  }
22
```


Algoritmos de Otimização - Implementação

► Classe - Testes

```
Output - Run (Aula_IA_MetodosOtimizacao) x
cd C:\Users\eduar\OneDrive\Documentos\NetBeansProjects\Aula_IA_MetodosOtimizacao; "JAVA_HOME=C:
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories c
Scanning for projects...
|
-----< com.mycompany:Aula_IA_MetodosOtimizacao >-----
Building Aula_IA_MetodosOtimizacao 1.0-SNAPSHOT
-----[ jar ]-----
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Aula_IA_MetodosOtimizacao ---
Lista de Objetos:
1(0.5) | 2(0.4) | 3(0.42857142857142855) | 4(0.4444444444444444) | 5(0.6666666666666666)

Lista de Objetos Carregados:
5(0.6666666666666666) | 1(0.5) | 4(0.4444444444444444)

VETOR SOLUÇÃO:
1 | 0 | 0 | 1 | 1 |

LUCRO TOTAL = R$1000.0
-----
```

Bibliografia Básica

- ▶ Braga, A. P. Carvalho, A. P. L.; Ludermir, T. B. - **Redes neurais artificiais - teoria e aplicações** , Editora LTC, 1ª. Edição, 2000.
- ▶ Shaw, I. S. Simões, M. G. - **Controle e Modelagem Fuzzy** , Editora Edgard Blucher Ltda, 1ª. Edição, 2001.

Bibliografia Complementar

- ▶ Barreto, J. M. - **Inteligência artificial no limiar do Século XXI - abordagem híbrida: simbólica, conexionista e evolucionária**, Editora rrr UFSC Florianópolis, 2ª. Edição, 1999.
- ▶ Jyh-Shing, Roger Jang, Chuen-Tsai Sun, Eiji Mizutani. **Neuro-Fuzzy and Soft Computing**. PrenticeHall, 1997.

Obrigado!

► Até o próximo encontro.