

**UNIVERSIDADE FEDERAL DE MINAS GERAIS**  
**Escola de Engenharia**  
**Curso de Bacharelado em Engenharia de Sistemas**

Raphael Henrique Braga Leivas 2020028101

Milton Pereira Bravo Neto 2018072549

Daniel Felipe de Almeida Araújo 2023422617

**Trabalho Computacional 01 – Teoria da Decisão (ELE088)**

## SUMÁRIO

<b>1</b>	<b>MODELAGEM</b>	<b>3</b>
1.1	Problema 1	3
1.2	Problema 2	3
1.3	Modelagem Multiobjetivo	4
<b>2</b>	<b>ALGORITMOS</b>	<b>5</b>
2.1	BVNS - Basic Variable Neighborhood Search	5
2.2	Estratégias de Refinamento	6
2.3	Heurística Construtiva	7
<b>3</b>	<b>RESULTADOS E DISCUSSÃO</b>	<b>8</b>
3.1	Problema 1	8
3.2	Problema 2	8
3.3	Comparação de Estratégias de Refinamento	9
<b>4</b>	<b>REFERÊNCIAS</b>	<b>10</b>

## 1 MODELAGEM

Temos que modelar dois problemas mono-objetivos:

- Problema 1: minimização do custo de manutenção total  $f_1(\cdot)$
- Problema 2: minimização do custo esperado de falha total  $f_2(\cdot)$

### 1.1 Problema 1

Temos essencialmente um problema de designação simples. Seja  $N$  o número de equipamentos e  $J$  o número de políticas de manutenção, definimos a variável de decisão  $x_{ij}$  por

$$x_{ij} : \text{ se o equipamento } i \text{ executa a manutenção } j \quad (1.1)$$

onde

$$x_{ij} \in \{0, 1\} \quad , \quad i = \{1, 2, \dots, N\} \quad , \quad j = \{1, 2, \dots, J\}$$

Para a função objetivo, seja  $c_j$  o custo de executar a manutenção  $j$ . Note que esse custo independe do equipamento  $i$  que estamos executando a manutenção. Temos a função objetivo

$$\min f_1 = \sum_{i=1}^N \sum_{j=1}^J c_j x_{ij} \quad (1.2)$$

sujeito a

$$\sum_{j=1}^J x_{ij} = 1 \quad , \quad \forall i = 1, 2, \dots, N \quad (1.3)$$

A Equação 1.3 indica que todo equipamento executa exatamente uma política de manutenção. Além disso, note que solução de Equação 1.3 é trivial: basta escolher o plano de manutenção com o menor custo para todos os equipamentos.

### 1.2 Problema 2

O custo da falha de cada equipamento é dado pelo produto da probabilidade de falha  $p_{ij}$  pelo custo da falha do equipamento, dada por  $d_i$ . Assim, temos

$$\min f_2 = \sum_{i=1}^N \sum_{j=1}^J p_{ij} d_i x_{ij} \quad (1.4)$$

onde

$$x_{ij} \in \{0, 1\} \quad , \quad i = \{1, 2, \dots, N\} \quad , \quad j = \{1, 2, \dots, J\}$$

$$p_{ij} = \frac{F_i(t_0 + k_j \Delta t) - F_i(t_0)}{1 - F_i(t_0)} \quad , \quad F_i(t) = 1 - \exp \left[ - \left( \frac{t}{\eta_i} \right)^{\beta_i} \right]$$

sujeito a

$$\sum_{j=1}^J x_{ij} = 1 \quad , \quad \forall i = 1, 2, \dots, N \quad (1.5)$$

Note que na Equação 1.4 temos essencialmente um problema de programação linear inteira.

### 1.3 Modelagem Multiobjetivo

Juntando as modelagens dos problemas mono-objetivos acima, temos o a modelagem multiobjetivo do problema.

$$\begin{aligned} \min f_1 &= \sum_{i=1}^N \sum_{j=1}^J c_j x_{ij} \\ \min f_2 &= \sum_{i=1}^N \sum_{j=1}^J p_{ij} d_i x_{ij} \end{aligned}$$

$x_{ij}$  : se o equipamento  $i$  executa a manutenção  $j$

sujeito a

$$\sum_{j=1}^J x_{ij} = 1 \quad , \quad \forall i = 1, 2, \dots, N$$

$$x_{ij} \in \{0, 1\} \quad , \quad i = \{1, 2, \dots, N\} \quad , \quad j = \{1, 2, \dots, J\}$$

## 2 ALGORITMOS

### 2.1 BVNS - Basic Variable Neighborhood Search

O Algoritmo 1 mostra a versão do BNVS implementada no trabalho.

---

**Algoritmo 1** BVNS implementado no trabalho.

---

```

1: procedure BVNS( $\mathbf{x}$ ,  $k_{\max}$ )
2:   while num_sol_avaliables < max_sol_avaliables do
3:      $k \leftarrow 1$ 
4:     while  $k < k_{\max}$  do
5:        $\mathbf{x}' \leftarrow \text{SHAKE}(\mathbf{x}, k)$ 
6:        $\mathbf{x}'' \leftarrow \text{FIRSTIMPROVEMENT}(\mathbf{x}, \mathbf{x}', k)$ 
7:        $\mathbf{x}, k \leftarrow \text{NEIGHBORHOODCHANGE}(\mathbf{x}, \mathbf{x}'', k)$ 
8:     end while
9:   end while
10: end procedure

```

---

O Algoritmo 2 mostra a função Shake. Nela estão definidas as três estruturas de vizinhança escolhidas para implementação. As duas primeiras são vizinhanças de refinamento, mas com abordagens diferentes. E a terceira é uma vizinhança de perturbação para buscar sair de mínimos locais:

- A primeira estrutura é o que chamamos de um movimento de *1-swap*, onde é escolhido aleatoriamente um equipamento e trocado seu plano para um dos outros dois restantes, a escolha do plano também é aleatória.
- A segunda estrutura é a troca ou permutação dos planos de dois equipamentos diferentes escolhidos também aleatoriamente.
- A terceira estrutura por sua vez, altera um bloco de 50 equipamentos em sequência, onde o início do bloco é aleatório. Nesse bloco é avaliado qual o plano mais comum e troca-se o plano de manutenção de todos os integrantes do bloco para um mesmo plano, diferente do mais comum encontrado anteriormente.

---

**Algoritmo 2** Função Shake.

▷ Gera uma solução aleatória na  $k$ -ésima estrutura de vizinhança.

```

1: procedure SHAKE( $\mathbf{x}$ ,  $k$ )
2:   if  $k = 1$  then
3:      $\mathbf{y} \leftarrow$  1-swap
4:   end if
5:   if  $k = 2$  then
6:      $\mathbf{y} \leftarrow$  Permutação de dois planos de manutenção
7:   end if
8:   if  $k = 3$  then
9:      $\mathbf{y} \leftarrow$  Mudança de um bloco de equipamentos para outro plano
10:  end if

11:  return  $\mathbf{y}$ 
12: end procedure

```

---

## 2.2 Estratégias de Refinamento

O Algoritmo 3 mostra a função de busca local implementada após gerar uma solução aleatória com o Shake. Ela basicamente realiza uma busca em até  $N = 100$  vizinhos à solução do Shake, e retorna a primeira solução cujo valor da função objetivo é menor do que o valor de  $f$  na solução do Shake.

---

**Algoritmo 3** Função FirstImprovement.

▷ Busca uma primeira solução na vizinhança de  $\mathbf{x}'$  melhor que  $\mathbf{x}'$ .

```

1: procedure FIRSTIMPROVEMENT( $\mathbf{x}'$ ,  $k$ )
2:    $N \leftarrow 100$ 
3:   for all  $i$  in range( $N$ ) do
4:      $\mathbf{x}'' \leftarrow$  SHAKE( $\mathbf{x}'$ ,  $k$ )
5:     if  $f(\mathbf{x}'') < f(\mathbf{x}')$  then
6:       return  $\mathbf{x}''$ 
7:     end if
8:   end for

9:   return  $\mathbf{x}'$ 
10: end procedure

```

---

É possível fazer uma pequena modificação no Algoritmo 3 para obter o Best Improvement, exibido no Algoritmo 4. Note que essa função sempre executa as  $N$  buscas por uma melhor solução, e portanto o código é mais caro computacionalmente que no Algoritmo 3. No entanto, em geral, a solução encontrada pelo BestImprovement será melhor que a do FirstImprovement.

---

**Algoritmo 4** Função BestImprovement.

▷ Busca a melhor solução na vizinhança de  $\mathbf{x}'$  melhor que  $\mathbf{x}'$ .

```

1: procedure BESTIMPROVEMENT( $\mathbf{x}'$ ,  $k$ )
2:    $N \leftarrow 100$ 
3:    $\mathbf{x\_melhor} \leftarrow \mathbf{x}'$ 
4:   for all  $i$  in range( $N$ ) do
5:      $\mathbf{x}'' \leftarrow \text{SHAKE}(\mathbf{x}', k)$ 
6:     if  $f(\mathbf{x}'') < f(\mathbf{x\_melhor})$  then
7:        $\mathbf{x\_melhor} \leftarrow \mathbf{x}''$ 
8:     end if
9:   end for

10:  return  $\mathbf{x\_melhor}$ 
11: end procedure

```

---

### 2.3 Heurística Construtiva

O Algoritmo 5 mostra a heurística construtiva utilizada para a criação da solução inicial. Basicamente o problema se reduz em escolher um plano de manutenção, dentre os três disponíveis, para cada equipamento minimizando o custo da manutenção e o custo de falha dos equipamentos. A minimização do custo da manutenção se dá escolhendo a manutenção mais barata para todos os equipamentos, e a minimização do custo de falha escolhendo a manutenção mais cara.

Olhando para o segundo problema, temos a matriz de custos de falha  $p_{ij}d_i$  onde  $i$  é cada equipamento e  $j$  os planos de manutenção, para cada equipamento  $i$  fixo avalia-se a variância de  $p_{ij}d_i$  e caso esse valor seja maior que o limiar de 0.5 escolhe a manutenção mais cara para compor a solução inicial daquele equipamento, caso seja menor que o limiar é escolhida a manutenção mais barata.

A lógica envolvida é que se o custo de falha não varia tanto para aquele equipamento, não é necessário a manutenção mais cara.

---

**Algoritmo 5** Heurística construtiva para gerar a solução inicial.

```

1: procedure SOLUCAOINICIAL()
2:    $\mathbf{x} \leftarrow$  Solução aleatória
3:   for all  $i$  in  $\mathbf{x}$  do
4:     if  $\text{variancia}(p_{ij}d_i) \geq \text{limiar}$  then
5:        $\mathbf{x}[i] \leftarrow$  Manutenção mais cara
6:     else
7:        $\mathbf{x}[i] \leftarrow$  Manutenção mais barata
8:     end if
9:   end for

10:  return  $\mathbf{x}$ 
11: end procedure

```

---

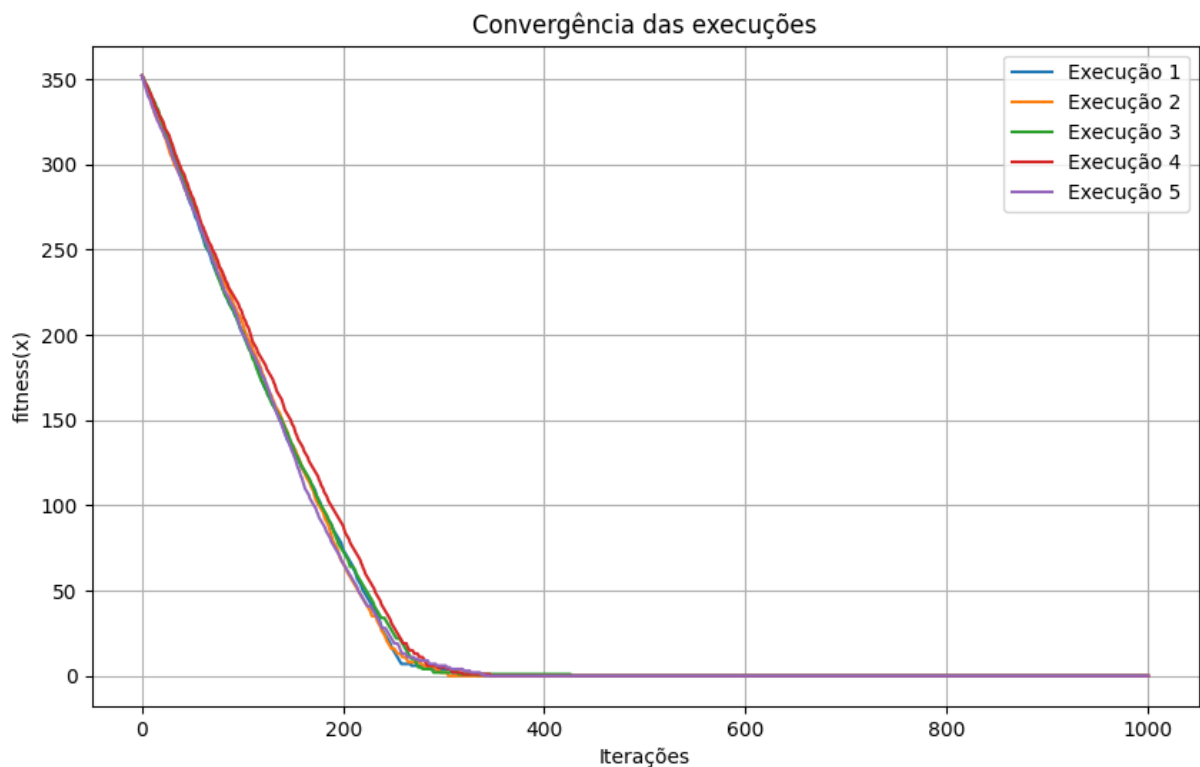
### 3 RESULTADOS E DISCUSSÃO

#### 3.1 Problema 1

A Figura 1 mostra a convergência de 5 execuções do BNVS para o Problema 1 Isolado. A solução final encontrada foi

$$x^* = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \quad f(x^*) = 0 \pm 0$$

Figura 1 – Convergência do BNVS implementado para o Problema 1 Isolado.



Fonte: elaboração própria.

Vemos que o algoritmo converge após 300 iterações para a solução ótima global, que já sabemos a priori uma vez que o problema modelado tem solução trivial. Cada iteração gastou em média 5 segundos para executar.

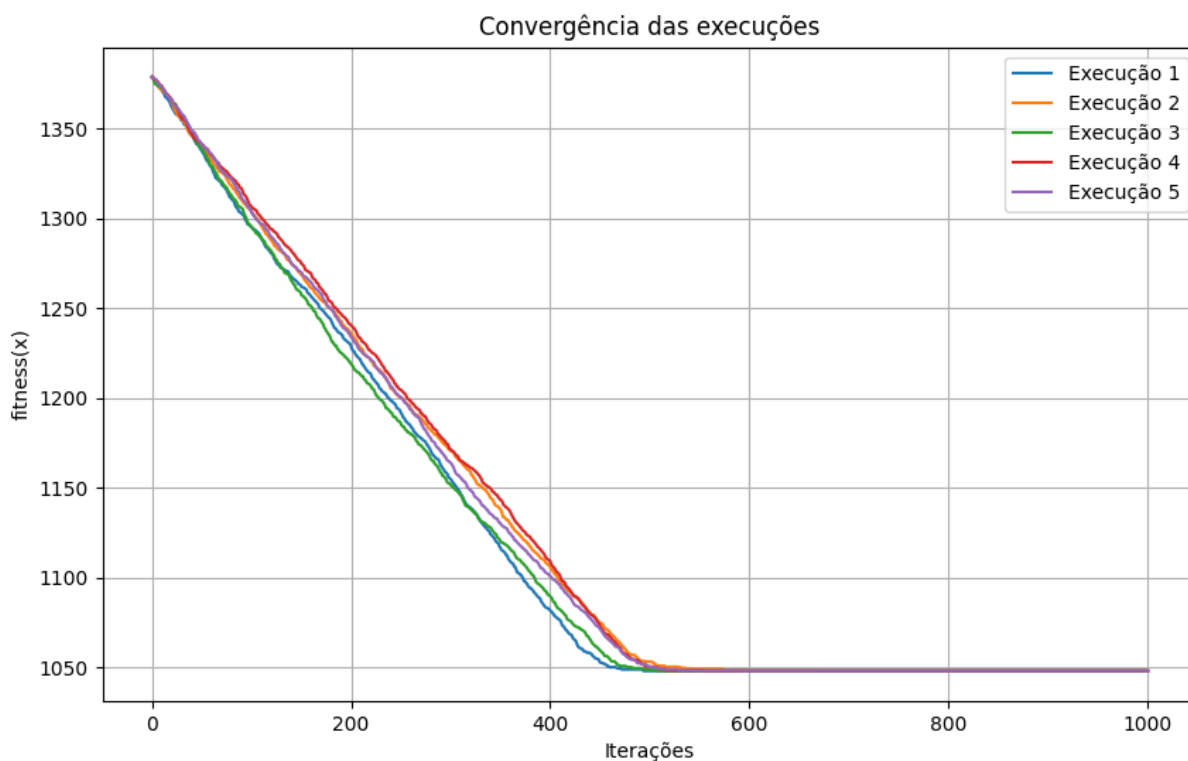
#### 3.2 Problema 2

A Figura 2 mostra a convergência de 5 execuções para o Problema 2. A solução final encontrada foi

$$x^* = \begin{bmatrix} 2 & 2 & 2 & \cdots & 2 \end{bmatrix}, \quad f(x^*) = 1048.2 \pm 0$$



Figura 2 – Convergência do BNVS implementado para o Problema 1 Isolado.



Fonte: elaboração própria.

De forma similar ao Problema 2, vemos que o algoritmo converge após 500 iterações para a solução ótima global dada pelo Simplex. Cada iteração gastou em média 12 segundos para executar.

### 3.3 Comparação de Estratégias de Refinamento

As convergências das Figuras 1 e 2 foram obtidas usando o BVNS com o FirstImprovement. Com o BestImprovement, obtemos as convergências das Figuras completas

Vemos que com o BestImprovement o algoritmo converge com menos iterações, no entanto ele gasta mais tempo para executar uma vez que o custo computacional de cada iteração é maior. Não obstante, ambas estratégias convergiram para a mesma solução ótima global.

#### **4 REFERÊNCIAS**

M. Gendreau, J.-Y. Potvin (eds.), *Handbook of Metaheuristics*, Springer, 2nd ed., 2010.