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 02 – Teoria da Decisão (ELE088)

Belo Horizonte
2025

SUMÁRIO

1 MODELAGEM MULTIOBJETIVO

No TC01, modelamos duas funções objetivo:

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

definidas por

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

$$\min f_2 = \sum_{i=1}^{N} \sum_{j=1}^{J} p_{ij} \, d_i \, x_{ij}$$

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

sujeito a

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

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

Considerando agora esse problema multiobjeitvo, podemos aplicar duas abordagens escalares para obter a fronteira Pareto-ótima no espaço de objetivos, descritas a seguir.

1.1 Formulação Soma Ponderada P_w

Seja $0 \leq w \leq 1$ um peso qualquer gerado aleatóriamente de uma distribução uniforme no intervalo $[0, 1]$. Usando a abordagem da soma ponderada, podemos escrever nosso problema multiobjetivo na forma:

$$\min w f_1 + (1 - w) f_2$$

1.2 Formulação ϵ-Restrito P_ϵ

Com a abordagem do ϵ-Restrito, vamos minimizar apenas f_1 usando f_2 como restrição. Seja ϵ_2 um real qualquer tal que $\min f_2 \leq \epsilon_2 \leq \max f_2$. Temos

$$\min f_1 \tag{1.1}$$

sujeito a

$$f_2 \leq \epsilon_2$$

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

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

1.3 Normalização

Para garantir que as abordagens escalares sejam condizentes, precisamos normalizar f_1 e f_2 através de

$$f_1(\mathbf{x}) = \frac{f_1(\mathbf{x}) - \min f_1}{\max f_1 - \min f_1} \quad , \quad f_2(\mathbf{x}) = \frac{f_2(\mathbf{x}) - \min f_2}{\max f_2 - \min f_2}$$

A partir do tabalho realizado no TC01, já sabemos que

$$\min f_1 = 0 \quad , \quad \max f_1 = 1000 \quad , \quad \min f_2 = 1048.17 \quad , \quad \max f_2 = 1745.49$$

2 SOLUÇÃO VIA SIMPLEX

3 SOLUÇÃO VIA BVNS

4 ALGORITMOS

4.1 BVNS - Basic Variable Neighborhood Search

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

Algoritmo 1 BVNS implementado no trabalho.

```
procedure BVNS(x, k_max)
    while num_sol_avaliadas < max_sol_avaliadas do
        k ← 1
        while k < k_max do
            x' ← SHAKE(x, k)
            x'' ← FIRSTIMPROVEMENT(x, x', k)
            x, k ← NEIGHBORHOODCHANGE(x, x'', k)
        end while
    end while
end procedure
```

O Algoritmo 2 mostra a função Shake. Nela estão definidas as três estruturas de vizinhança escolhidas para implementação. As duas primeiras são vizinhaças 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 aleatorimente 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 aleatóriamente.
- 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.

```
procedure SHAKE(x, k)
    if k = 1 then
        y ← 1-swap
    end if
    if k = 2 then
        y ← Permutação de dois planos de manutenção
    end if
    if k = 3 then
        y ← Mudança de um bloco de equipamentos para outro plano
    end if

    return y
end procedure
```

4.2 Estratégias de Refinamento

O Algoritmo 3 mostra a função de busca local implementanda após gerar uma solução aleatória com o Shake. Ela basicamente realiza uma busca em até $N = 100$ vizinhos à solução inicial **x’** do Shake, e retorna a primeira solução **x”** cujo valor da solução objetivo é menor do que o valor do objetivo na solução inicial **x’** do Shake. Caso nenhuma solução melhor é encontrada, retorna a solução inicial **x’**.

Algoritmo 3 Função FirstImprovement.

▷ Busca uma primeira solução na vizinhança de **x’** melhor que **x’**.

```
procedure FIRSTIMPROVEMENT(x’, k)
    N ← 100
    for all i in range(N) do
        x” ← SHAKE(x’, k)
        if f(x”) < f(x’) then
            return x”
        end if
    end for

    return x’
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 **x'** melhor que **x'**.

procedure BESTIMPROVEMENT(**x'**, k)
    $N \leftarrow 100$
    x_melhor $\leftarrow$ **x'**
    for all i **in** range(N) **do**
        x" $\leftarrow$ SHAKE(**x'**, k)
        if f(**x"**) $<$ f(**x_melhor**) **then**
            x_melhor $\leftarrow$ **x"**
        end if
    end for
    return x_melhor
end procedure

4.3 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 manutencão. 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.

procedure SOLUCAOINICIAL()
    x $\leftarrow$ Solução aleatória
    for all i **in x do**
        if variancia($p_{ij}d_i$) $\geq$ limiar **then**
            x[i] $\leftarrow$ Manutenção mais cara
        else
            x[i] $\leftarrow$ Manutenção mais barata
        end if
    end for
    return x
end procedure

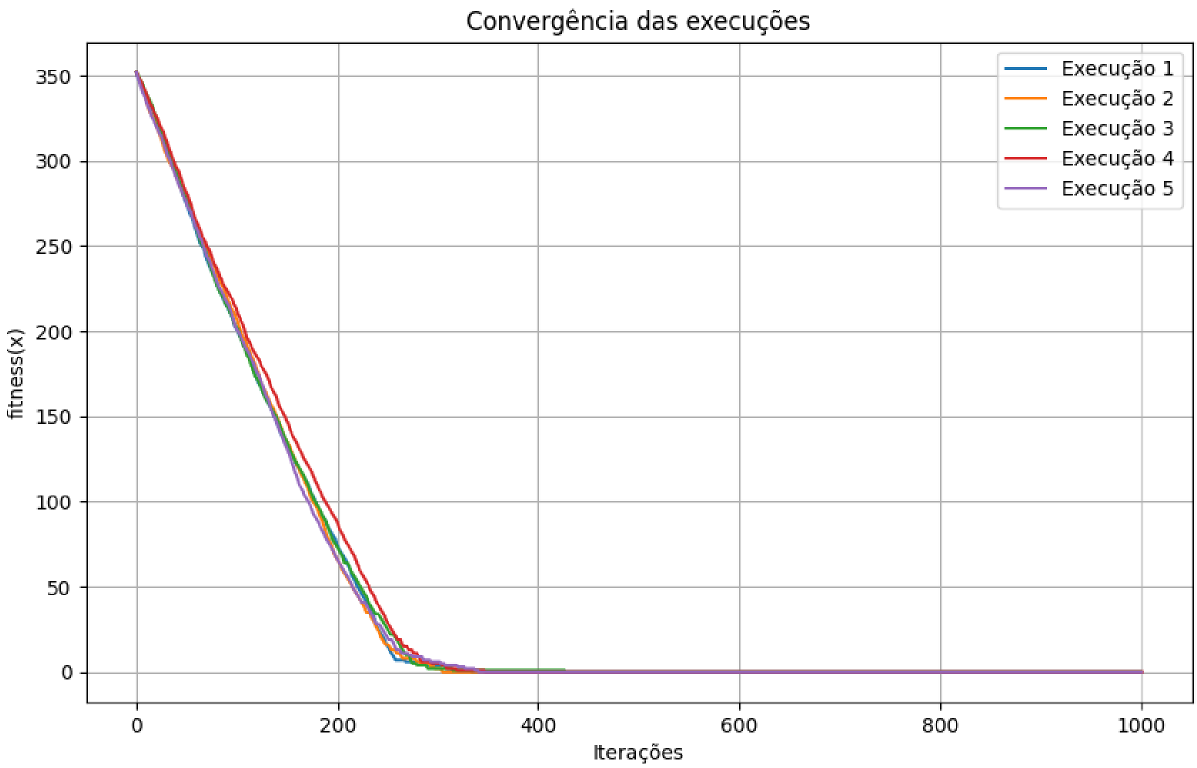
5 RESULTADOS E DISCUSSÃO

5.1 Problema 1

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

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

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



Fonte: elaboração própria.

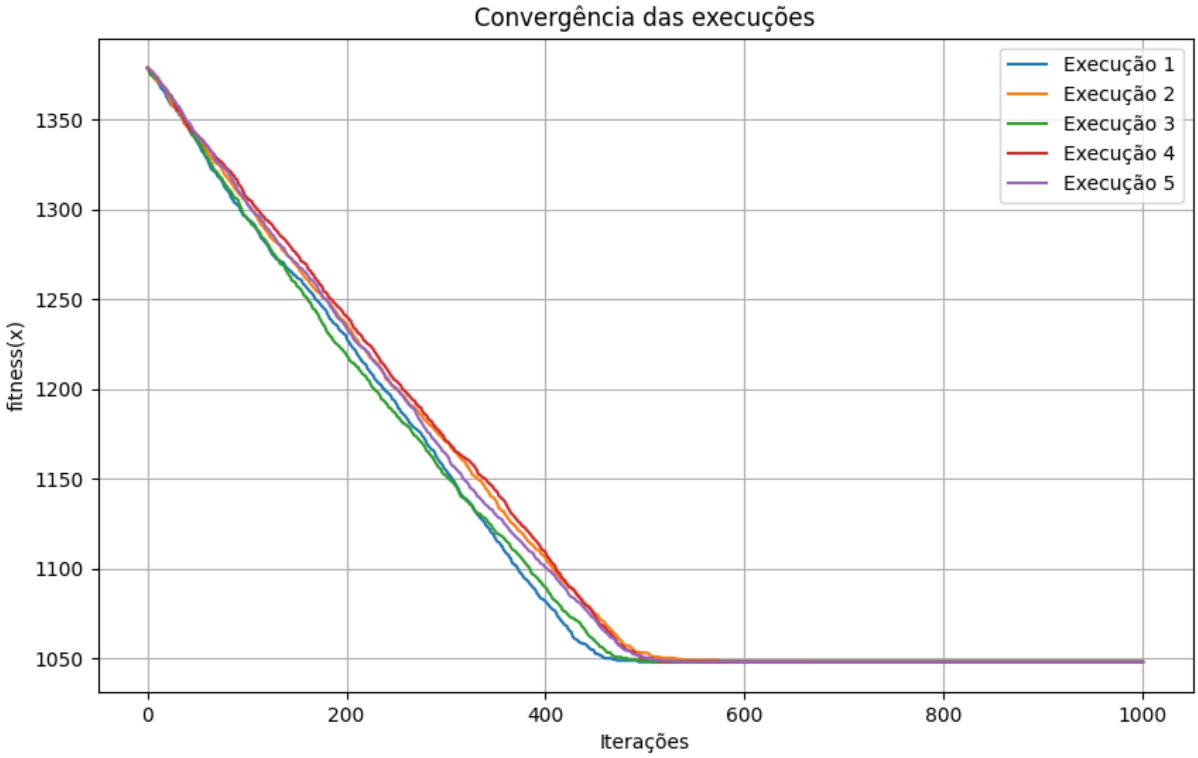
Vemos que o algoritmo converge após 300 iterações para a solução ótima global, que já sabemos a priori uma vez que o problema modelado tem solução trivial. Cada iteração gastou em média 5 segundos para executar.

5.2 Problema 2

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

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

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



Fonte: elaboração própria.

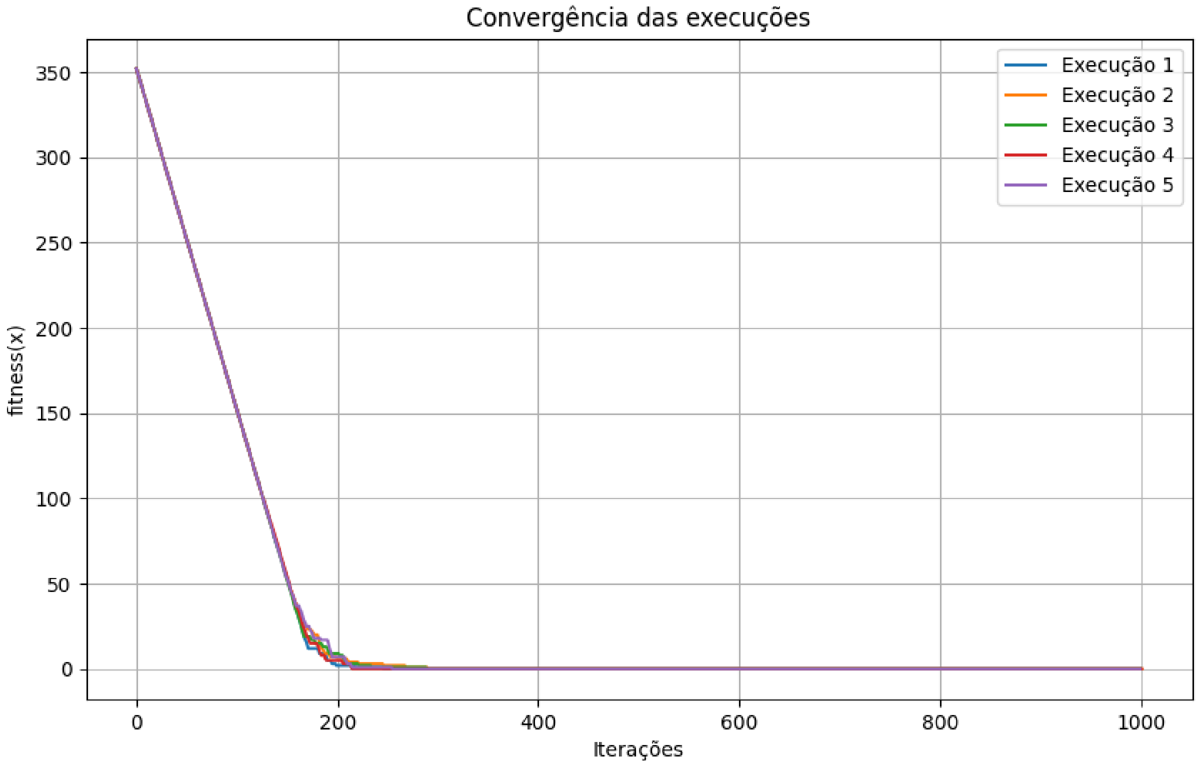
De forma similiar ao Problema 2, vemos que o algoritmo converge após 500 iterações para a solução ótima global dada pelo Simplex. Cada iteração gastou em média 12 segundos para executar.

5.3 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 3 e 4 para os Problemas 1 e 2, respectivamente.

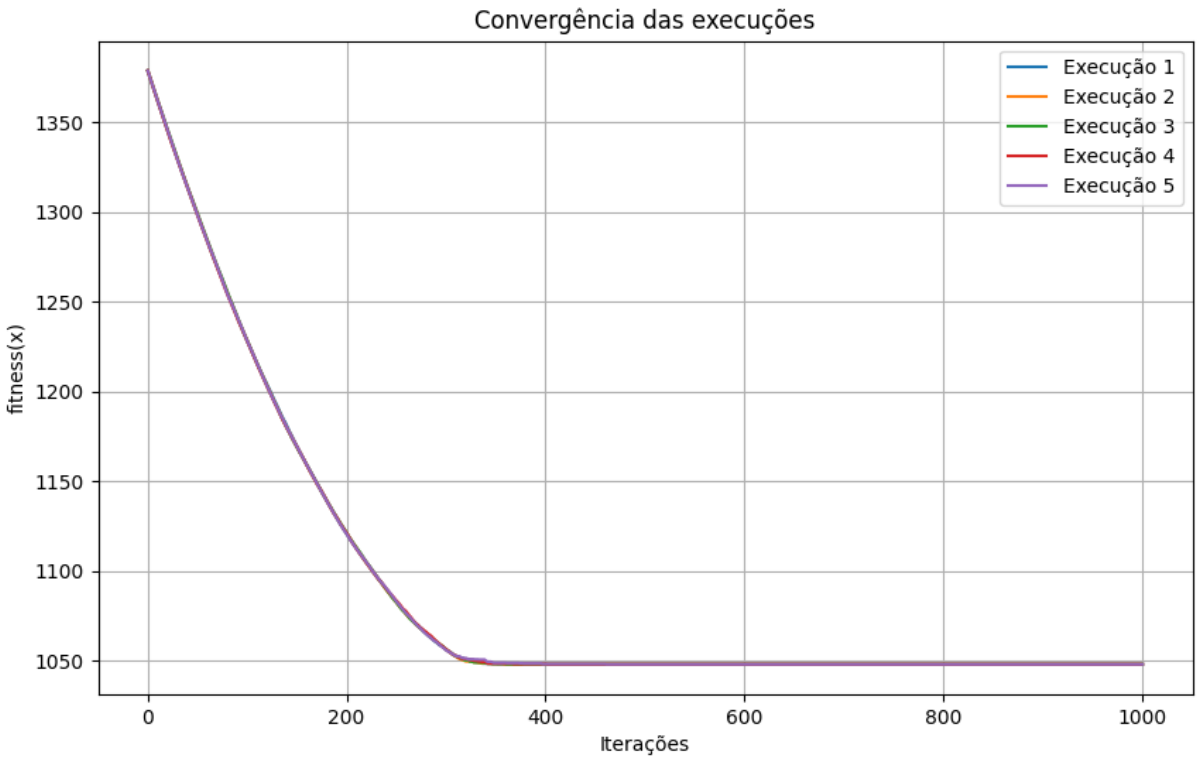
Vemos que com o BestImprovement o algoritmo converge com menos iterações quando comparado com o FirstImprovement: antes precisava de 300 e 500 iterações respectivamente; agora converge em 200 e 300. Contudo, ele gasta mais tempo para executar uma vez que o custo computacional de cada iteração é maior. Em particular, o Problema 2 gasta mais de 40 segundos por execução com o Best Improvement, sendo que antes gastava apenas 12. Não obstante, ambas estratégias convergiram para a mesma solução ótima global.

Figura 3 – Convergência do BVNS implementado para o Problema 1 usando a Best Improvement.



Fonte: elaboração própria.

Figura 4 – Convergência do BVNS implementado para o Problema 2 usando a Best Improvement.



Fonte: elaboração própria.

6 REFERÊNCIAS

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