Trabalho Computacional 3 - Teoria da Decisão (ELE088)

Daniel Felipe de Almeida Araújo
Universidade Federal de Minas Gerais
Matrícula: 2023422617

Milton Pereira Bravo Neto
Universidade Federal de Minas Gerais
Matrícula: 2018072549

Raphael Henrique Braga Leivas
Universidade Federal de Minas Gerais
Matrícula: 2020028101

***Abstract*— The abstract goes here.**

I. Introdução

This demo file is intended to serve as a “starter file" for IEEE conference papers produced under LaTeX using IEEEtran.cls version 1.6b and later.

May all your publication endeavors be successful.

mds
November 18, 2002

II. Metodologia

A. Modelagem do Problema

Inicialmente podemos ver o trabalho como sendo dois problemas mono-objetivo distintos:

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

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

$$x_{ij}: \text{ se o equipamento } i \text{ executa a manutenção } j \tag{1}$$

onde

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

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

$$\min f_1 = \sum_{i=1}^{N}\sum_{j=1}^{J} c_j x_{ij} \tag{2}$$

sujeito a

$$\sum_{j=1}^{J} x_{ij} = 1 \quad , \quad \forall i = 1,2,...,N \tag{3}$$

A 3 indica que todo equipamento executa exatamente uma política de manutenção. Além disso, note que solução da 2 é trivial: basta escolher o plano de manutenção com o menor custo para todos os equipamentos.

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

$$\min f_2 = \sum_{i=1}^{N}\sum_{j=1}^{J} p_{ij}\, d_i\, x_{ij} \tag{4}$$

onde

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

$$p_{ij} = \frac{F_i\left(t_0 + k_j \Delta t\right) - F_i\left(t_0\right)}{1 - F_i\left(t_0\right)} \tag{5}$$

$$F_i(t) = 1 - \exp\left[-\left(\frac{t}{\eta_i}\right)^{\beta_i}\right] \tag{6}$$

sujeito a

$$\sum_{j=1}^{J} x_{ij} = 1 \quad , \quad \forall i = 1,2,...,N \tag{7}$$

Note que na Equação 4 temos essencialmente um problema de programação linear inteira. Assim, é possível usar o método Simplex visto em Pesquisa Operacional para resolver esse problema com garantia de otimalidade. Usando o Simplex, a solução encontrada foi

$$\mathbf{x}^* = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 1 \end{bmatrix} \quad , \quad f_2\left(\mathbf{x}^*\right) = 1048.17$$

Assim, antes mesmo de começar a implementar o BVNS para resolver os problemas isoladamente, já sabemos as soluções ótimas para eles.

3) Modelagem Multiobjetivo: Juntando as modelagens dos problemas mono-objetivos acima, temos a modelagem multi-objetivo do problema.

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

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

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

sujeito a

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

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

onde

- $N = 500$: número de equipamentos
- $J = 3$: número de planos de manutenção
- c_j: custo de executar a manutenção j
- p_{ij}: probabilidade de falha do equipamento i executando a manutenção j, definido em 5 e 6
- d_i: custo de falha do equipamento i

A partir dessa modelagem, temos o nosso problema multiobjetivo

$$\min \mathbf{f}\left(\mathbf{x}\right) = \left[f_1\left(\mathbf{x}\right),\, f_2\left(\mathbf{x}\right)\right] \tag{8}$$

Considerando o problema de (8), podemos aplicar duas abordagens escalares para obter a fronteira Pareto-ótima no espaço de objetivos, descritas a seguir.

B. Formulações para resolução do problema multiobjetivo

1) Formulação Soma Ponderada P_w: Seja $0 \leq \mathrm{w} \leq 1$ um peso qualquer gerado aleatóriamente de uma distribuição uniforme no intervalo $[0, 1]$. Usando a abordagem da soma ponderada, podemos reescrever (8) na forma de mono-objetivo de

$$\min f_{\mathrm{w}} = \min \mathrm{w} f_1 + (1 - \mathrm{w}) f_2 \tag{9}$$

onde (9) está sujeito às mesmas restrições do problema original. Como (9) é escalar, podemos minimizar f_{w} através de métodos já conhecidos como o Simplex e o BVNS.

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

$$\min f_1 \tag{10}$$

sujeito a

$$\begin{cases} f_2 \leq \epsilon_2 \\ \sum_{j=1}^{J} x_{ij} = 1 \quad , \quad \forall i = 1, 2, ..., N \end{cases} \tag{11}$$

em que (10) possui as mesmas restrições do problema original mais a restrição de $f_2 \leq \epsilon_2$.

Contudo, como o BVNS é usado para resolver problemas de otimização irrestritos, precisamos converter (10) em um problema irrestrito. Para isso, adicionamos o termo um termo de penalidade $p(x, u)$ da seguinte forma:

$$p(x, u) = u\, \max\left[0, g(x)\right]^2$$

onde $g(x)$ é a nossa restrição de desigualdade, dada por

$$g(x) \leq 0 \quad \Longrightarrow \quad f_2 - \epsilon_2 \leq 0 \quad \Longrightarrow \quad g(x) = f_2 - \epsilon_2$$

de modo que o nosso problema irrestrito se torna:

$$\min f_1 + u\, \max\left[0, f_2 - \epsilon_2\right]^2 \tag{12}$$

Note que as demais restrições já estão naturalmente incluídas no BVNS devido à maneira como nós fizemos a representação computacional das variáveis de decisão, de modo que só precisamos fazer a correção para a restrição do ϵ em (12).

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

$$f_1(\mathbf{x}) = \frac{f_1(\mathbf{x}) - \min f_1}{\max f_1 - \min f_1} \quad , \quad f_2(\mathbf{x}) = \frac{f_2(\mathbf{x}) - \min f_2}{\max f_2 - \min f_2} \tag{13}$$

de modo que a formulação da soma ponderada de (9) pode ser reescrita como

$$\min \left(\mathrm{w} \frac{f_1(\mathbf{x}) - \min f_1}{\max f_1 - \min f_1} + (1 - \mathrm{w}) \frac{f_2(\mathbf{x}) - \min f_2}{\max f_2 - \min f_2} \right)$$

que será usado como função objetivo no código do BVNS.

4) BVNS - Basic Variable Neighborhood Search: O Algoritmo 1 mostra a versão do BVNS implementada no trabalho.

Algoritmo 1 BVNS implementado no trabalho.

```
1:  procedure BVNS(x, k_max)
2:      while num_sol_avaliadas < max_sol_avaliadas do
3:          k ← 1
4:          while k < k_max do
5:              x' ← SHAKE(x, k)
6:              x'' ← FIRSTIMPROVEMENT(x, x', k)
7:              x, k ← NEIGHBORHOODCHANGE(x, 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 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.

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

11:     return y
12: end procedure
```

5) 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'**.

```
1:  procedure FIRSTIMPROVEMENT(x', k)
2:      N ← 100
3:      for all i in range(N) do
4:          x" ← SHAKE(x', k)
5:          if f(x") < f(x') then
6:              return x"
7:          end if
8:      end for

9:      return 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 **x'** melhor que **x'**.

```
1:  procedure BESTIMPROVEMENT(x', k)
2:      N ← 100
3:      x_melhor ← x'
4:      for all i in range(N) do
5:          x" ← SHAKE(x', k)
6:          if f(x") < f(x_melhor) then
7:              x_melhor ← x"
8:          end if
9:      end for

10:     return x_melhor
11: end procedure
```

6) 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.

```
1:  procedure SOLUCAOINICIAL()
2:      x ← Solução aleatória
3:      for all i in x do
4:          if variancia(p_ij d_i) ≥ limiar then
5:              x[i] ← Manutenção mais cara
6:          else
7:              x[i] ← Manutenção mais barata
8:          end if
9:      end for

10:     return x
11: end procedure
```

III. RESULTADOS

A. Problemas mono objetivos

A partir da execução do algoritmo BVNS para a resolução dos problemas mono-objetivo foi obtido os seguintes valores:

$$\min f_1 = 0 \tag{14}$$
$$\max f_1 = 1000 \tag{15}$$
$$\min f_2 = 1048.17 \tag{16}$$
$$\max f_2 = 1745.49 \tag{17}$$

Esses resultados são utilizados para a realizar a normalização dos valores ao se utilizar a formulação de Soma Ponderada e os mínimos estão ilustrados nas Figuras 1 e 2.

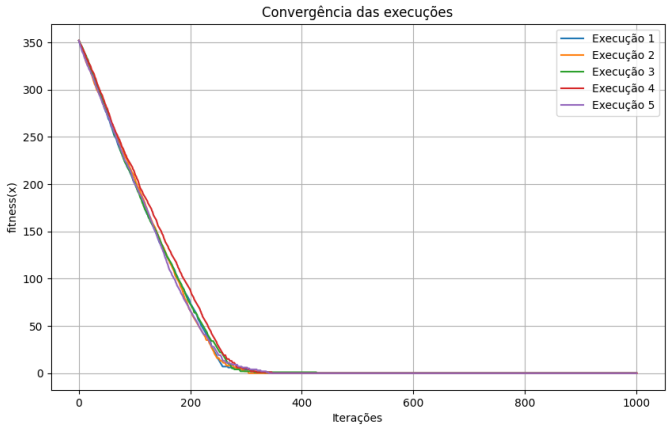

Fig. 1. Convergência do BVNS implementado para o Problema 1 Isolado.

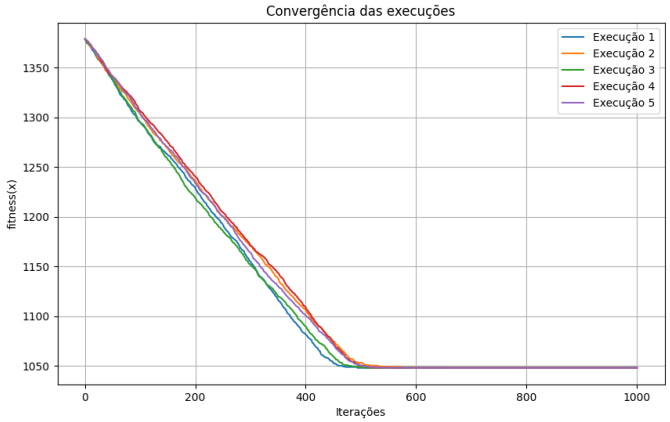

Fig. 2. Convergência do BVNS implementado para o Problema 2 Isolado.

B. Indicador de Hipervolume (HVI)

Para a realização do cálculo a métrica de HVI foi permitido o uso de mais de 20 soluções pareto na fronteira, de maneira que o valor para a métrica atingisse os valores indicados para comparação. Dessa forma, foram utilizadas 200 soluções e a visualização da fronteira pode ser vista na Figura 3.

O valor de HVI obtido foi:

$$\text{HVI: } 0.602974 \tag{18}$$

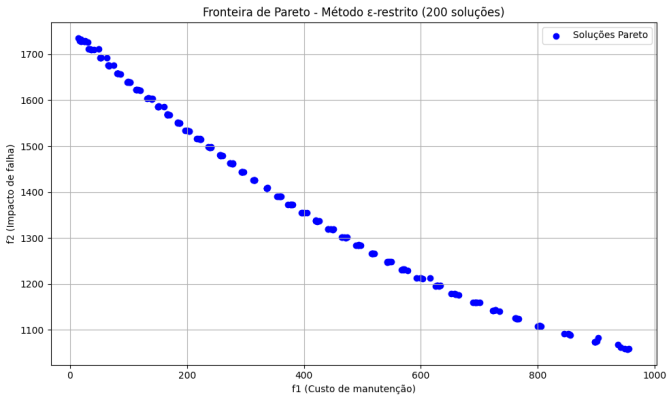

Fig. 3. Fronteira Pareto para 200 soluções com método ϵ-restrito usada para cálculo do HVI

C. Tomada de Decisão Multicritério

Para a tomada de decisão multicritério, vamos considerar 20 dentre as 200 soluções exibidas na Figura 3. Vamos comparar as soluções usando 4 atributos de interesse, definidos abaixo. A justificativa dos atributos 3 e 4 será dada a seguir.

1) Valor de f_1;
2) Valor de f_2;
3) Razão entre o número de equipamentos no plano 1 pelo número no plano 3;
4) Soma do número de equipamentos do Cluster 4 no plano 1 e o número de equipamentos de equipamentos do Cluster 1 no plano 1.

Para entender o atributo 3, vamos analisar algumas soluções da fronteira em termos dos equipamentos, como mostra a Figura 4. Quando o ϵ é pequeno, estamos priorizando a função f_2 e assim o algoritmo escolhe a solução que coloca quase todos no plano mais caro (M3), a fim de minimizar o custo esperado de falha. Quando ϵ é pequeno, priorizamos f_1 e o algoritmo coloca a manutenção mais barata (M1) para todos, minimizando o custo total. Quando ϵ está no meio termo, estamos no meio da fronteira Pareto e ele escolhe uma solução "meio-termo".

É importante notar que o plano M2 na Figura 4 quase nunca é utilizado. Assim, podemos analisar apenas as distribuições entre M1 e M3 para ter uma noção de como os equipamentos estão sendo distribuídos entre os planos. Além disso, podemos considerar que é indesejável colocar todos os equipamentos em apenas um plano: se a empresa tem 3 planos de manuteção, é esperado que os equipamentos sejam distribuídos ao longo desses três planos, evitando-se colocar todos os equipamentos em apenas um.

Assim, definimos o atributo 3: a razão entre o número de equipamentos no plano 1 pelo número no plano 3. Quanto mais próximo de 1 esse número estiver, melhor a classificação da solução, indicando que os equipamentos estão igualmente distribuídos ao longo de M1 e M3.

Agora vamos analisar como os valores de f_2 se comportam para diferentes clusters, a fim de entender melhor o que eles

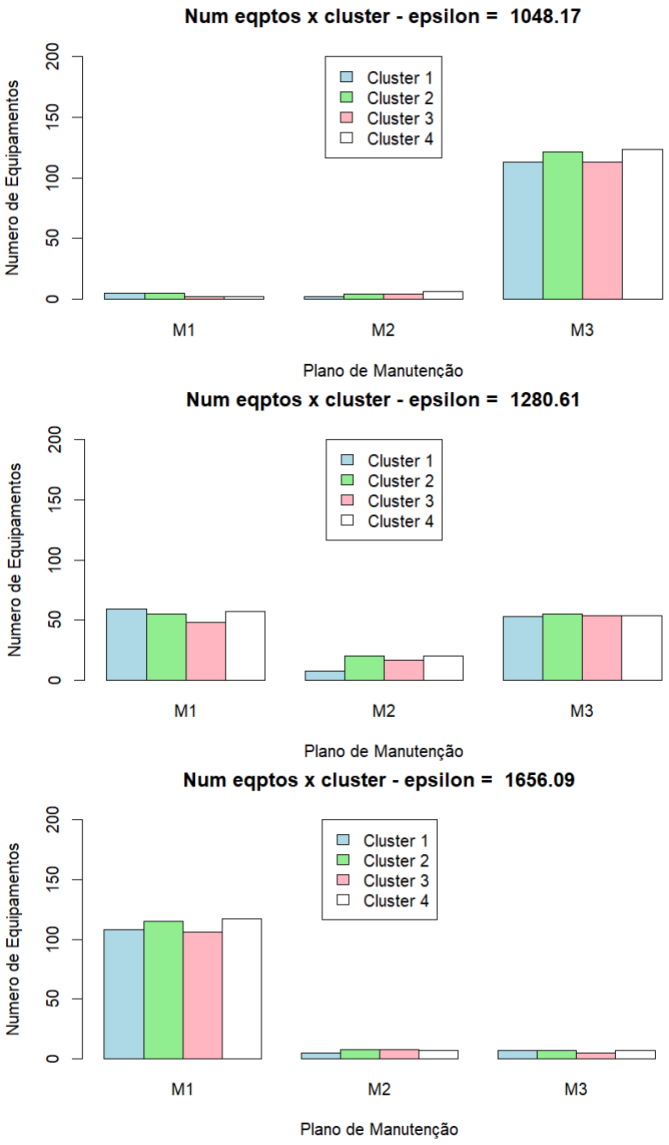

Fig. 4. Distribuição de equipamentos por plano de manutenção para diferentes valores de ϵ.

significam. A Figura 5 mostra o valor de f_2 para diferentes clusters. Vemos que f_2 - o custo esperado de falha - é menor para os equipamentos do Cluster 4 do que para os do Cluster 1. Esse padrão se repete para os três planos de manuteção. Logo, podemos interpretar isso como um indicativo de que os equipamentos do Cluster 1 são mais valiosos do que os equipamentos do Cluster 4, uma vez que o custo de falha deles é maior. Assim, quanto mais equipamentos do cluster 1 estiverem no plano mais caro (M3) e quanto mais equipamentos do cluster 4 estiverem no mais barato (M1), melhor.

Dessa forma, definimos o atributo 4: soma do número de equipamentos do Cluster 4 no plano 1 e o número de equipamentos de equipamentos do Cluster 1 no plano 1. Quanto maior esse número, melhor.

Para as 20 soluções escolhidas aleatóriamente, calculamos

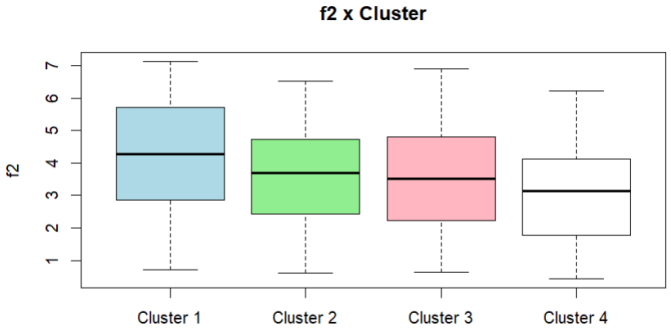

Fig. 5. Valor de f_2 para diferentes clusters.

os valores dos 4 atributos, obtendo os resultados da Figura 6.

Solução	f1	f2	A3	A4
1	765	1,124	0.27	121
2	36	1,710	78.33	120
3	728	1,143	0.31	115
4	14	1,736	100.00	125
5	450	1,320	1.26	110
6	420	1,339	1.45	118
7	517	1,266	0.93	110
8	763	1,126	0.26	117
9	956	1,059	0.03	115
10	397	1,356	1.61	108
11	80	1,658	17.15	124
12	569	1,232	0.73	112

Fig. 6. Valores calculados para os 4 atributos para as 20 soluções consideradas.

D. *Solução via AHP*

Usando o AHP, temos que montar as tabelas de comparação entre as 20 soluções para cada atributo, e depois uma tabela de comparação entre os atributos em si. Como os atributos são todos numéricos, elas podem ser feitas programaticamente através de condicionais. Por exemplo, para o atributo 3, se uma solução possui A3 igual a 1.2 e outra possui A3 = 20, colocamos o valor 9 na tabela indicando que a primeira é bem melhor do que a segunda.

O vetor de prioridades de cada tabela é calculado através do autovetor direito principal, e o índice de inconsistência (IC) é calculado para cada um das tabelas. A tabela de prioridades dos atributos tem dimensão 4 x 4 e está exibida na Figura 7.

Na Figura 7, colocamos alta prioridade para o atributo f_1. Nesse caso, a solução encontrada pelo AHP está indicada na Figura 8. Ele selecionou a solução que coloca todos no mais barato, de fato priorizando f_1, conforme esperado.

Modificando as prioridades na tabela de atributos, a solução selecionada muda para atender às novas relações entre os atributos. Por exemplo, colocando alta prioridade para o atributo A3 em relação aos demais, temos a solução indicada na Figura 9.

Att	f1	f2	A3	A4
f1	1.00	0.20	5	3.00
f2	5.00	1.00	5	3.00
A3	0.20	0.20	1	0.33
A4	0.33	0.33	3	1.00

Fig. 7. Tabela de atributos do AHP. IC = 0.12.

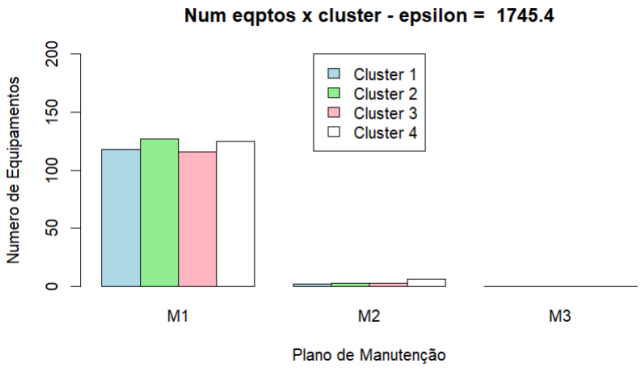

Fig. 8. Tabela de atributos do AHP. IC = 0.12.

Ele seleciona a solução em que a razão entre M1 e M3 é próxima de 1, conforme definimos o atributo.

E. Solução via PROMETHEE

IV. Conclusão

The conclusion goes here.

Referências

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

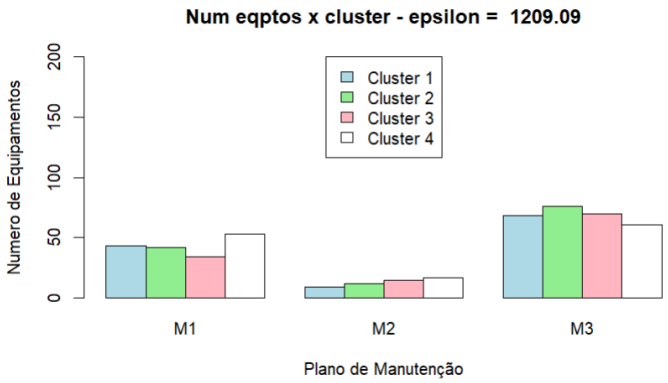

Fig. 9. Tabela de atributos do AHP. IC = 0.12.