UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA

Formulação Matemática Carrossel Seguro

João Francisco Hirtenkauf Munhoz
Leonardo Gonzatti Heisler

Introdução

Objetivo: Implementar uma metaheurística para resolver o problema do Carrossel Seguro.

Metaheurística escolhida: Simulated Annealing.

Descrição do problema:

- Carrossel com n assentos e $n=2k$ crianças;
- Atribuição um-para-um de assentos para as crianças, considerando os pesos;
- Objetivo do problema é deixar o Carrossel o mais balanceado possível, de maneira a minimizar o somatório dos pesos de suas metades.

Parâmetros e Variáveis de Decisão

- **Parâmetros:**
  - $n \in \mathbb{Z}+$: Número de crianças pertencentes ao carrossel;
  - $w[j] \in R+$: Peso associado a criança j:
    - $j \in [n]$

- **Variáveis de Decisão:**
  - $x[i][j] = \{$ 1, se a criança j está na posição i; 0, caso contrário$\}$
    - $i, j \in [n]$

Função Objetivo e Restrições

$$\textbf{minimiza} \quad W$$

$$\textbf{sujeito a} \quad W \geq W_i \qquad \forall i \in [n], \qquad (1)$$

Seleciona o maior W[i];

$$W_i = \sum_{k=0}^{\frac{n}{2}-1} p_{i+k} \qquad \forall i \in [n], \qquad (2)$$

Faz W[i] ser igual ao somatório dos pesos associados a ele;

$$p_i = \sum_{j=0}^{n} w_j x_{i,j} \qquad \forall i \in [n], \forall j \in [n], \qquad (3)$$

Determina o peso associado a posição da criança no carrossel;

$$\sum_{i=0}^{n} x_{i,j} = 1 \qquad \forall j \in [n], \qquad (4)$$

Garante que uma criança só está sentada em um lugar;

$$\sum_{j=0}^{n} x_{i,j} = 1 \qquad \forall i \in [n], \qquad (5)$$

Garante que um lugar só está ocupado por uma criança.

$$x_{i,j} \in \{0, 1\} \qquad \forall i \in [n], \forall j \in [n]. \qquad (6)$$

Algoritmo - Simulated Annealing

- O Simulated Annealing é uma metaheurística baseada no resfriamento de metais.
- Aceitação de novas soluções depende da variação ΔW e da temperatura.
- A temperatura diminui ao longo das iterações.
- Entradas:
    a. weights
    b. initial_temp
    c. cooling_rate
    d. max_iterations
- Processo:
    a. Gera uma solução inicial.
    b. Itera sobre possíveis trocas de posições (perturbações).
    c. Calcula a nova solução e decide se a solução é aceita com base na variação da função objetivo.
    d. Repete até que a temperatura atinja um limite mínimo.

Algoritmo - Pseudocódigo

Algorithm 1 Simulated Annealing Algorithm

```
0: procedure SimulatedAnnealing(weights, initial_temp, cooling_rate, max_iterations, input_file)
0:     n ← size of weights
0:     k ← n/2
0:     initial_solution ← [0, 1, 2, ..., n − 1]
0:     best_solution ← initial_solution
0:     best_W ← compute_W(initial_solution, weights, k)
0:     temp ← initial_temp
0:     current_solution ← initial_solution
0:     current_W ← best_W
0:     while temp > 1e − 10 do
0:         for i ← 1 to max_iterations do
0:             new_solution ← current_solution
0:             pos1, pos2 ← RandomPositions(n)
0:             Swap(new_solution, pos1, pos2)
0:             new_W ← compute_W(new_solution, weights, k)
0:             delta ← new_W − current_W
0:             if delta ≤ 0 or exp(−delta/temp) > RandomValue then
0:                 current_solution ← new_solution
0:                 current_W ← new_W
0:                 if current_W < best_W then
0:                     best_solution ← current_solution
0:                     best_W ← current_W
0:                     print input_file, best_W
0:                 end if
0:             end if
0:         end for
0:         temp ← temp ∗ cooling_rate
0:     end while
0:     return best_solution
0: end procedure=0
```

Algorithm 2 Compute Objective Function W

```
0: procedure ComputeW(assignment, weights, k)
0:     n ← size of weights
0:     W ← [0, 0, ..., 0] {Initialize W with zeros}
0:     for i ← 0 to n − 1 do
0:         for j ← 0 to k − 1 do
0:             W[i] ← W[i] + weights[assignment[(i + j) mod n]]
0:         end for
0:     end for
0:     return max(W) {Return the maximum value of W}
0: end procedure=0
```

Implementação

O trabalho foi implementado e testado em uma máquina utilizando as seguintes configurações:

- Sistema Operacional: Ubuntu 24.04.1 LTS x86 64;
- CPU: 12th Gen Intel i5-1235U (12);
- GPU: Intel Alder Lake-UP3 GT2 [Iris];
- RAM: 16GB;
- Linguagens: C++ para Simm Ann e Julia para HiGHS;
- Compilador: gcc;

Resultados - HiGHS

Instâncias	BKV	Valor Obtido	Desvio para Referência(%)
ocs01	49995	50237	0,48
ocs02	51150	51404	0,50
ocs03	193017	193543	0,27
ocs04	198406	198937	0,27
ocs05	433384	434349	0,22
ocs06	443476	0	100,00
ocs07	779642	0	100,00
ocs08	784200	0	100,00
ocs09	1243399	0	100,00
ocs10	1232361	0	100,00

Resultados - Heurística

Parâmetros:

- $initial\_temp = 100$;
- $cooling\_rate = 0.80$;
- $max\_iterations = 100$;

Instâncias	BVK	Valor Obtido	Desvio para Referência (%)
ocs1	49.995	50.614	1,24
ocs2	51.150	51.742	1,16
ocs3	193.017	194.643	0,84
ocs4	198.406	200.099	0,85
ocs5	433.384	436.788	0,79
ocs6	443.476	446.319	0,64
ocs7	779.642	784.020	0,56
ocs8	784.200	789.000	0,61
ocs9	1.243.399	1.249.893	0,52
ocs10	1.232.361	1.238.421	0,49

Resultados - Heurística

Parâmetros:

- $initial\_temp = 1000$;
- $cooling\_rate = 0.80$;
- $max\_iterations = 100$;

Instâncias	BVK	Valor Obtido	Desvio para Referência (%)
ocs1	49.995	50.578	1,17
ocs2	51.150	51.812	1,29
ocs3	193.017	194.656	0,85
ocs4	198.406	199.989	0,80
ocs5	433.384	436.298	0,67
ocs6	443.476	446.362	0,65
ocs7	779.642	784.043	0,56
ocs8	784.200	788.549	0,55
ocs9	1.243.399	1.249.148	0,46
ocs10	1.232.361	1.238.767	0,52

Resultados - Heurística

Parâmetros:

- $initial\_temp = 100$;
- $cooling\_rate = 0.90$;
- $max\_iterations = 100$;

Instâncias	BVK	Valor Obtido	Desvio para Referência (%)
ocs1	49.995	50.514	1,04
ocs2	51.150	51.664	1,00
ocs3	193.017	194.405	0,72
ocs4	198.406	199.849	0,73
ocs5	433.384	435.990	0,60
ocs6	443.476	445.892	0,54
ocs7	779.642	783.459	0,49
ocs8	784.200	787.966	0,48
ocs9	1.243.399	1.248.831	0,44
ocs10	1.232.361	1.237.582	0,42

Resultados - Heurística

Parâmetros:

- $initial\_temp = 100$;
- $cooling\_rate = 0.80$;
- $max\_iterations = 1000$;

Instâncias	BVK	Valor Obtido	Desvio para Referência (%)
ocs1	49.995	50.414	0,84
ocs2	51.150	51.536	0,75
ocs3	193.017	194.138	0,58
ocs4	198.406	199.488	0,55
ocs5	433.384	435.328	0,45
ocs6	443.476	445.493	0,45
ocs7	779.642	782.654	0,39
ocs8	784.200	787.147	0,38
ocs9	1.243.399	1.247.557	0,33
ocs10	1.232.361	1.236.374	0,33

Resultados - Heurística

Parâmetros:

- $initial\_temp = 1000$;
- $cooling\_rate = 0.90$;
- $max\_iterations = 1000$;

Instâncias	BVK	Valor Obtido	Desvio para Referência (%)
ocs1	49.995	50.351	0,71
ocs2	51.150	51.506	0,70
ocs3	193.017	193.940	0,48
ocs4	198.406	199.327	0,46
ocs5	433.384	435.176	0,41
ocs6	443.476	445.049	0,35
ocs7	779.642	782.271	0,34
ocs8	784.200	786.817	0,33
ocs9	1.243.399	1.247.040	0,29
ocs10	1.232.361	1.235.836	0,28

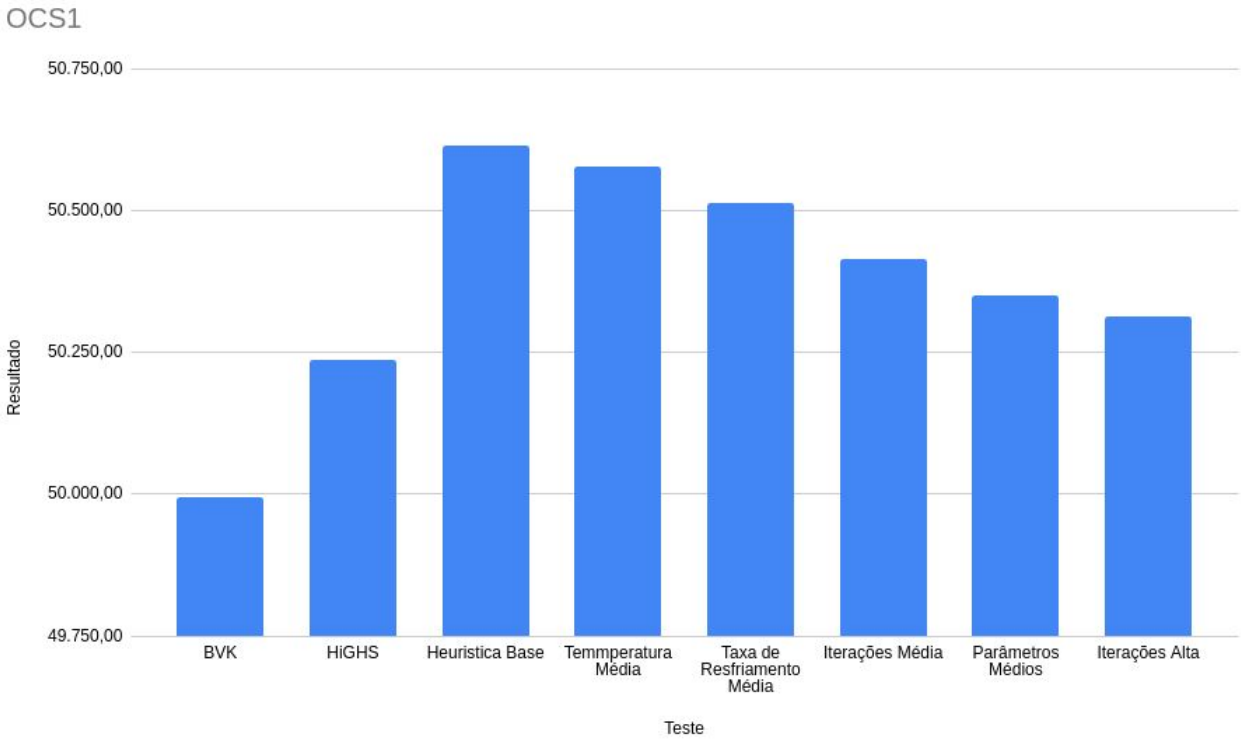
Resultados - Heurística

Parâmetros:

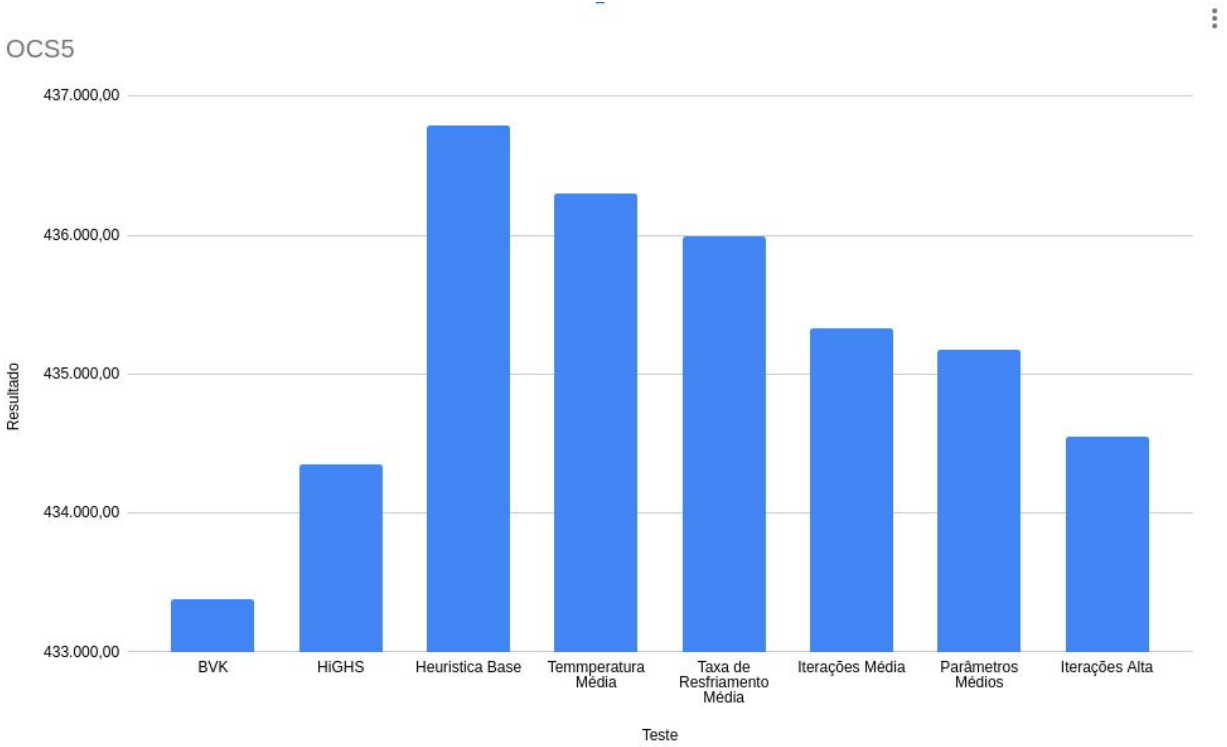
- $initial\_temp = 1000$;
- $cooling\_rate = 0.90$;
- $max\_iterations = 10000$;

Instâncias	BVK	Valor Obtido	Desvio para Referência (%)
ocs1	49.995	50.314	0,64
ocs2	51.150	51.491	0,67
ocs3	193.017	193.672	0,34
ocs4	198.406	199.147	0,37
ocs5	433.384	434.553	0,27
ocs6	443.476	444.584	0,25
ocs7	779.642	781.228	0,20
ocs8	784.200	785.971	0,23
ocs9	1.243.399	1.245.755	0,19
ocs10	1.232.361	1.234.642	0,19

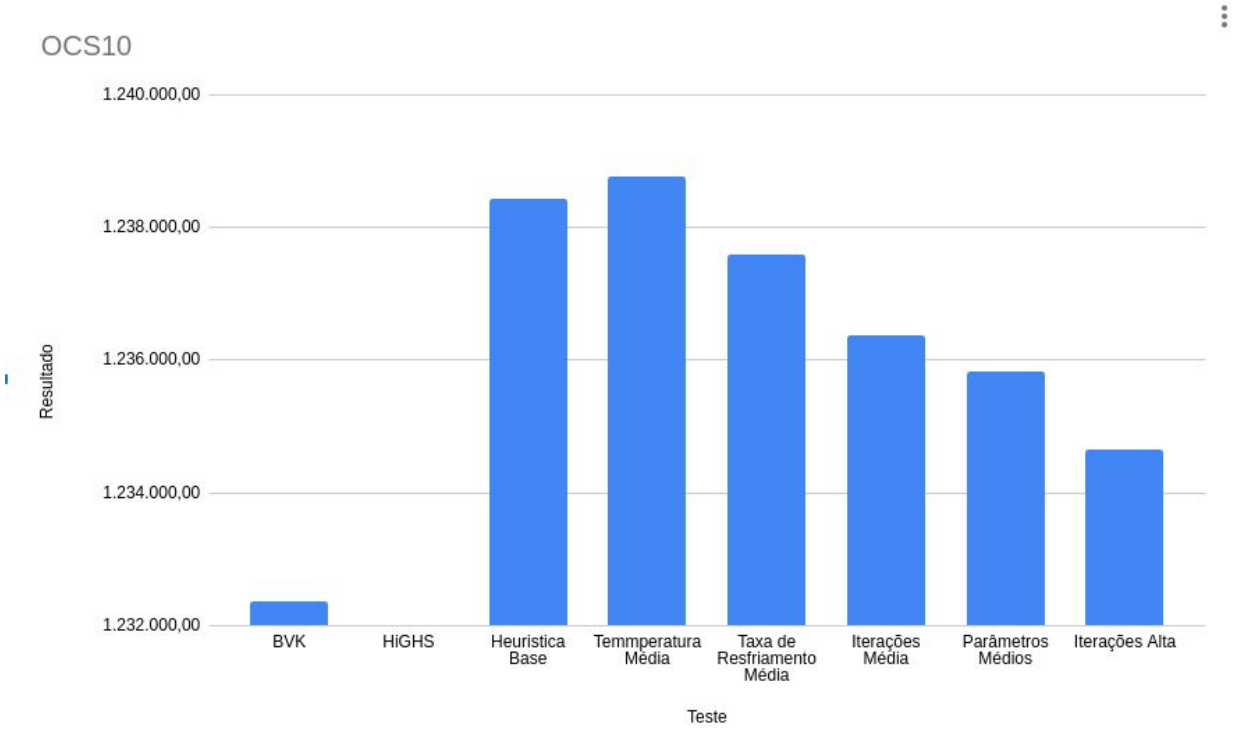
Resultados - Comparação



Resultados - Comparação



Resultados - Comparação



Conclusões

Instâncias:

- O tempo de resolução aumenta significativamente com o tamanho das instâncias.
- Para instâncias grandes, o solver se torna inviável.
- A abordagem heurística é mais eficaz em instâncias grandes, com desvios entre 0,49% e 1,24% em segundos, mas com menor precisão.

Parâmetros:

- O aumento dos parâmetros melhora a qualidade das soluções, mas também aumenta o tempo de execução.
- A heurística com parâmetros altos não superou o solver em termos de precisão, apesar de exigir várias horas.
- Testes com parâmetros mais baixos resultaram em soluções semelhantes, com tempo de execução reduzido (alguns minutos).
- O aumento da taxa de resfriamento ou do máximo de iterações melhora a solução, com maior impacto do máximo de iterações.
- O parâmetro mais interessante a ser priorizado é o máximo de iterações.

Observações

- As soluções obtidas pelo solver não atingiram o BKV (Best Known Value)
- O tempo de execução da heurística não foi registrado, o que dificultou a comparação entre os testes. Esse dado deveria ter sido coletado.
- A introdução de um parâmetro de tempo de execução máximo para a heurística poderia ter otimizado o processo, evitando execuções excessivamente longas.
- A temperatura mínima poderia ser melhor controlada, tornando-a configurável como um parâmetro para ajustar a exploração do algoritmo.