Formulação Matemática Carrossel Seguro

João Francisco Hirtenkauf Munhoz Leonardo Heisler

November 2024

1 Introdução:

O objetivo deste trabalho foi implementar uma metaheurística para solucionar o problema do Carrossel Seguro e compará-lo com as ferramentas de otimização disponíveis atualmente. Para esse fim, foi selecionada a heurística baseada na estratégia de Simulated Annealing. Segue a definição do problema abordado: Temos um carrossel com n assentos equidistantes (i.e. nos ângulos i2/n para i [0, n)) e n = 2k crianças com pesos w1, w2, . . . , wn 0. A partir disso, buscamos uma atribuição um-para-um a : [n] → [n] de assentos a crianças. Para n ¿ n seja a(n) = a(n n). Para cada i [n] define Wi como peso total das crianças nas posições i, i + 1, . . . , i + k 1, e seja W = maxi [n] Wi. Queremos minimizar W.

2 Formulação Matemática:

2.1 Parâmetros

- $n \in \mathbb{Z}_+$: Número de crianças.
- $w_j \in \mathbb{R}_+$: Peso de cada criança j.
  - $j \in [n]$.

2.2 Variáveis de Decisão:

- $x_{i,j} = \begin{cases} 1, & \text{se a criança } j \text{ está na posição } i, \\ 0, & \text{caso contrário.} \end{cases}$
  - $i \in [n]$,
  - $j \in [n]$.

2.3 Formulação:

$$
\begin{aligned}
\textbf{minimiza} \quad & W \\
\textbf{sujeito a} \quad & W \geq W_i && \forall i \in [n], && (1) \\
& W_i = \sum_{k=0}^{\frac{n}{2}-1} p_{i+k} && \forall i \in [n], && (2) \\
& p_i = \sum_{j=0}^{n} w_j x_{i,j} && \forall i \in [n], \forall j \in [n], && (3) \\
& \sum_{i=0}^{n} x_{i,j} = 1 && \forall j \in [n], && (4) \\
& \sum_{j=0}^{n} x_{i,j} = 1 && \forall i \in [n], && (5) \\
& x_{i,j} \in \{0,1\} && \forall i \in [n], \forall j \in [n]. && (6)
\end{aligned}
$$

2.4 Descrição das restrições:

(1) Garanante que W sejá igual ao maior W_i encontrado.

(2) Determina W_i como a soma dos pesos das crianças sentadas na metade do carrossel que começa na posição i.

(3) Determina p_i como o peso da criança j que está sentada na posição i.

(4) Garante que cada criança j está sentada em apenas uma posição i.

(5) Garante que cada posição i possui apenas uma criança j sentada.

(6) As variáveis que determinam a alocação das crianças são valores binários (0 ou 1).

3 Algoritmo

O *Simulated Annealing* é um algoritmo de otimização estocástica inspirado no resfriamento de metais, onde uma solução inicial é iterativamente perturbada para gerar soluções vizinhas. A aceitação de uma solução nova depende da variação no valor da função objetivo e da temperatura, com a probabilidade de aceitação calculada com base nessa variação. A temperatura diminui ao longo do tempo conforme um esquema de resfriamento, e o processo é repetido até que o algoritmo converja para uma solução ótima ou próxima disso.

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
```

3.1 Representação da Solução

A solução será representada por um vetor de inteiros com n linhas, onde n é o número de crianças. Cada linha do vetor representa um assento e o conteúdo indica qual criança está sentada no assento.

3.2 Valor da Função Objetivo

A função objetivo W é calculada avaliando uma solução representada pelo vetor "assignment", com base nas somas ponderadas dos elementos do vetor "weights". Para cada posição i no vetor W, a função percorre k elementos consecutivos da solução (onde k é metade do número de elementos), aplicando um deslocamento circular usando o operador módulo para acessar os índices no vetor "assignment". O valor de W[i] é incrementado com o peso associado ao índice "assignment[i + j mod n]", onde j varia de 0 até k-1. Após a iteração para todas as posições, a função retorna o maior valor encontrado em W, que é o valor da função objetivo, usado para avaliar a qualidade da solução no processo de otimização.

Algorithm 2 Compute Objective Function W

procedure COMPUTEW($assignment, weights, k$)
  $n \leftarrow$ size of weights
  $W \leftarrow [0, 0, ..., 0]$ {Initialize W with zeros}
  for $i \leftarrow 0$ to $n - 1$ **do**
    for $j \leftarrow 0$ to $k - 1$ **do**
      $W[i] \leftarrow W[i] + weights[assignment[(i + j) \mod n]]$
    end for
  end for
  return max(W) {Return the maximum value of W}
end procedure=0

3.3 Parâmetros

- *initial_temp* : a temperatura inicial do algoritmo de simulated annealing, que controla a probabilidade de aceitar soluções subótimas no início do processo.
- *cooling_rate* : a taxa de resfriamento que determina a velocidade com que a temperatura é reduzida ao longo das iterações.
- *max_iterations* : o número máximo de iterações que o algoritmo executa a cada ciclo de resfriamento para explorar o espaço de soluções.

3.4 Critério de Terminação

O critério de terminação é a temperatura do algorítmo. Quando a temperatura cai abaixo da temperatura mínima determinada o algoritmo é encerrado.

4 Implementação

4.1 Plataforma de 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];
- Memória RAM: 16GB;
- Linguagens: C++ para Simm Ann e Julia para HiGHS;
- Compilador: gcc;

4.2 Estruturas de Dados Utilizadas

No algoritmo de Simulated Annealing e no cálculo da função objetivo W, as principais estruturas de dados são vetores e variáveis escalares. O vetor assignment é um array de tamanho n, que armazena os índices de uma solução e é manipulado para explorar diferentes permutações. O vetor weights é também um array de tamanho n, contendo os pesos associados a cada elemento. A função objetivo W é calculada usando um vetor W[i] de tamanho n, onde cada posição armazena a soma ponderada das soluções. O cálculo é feito utilizando o operador módulo para garantir a continuidade circular do índice. Além disso, variáveis escalares como temp, best_W e current_W são usadas para controlar a temperatura e avaliar a qualidade da solução. Essas estruturas permitem o gerenciamento eficiente dos dados durante a otimização.

5 Teste de Parâmetro

5.1 Teste Base

Foi feito um teste base com parâmetros baixos para a partir dele aumentar cada parâmtro e verificar a diferença nos resultados.
Os valores base dos parâmetros são:

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

5.2 Testes Individuais de Cada Parâmetro

A partir dos valores base, cada parâmetro foi alterado individualmente para analisar o impacto nas soluções. Os novos valores de cada parâmetro são:

- $nova\_initial\_temp = 1000$;

- *nova_cooling_rate* = 0.90;
- *nova_max_iterations* = 1000;

5.3 Outros Testes de Parâmetro

Também foi executado um teste utilizando os novos valores dos parâmetros todos ao mesmo tempo e outro teste que a partir desses novos parâmetros aumenta mais ainda a variável *max_iterations*. Não foram testadas outras variáveis com valores mais altos porque o teste demora significantemente mais tempo, então escolhemos apenas a variável que acreditamos que teria mais impacto.

6 Resultados de Testes das Instâncias

Table 1: Valores referentes a cada instância

Instâncias	**BKV**
ocs01	49995
ocs02	51150
ocs03	193017
ocs04	198406
ocs05	433384
ocs06	443476
ocs07	779642
ocs08	784200
ocs09	1243399
ocs10	1232361

6.1 HiGHS

Table 2: Valores referentes a cada instância

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

6.2 Heuristica Base

Parâmetros:

- *initial_temp* = 100;
- *cooling_rate* = 0.80;
- *max_iterations* = 100;

Table 3: Valores referentes a cada instância

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

6.3 Heuristica - Temperatura Média

Parâmetros:

- *initial_temp* = 1000;
- *cooling_rate* = 0.80;
- *max_iterations* = 100;

Table 4: Valores referentes a cada instância

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

6.4 Heuristica - Taxa de Resfriamento Média

Parâmetros:

- *initial_temp* = 100;
- *cooling_rate* = 0.90;
- *max_iterations* = 100;

Table 5: Valores referentes a cada instância

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

6.5 Heuristica - Máximo de Iterações Média

Parâmetros:

- *initial_temp* = 100;
- *cooling_rate* = 0.80;
- *max_iterations* = 1000;

Table 6: Valores referentes a cada instância

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

6.6 Heuristica Média

Parâmetros:

- *initial_temp* = 1000;
- *cooling_rate* = 0.90;
- *max_iterations* = 1000;

Table 7: Valores referentes a cada instância

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

6.7 Heuristica - Max Iterations Alto

Parâmetros:

- *initial_temp* = 1000;
- *cooling_rate* = 0.90;
- *max_iterations* = 10000;

Table 8: Valores referentes a cada instância

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

7 Conclusões

7.1 Instâncias

Observamos que o tempo necessário para resolver os problemas aumenta consideravelmente à medida que instâncias de maior tamanho são introduzidas, tornando inviável o uso do solver para essas situações. Para as instâncias maiores, a abordagem heurística se mostrou mais eficaz, apesar de oferecer menor precisão. Mesmo assim, ela foi capaz de encontrar soluções com desvios entre 0,49 e 1,24 em apenas alguns segundos, na versão Base.

7.2 Parâmetros

O aumento dos parâmetros resulta em uma melhoria na qualidade das soluções, porém, também acarreta um aumento significativo no tempo de resolução. Mesmo no caso extremo testado (*Heurística - Máximo de Iterações Alto*), o método não conseguiu atingir um desvio menor do que o alcançado pelo solver, apesar de exigir várias horas para completar todos os testes. Nos demais testes, que utilizaram valores mais baixos para os parâmetros, o tempo de execução foi consideravelmente reduzido, não ultrapassando alguns minutos, e as soluções obtidas foram semelhantes às de configurações mais complexas.
Considerando os parâmetros separadamente, pode-se observar que manter a taxa de resfriamento e o máximo de iterações baixos e aumentar a temperatura inicial pode acabar piorando a solução final. Isso faz sentido, pois está introduzindo mais caos no início do algorítmo. Aumentar a taxa de resfriamento ou o máximo de iterações por ciclo melhora a solução, mas no nosso caso o máximo de iterações alcançou soluções melhores. O quanto a solução melhora depende de quanto foi aumentado, mas considerando que os dois testes tiveram um aumento semelhante no tempo de execução, concluimos que o parâmeto mais interessante a ser priorizado é o máximo de iterações.