Simulated Annealing:
Uma abordagem ao Traveling Salesman Problem (TSP)

Amyr Allan[1], **Wyllen Brito da Silva**[1]

[1]Centro de Ciências Tecnológicas – Universidade do Estado de Santa Catarina (UDESC)
R. Paulo Malschitzki, 200 - Zona Industrial Norte, Joinville - SC, 89219-710

amyr.allan@hotmail.com, wyllen2015@gmail.com

Resumo. *Este trabalho utiliza o algoritmo de Simulated Annealing para resolver o problema do Caixeiro Viajante (TSP), com o objetivo de minimizar a distância total entre as cidades. Foram exploradas diferentes configurações do algoritmo, como valores de SA_Max, rotinas de resfriamento e o parâmetro Swap_Factor, a fim de otimizar a solução. Os experimentos foram realizados com instâncias de 51 e 100 cidades, e os resultados mostraram que a rotina de resfriamento R_2, com SA_Max = 10 e Swap_Factor = 1, obteve as melhores soluções, com a menor distância total e maior consistência. A solução da minimização da distância total entre as cidades chegou a valores aproximados da solução ótima.*

Abstract. *This work applies the Simulated Annealing algorithm to solve the Traveling Salesman Problem (TSP), aiming to minimize the total distance between cities. Different configurations of the algorithm were explored, such as SA_Max values, cooling routines, and the Swap_Factor parameter, in order to optimize the solution. Experiments were carried out with instances of 51 and 100 cities, and the results showed that the cooling routine R_2, with SA_Max = 10 and Swap_Factor = 1, yielded the best solutions, with the lowest total distance and highest consistency. The minimization solution for the total distance between cities reached values close to the optimal solution.*

1. Introdução

1.1. Contextualização do Problema e Revisão da Literatura

O Simulated Annealing é um método para resolver problemas de otimização com e sem restrições. O método modela o processo físico de aquecimento de um material e, em seguida, a redução lenta da temperatura para reduzir defeitos. [Mathworks 2024]

A cada iteração, um novo ponto (vizinho) é gerado aleatoriamente. A distância do novo ponto ao ponto atual, ou a extensão da busca, é baseada em uma distribuição de probabilidade com escala proporcional à temperatura. O algoritmo aceita todos os novos pontos que reduzem a função objetivo, mas também, com certa probabilidade, pontos que elevam a função objetivo. Ao aceitar pontos que elevam, o algoritmo evita ficar preso em mínimos locais e é capaz de explorar globalmente mais soluções possíveis. Um cronograma é realizado para diminuir sistematicamente a temperatura à medida que o algoritmo avança. À medida que a temperatura diminui, o algoritmo reduz a extensão de sua busca para convergir a um mínimo. [Mathworks 2024]

Dessa forma, o algoritmo enquadra-se como técnica de inteligência artificial, sendo adequado para resolver problemas NP e NP-Hard, como o problema do Caixeiro Viajante - Traveling Salesman Problem (TSP).

1.2. Justificativa e Motivação

Neste trabalho, o algoritmo Simulated Annealing é empregado para **minimizar** a distância total entre cidades, a fim de obter um caminho hamiltoniano com custo mínimo, isto é, passar por todas as cidades sem repeti-las de forma a retornar ao ponto(cidade) inicial, minimizando o custo total do percurso (um problema de otimização).

1.3. Organização do Relatório

Este relatório está organizado da seguinte forma: na seção "Métodologia de Desenvolvimento", é apresentada a implementação do algoritmo de Simulated Annealing no contexto do TSP; na seção "Descrição de Experimentos e Resultados Obtidos", os experimentos são detalhados em ordem de execução, juntamente com seus respectivos resultados e análises; por fim, a seção "Conclusão" apresenta as considerações finais sobre o desempenho do algoritmo.

2. Métodologia de Desenvolvimento

2.1. Instâncias do TSP

Instâncias do TSP foram representadas no formato de pontos em um plano cartesiano, com as coordenadas (x,y) representando uma cidade do TSP. Foram utilizadas duas instâncias de complexidade crescente: uma com 51 cidades e outra com 100 cidades, abrangendo diferentes escalas do problema.

Neste trabalho, as instâncias representadas por pontos em um plano cartesiano são substituídas por uma **matriz de distâncias euclidianas**. Cada elemento dessa matriz, denotado como $d(i, j)$, representa o valor da distância euclidiana entre as cidades i e j. Assim, o **custo do percurso** de uma cidade para outra corresponde ao valor na posição (i, j) da matriz de distâncias, que indica a distância entre a cidade x e a cidade y.

A distância euclidiana entre duas cidades (x_1, y_1) e (x_2, y_2) em um plano cartesiano é calculada pela fórmula:

$$d(i, j) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Esse processo é realizado para evitar a necessidade de calcular repetidamente o custo de uma cidade para outra ao longo das iterações do algoritmo. Com a matriz de distâncias já pré-calculada, a obtenção do custo de qualquer percurso entre duas cidades pode ser feita de maneira instantânea, otimizando o tempo de execução do algoritmo.

2.2. Função Objetivo (Energia)

A energia de uma solução é definida como a distância total percorrida entre as cidades, para uma solução arbitrária gerada aleatoriamente. Esta solução é representada por um vetor permutado, cujo tamanho corresponde ao número de cidades no problema.

2.3. Valor Inicial e Geração de Vizinhos

O algoritmo de Simulated Annealing começa com uma permutação aleatória do vetor de cidades, onde cada elemento representa uma cidade, e a sequência das cidades é determinada de forma aleatória, sem repetição. O tamanho desse vetor corresponde ao número de cidades (para este trabalho, 51 ou 100 cidades). A geração de vizinhos é feita através de trocas (SWAPs) entre as posições das cidades no vetor, resultando em novas soluções possíveis para o problema.

O parâmetro SWAP_factor é uma constante definida pelo usuário e determina o número máximo de trocas que podem ser realizadas. A quantidade exata de trocas em uma iteração é determinada por um valor aleatório entre 1 e o valor de SWAP_factor. Isso significa que a quantidade de modificações realizadas na solução atual pode variar, proporcionando diversidade no processo de exploração do espaço de soluções.

```
def random_value(self):
    return numpy.random.permutation(self.num_cities)

def get_neighbour(self, current):
    num_swaps = random.randint(1, int(self.SWAP_factor))
    new = numpy.copy(current)

    for _ in range(num_swaps):
        i, j = numpy.random.choice(current.size, size=2, replace=False)
        new[i], new[j] = new[j], new[i]  # faz o swap

    return new
```

Figura 1. Código de geração de vizinhos e valores aletórios

2.4. Rotinas de Resfriamento

Três rotinas de resfriamento foram utilizadas: R_1, R_2 e R_3, todas dependentes do número de iterações i, da temperatura inicial T_0, da temperatura final T_f e do número máximo de avaliações N:

$$R_1 : T_i = T_0 \cdot \left(\frac{T_f}{T_0}\right)^{\frac{i}{N}}$$

$$R_2 : T_i = \frac{(T_0 - T_N)}{\cosh \frac{10i}{N}} + T_N$$

$$R_3 : T_i = \frac{1}{2}(T_0 - T_f) \cdot (1 - \tanh\left(\frac{10i}{N} - 5\right)) + T_f$$

As três rotinas foram escolhidas por exibirem perfis de resfriamento distintos. Tanto R_1 quanto R_2 combinam um resfriamento inicial rápido (associado a uma exploração global acelerada) com um resfriamento final suave (correspondente a uma busca local cautelosa). Em contraste, R_3 apresenta um resfriamento inicial mais gradual, indicando uma exploração global prolongada, mantendo igualmente uma fase final de busca local precisa.

2.5. Parâmetros

Durante a realização dos experimentos, alguns parâmetros do algoritmo foram mantidos invariantes e outros variáveis, como detalhado pela Tabela 1.

Para este trabalho, a temperatura inicial no algoritmo varia conforme o número de cidades no problema. Quando o número de cidades é maior, é necessário utilizar uma temperatura inicial mais alta para permitir que soluções piores sejam aceitas no início da busca, o que favorece a exploração do espaço de soluções e aumenta as chances de encontrar uma solução globalmente ótima. Isso ocorre porque, com um maior número de cidades, a distância total tende a ser maior, o que implica uma maior diversidade nas soluções. Para as instâncias com 100 cidades, foi atribuído um valor inicial de temperatura de 8000, enquanto para as instâncias com 51 cidades, a temperatura inicial foi definida como 100.

Tabela 1. Parâmetros utilizados nos experimentos

Parâmetro	**Valor**
Temperatura inicial (T_0)	{100,8000}
Temperatura final (T_f)	10^{-3}
Fator de swap ($SWAP\_FACTOR$)	{1,3,5}
Número de avaliações (N)	10^5
SaMax (k)	$\{1, 5, 10\}$
Rotinas de Resfriamento (R)	$\{R_1, R_2, R_3\}$

A escolha definitiva dos valores de parâmetros variáveis e suas justificativas são detalhadas na seção subsequente.

3. Descrição de Experimentos e Resultados Obtidos

Foram realizados experimentos comparativos para conduzir a escolha da rotina de resfriamento, do $SaMax$ e do valor de $Swap\_factor$, de forma que a distância total entre as cidades fossem minimizadas de forma adequada. Os resultados foram analisados sob as seguintes métricas:

- minimização média da função objetivo (distância total)
- variabilidade dos resultados finais sob trinta execuções
- variabilidade dos resultados finais sob dez execuções para escolha de $Swap\_factor$

3.1. Comparação de Rotinas de Resfriamento e Análise dos Resultados

Para a comparação das rotinas de resfriamento, fixou-se o parametro variável SaMax em SaMax = 1. Para 51 cidades fixou-se a temperatura inicial em 100, enquanto que para

100 cidades a temperatura inicial ficou em 8000. Os resultados de cada rotina podem ser visualizados na Figura 2 e na Tabela 2. Para a construção do gráfico Box Plot na Figura 2 e da Tabela 2 foram usadas 30 iterações do programa.

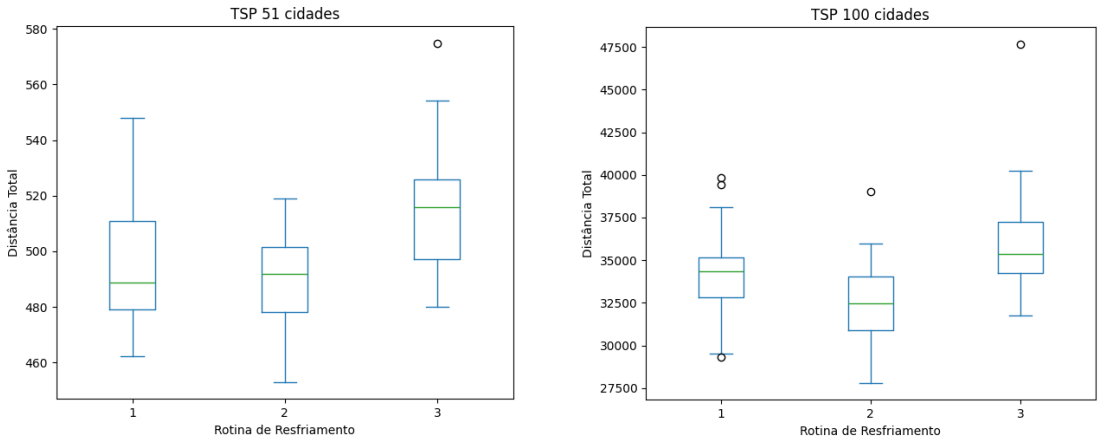

Figura 2. Box-plot para diferentes funções de resfriamento

Tabela 2. Resultados das rotinas de resfriamento para cada instância

Instância	Rotina de Resfriamento	Média	Desvio Padrão
51 Cidades	R1	497.02	21.79
	R2	489.18	16.79
	R3	514.99	21.09
100 Cidades	R1	34171.53	2540.12
	R2	32639.76	2406.51
	R3	36025.30	2971.49

A Rotina de Resfriamento 2 se destaca como a mais eficaz, pois apresenta os valores mais baixos de distância total entre as cidades, tanto para as instâncias de 51 cidades quanto para as de 100 cidades. Observa-se que a média de distância para a rotina 2 é consistentemente mais baixa em comparação com as outras rotinas. Além disso, o desvio padrão da rotina 2 também é relativamente baixo, indicando menor variação nos resultados, o que sugere uma maior estabilidade na solução.

A análise dos box plots apresentados na Figura 2 confirma essa observação. A Rotina de Resfriamento 2 apresenta uma mediana mais baixa, o que indica que ela alcançou uma solução mais próxima da minimização da distância total. Embora a Rotina 1 (R1) também mostre bons resultados, com médias e desvios padrão próximos da R2, a R2 consegue atingir valores mais baixos de distância total, como evidenciado tanto pela tabela quanto pelos box plots.

3.2. Comparação de Valores de SaMax e Análise dos Resultados

Já utilizando R_2 como rotina de resfriamento, comparou-se os valores de SaMax. Os resultados para cada valor podem ser visualizados na Figura 3 e na Tabela 3.

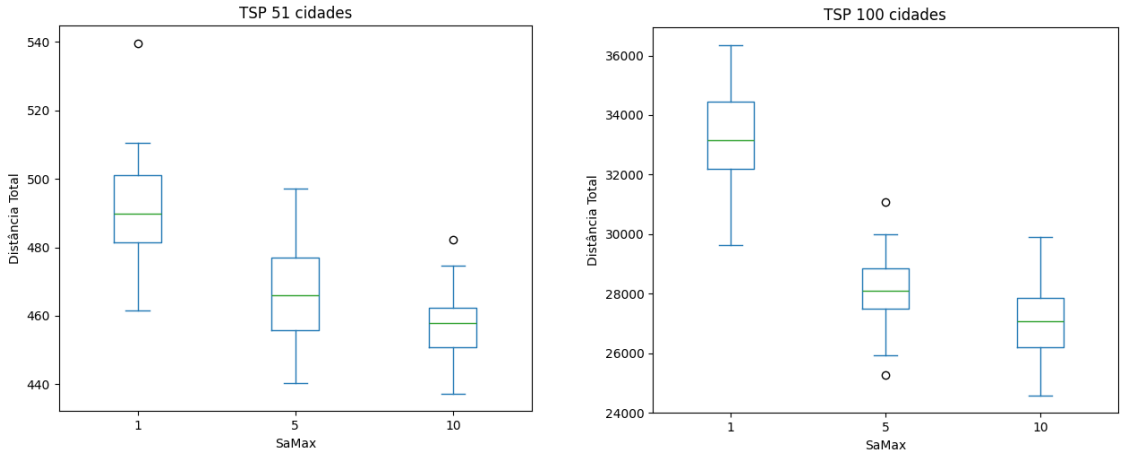

Figura 3. Box-plot para diferentes valores de SaMax

Tabela 3. Resultados das instâncias para cada SA_Max

Instância	SA_Max	Média	Desvio Padrão
51 Cidades	1	490.97	15.84
	5	466.44	14.18
	10	457.09	10.01
100 Cidades	1	33214.92	1656.85
	5	28170.96	1202.18
	10	27032.23	1166.21

No Box-plot apresentado na Figura 3, podemos observar que a rotina com SA_Max = 10, para ambas as instâncias (51 e 100 cidades), apresenta uma mediana mais baixa e uma menor dispersão, indicando uma solução mais estável e eficiente. Em contraste, as rotinas com SA_Max = 1 e SA_Max = 5 exibem distribuições mais dispersas e valores maiores de distância, o que mostra que essas rotinas são menos eficazes para minimizar a distância total entre as cidades.

Ao analisar a tabela de média e desvio padrão de SA_Max (Tabela 3), podemos concluir que SA_Max = 10 apresenta a menor média e o menor desvio padrão para ambas as instâncias (51 e 100 cidades), indicando maior consistência nas soluções encontradas. Isso sugere que o valor de SA_Max = 10 resulta em soluções de melhor qualidade e mais semelhantes entre as execuções.

3.3. Comparação de Valores de SWAP_Factor e Análise dos Resultados

Utilizando R_2 como rotina de resfriamento e SA_Max = 10, foram comparados os valores de SWAP_Factor. Os resultados para cada valor podem ser visualizados na Figura 4 e na Tabela 4. Para esta análise, foram consideradas apenas 10 iterações.

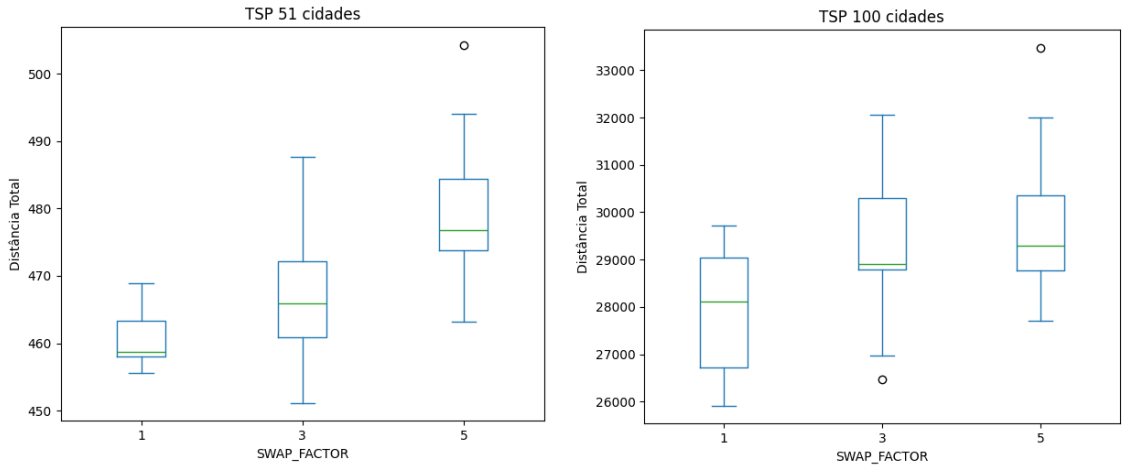

Figura 4. Box-plot para diferentes valores de SWAP_Factor

Tabela 4. Resultados para cada Swap_Factor nas instâncias de 51 e 100 cidades

Instância	Swap_Factor	Média	Desvio Padrão
51 Cidades	1	460.68	4.34
	3	467.98	10.85
	5	480.18	11.37
100 Cidades	1	27910.42	1314.01
	3	29218.85	1616.62
	5	29761.97	1722.20

Pode-se concluir que **Swap_Factor = 1** apresenta o melhor desempenho em termos de minimização da distância total entre as cidades. Analisando o *box-plot* na Figura 4, para a instância de 51 cidades, o Swap_Factor = 1 apresenta a menor mediana e uma dispersão mais estreita, indicando que essa configuração gera soluções mais consistentes e com menor distância total em média.

Da mesma forma, ao analisar a tabela 4, podemos concluir que os valores de **Swap_Factor = 1** apresentam as menores médias e desvio padrão, em comparação com as configurações de **Swap_Factor = 3** e **5**, sugerindo soluções de melhor qualidade e com menor variabilidade entre as execuções.

3.4. Teste de Convergência e Análise dos Resultados

Por fim, foi analisada a convergência do algoritmo para um mínimo global, testando a sua adequação. Por conta dos resultados expostos anteriormente, os parametros variáveis foram fixados em $SaMax = 10$, $R = R_2$ e $Swap\_Factor = 1$.

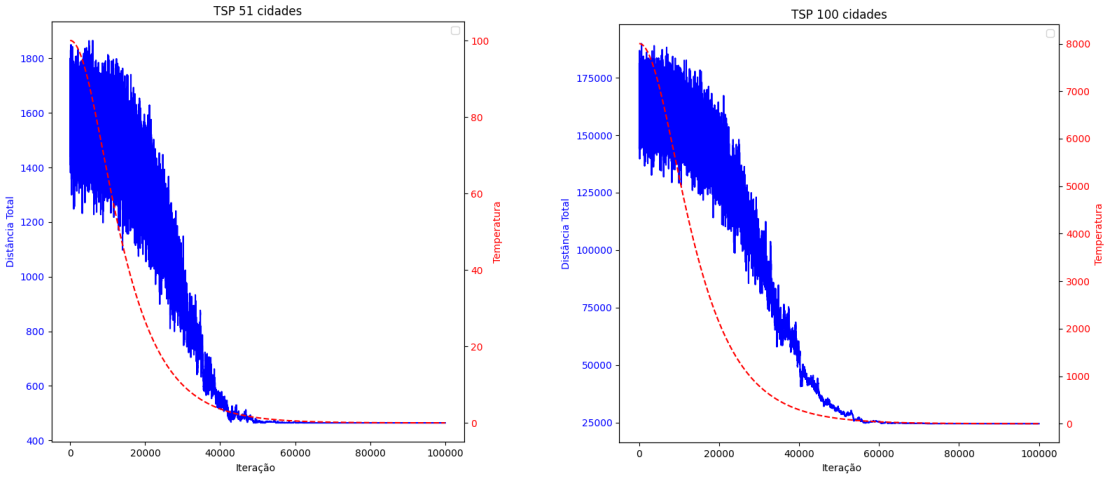

Figura 5. Gráfico de Convergência para 100.000 iterações

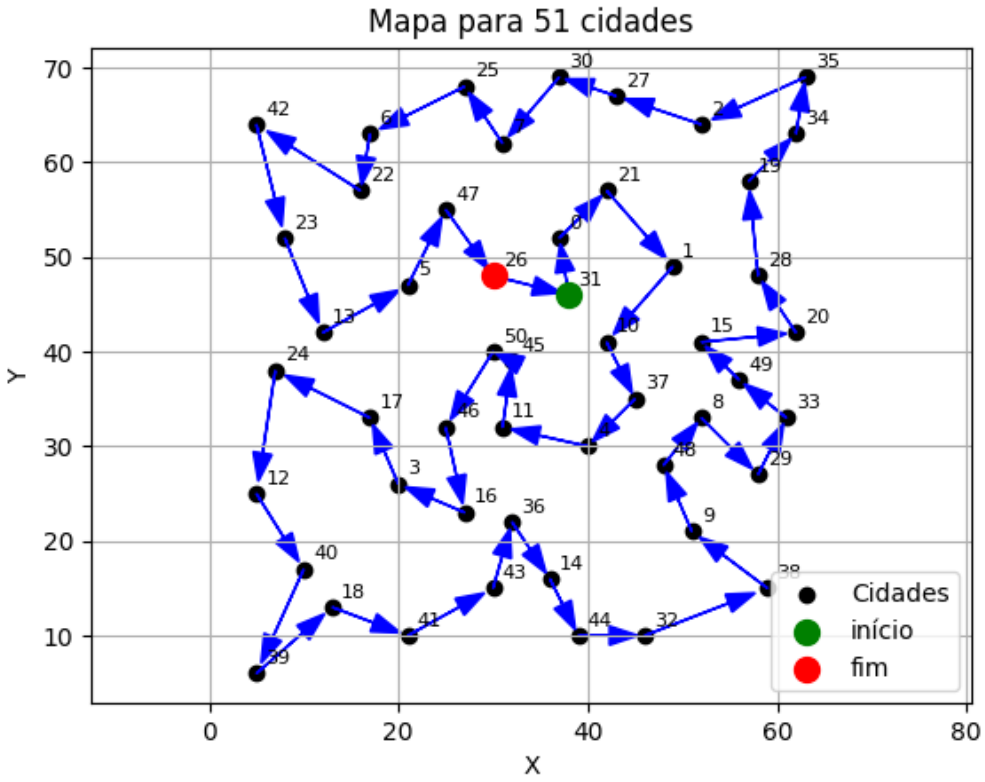

Figura 6. Caminho percorrido para 51 cidades

Distância Percorrida para 51 cidades, ao final da última iteração: 443.64

Ordem das cidades visitadas: [31 0 21 1 10 37 4 11 45 50 46 16 3 17 24 12 40 39 18 41 43 36 14 44 32 38 9 48 8 29 33 49 15 20 28 19 34 35 2 27 30 7 25 6 22 42 23 13 5 47 26]

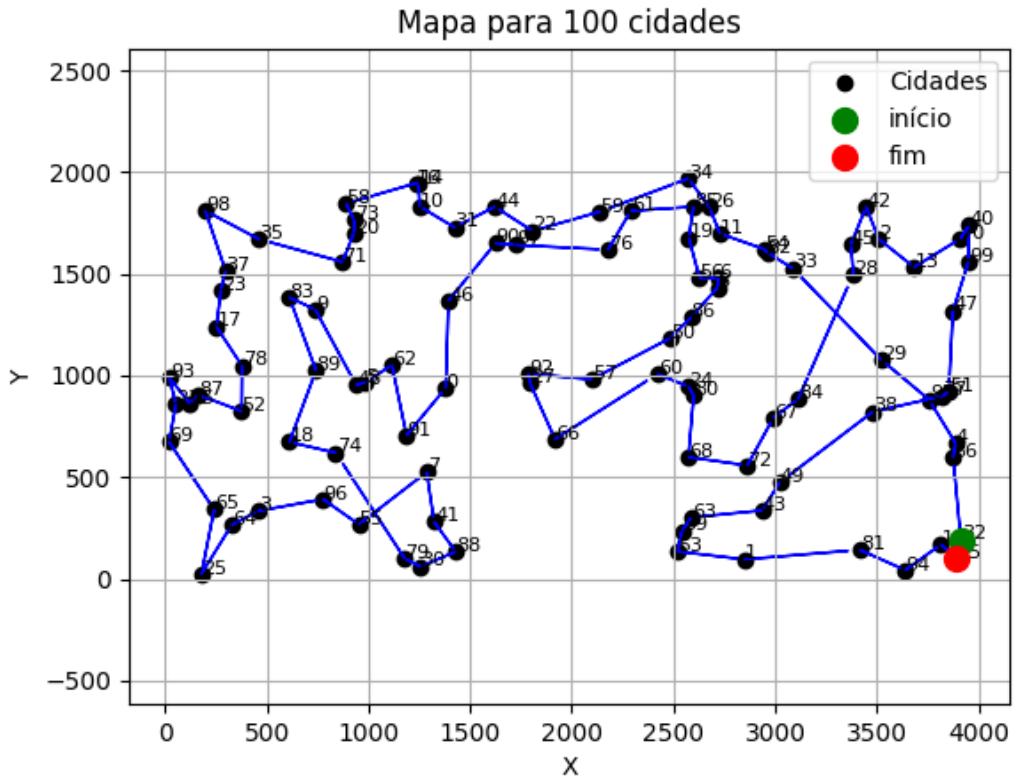

Figura 7. Caminho percorrido para 100 cidades

Distância percorrida para 100 cidades, ao final da última iteração: 24464.65

Ordem das cidades visitadas: [32 36 4 95 29 33 82 54 11 26 34 59 22 44 31 10 14 16 58 73 20 71 35 98 37 23 17 78 52 87 15 93 21 69 65 25 64 3 96 55 7 41 88 30 79 74 18 89 83 9 48 5 62 91 0 46 90 97 76 61 85 19 56 6 8 86 50 57 92 27 66 60 24 80 68 72 67 84 28 45 42 2 13 70 40 99 47 51 77 38 49 43 63 39 53 1 81 94 12 75]

Como é demonstrado no gráfico de convergência da Figura 5, a energia resultante diminui de forma constante ao longo da execução. No início do algoritmo, observa-se uma forte oscilação no valor da energia (representada pela distância total entre as cidades), o que indica uma fase exploratória intensa. Esse comportamento é esperado, pois, com a temperatura elevada, o algoritmo tende a aceitar soluções piores com maior probabilidade, o que é útil para evitar que o processo fique preso em mínimos locais.

À medida que o número de iterações aumenta e a temperatura diminui, a energia se reduz de forma mais gradual, com oscilações mais suaves, indicando que o algoritmo está convergindo para soluções de maior qualidade. No final do processo, o gráfico tende a se estabilizar, com pequenas variações, caracterizando a fase de ajuste fino. Durante essa fase, o algoritmo praticamente só aceita soluções melhores, ou, com baixa probabilidade, soluções ligeiramente piores.

4. Conclusão

Com base nos resultados, observa-se que o algoritmo foi capaz de trabalhar em melhores condições quando configurado para usar a rotina de Resfriamento R_2, valor de SaMax = 10 e $Swap\_Factor = 1$ apresentando melhores desempenho gerais quando testado sobre 30 execuções do algoritmo. Como foi demonstrado nos gráficos Box-plot e nas tabelas de média e desvio padrão para cada um desses parâmetros.

A análise do gráfico de convergência mostra que o algoritmo apresenta maior volatilidade nas fases iniciais (temperatura alta), mas se estabiliza gradualmente à medida que a temperatura diminui, convergindo para soluções com a menor distância total entre as cidades até o limite de iterações definido.

Além disso, esperava-se que o algoritmo de *Simulated Annealing* fosse capaz de encontrar o percurso mínimo ótimo para as duas instâncias. No entanto, isso não foi possível. Mesmo ajustando diferentes parâmetros do algoritmo, como os valores de SA_Max, a escolha da rotina de resfriamento e o valor de Swap_Factor, foi possível apenas obter uma distância total aproximada do valor ótimo, que seria 426 para 51 cidades e 21282 para 100 cidades.

Referências

Mathworks (2024). What is simulated annealing? https://www.mathworks.com/help/gads/what-is-simulated-annealing.html.