Relatório do Trabalho Prático de Computação Evolutiva

Arthur Negrão[1], **Igor Machado**[1]

[1]Departamento de Computação – Universidade Federal de Ouro Preto (UFOP)
Ouro Preto – MG – Brasil

{arthur.negrao,igor.machado}@aluno.ufop.edu.br,

1. Introdução

O Problema do Caixeiro Viajante (PCV), em inglês *Travelling Salesman Problem* (TSP), é um conhecido problema combinatório nos ramais de pesquisa operacional. De maneira informal, podemos defini-lo da seguinte forma: com base em um conjunto de cidades, o mesmo consiste em encontrar o caminho com a menor distância de forma que, partindo de uma cidade arbitrária, o caixeiro visite todas outras cidades exatamente uma vez e então retorne ao ponto de partida [Gavish and Graves 1978].

Ele foi formulado a primeira vez em 1930 e até hoje é um dos mais estudados problemas em seu ramo, visto a vastidão de aplicações no mundo real que o mesmo tem, como por exemplo a otimização de sistemas de entrega ou até mesmo de circuitos eletrônicos. Ademais, o PCV é um problema NP-Difícil, o que torna sua resolução ótima inviável para grandes instâncias [Goyal 2010].

Através deste trabalho prático, realizamos uma implementação de algoritmo genético para resolução do problema. Realizou-se a experimentação em múltiplas bases de dados com o *fine-tune* de hiper-parâmetros através de testes empíricos.

1.1. Organização do Relatório

Este relatório é composto pelas seguintes seções: na Seção 2 é apresentada a definição formal do problema; na Seção 3 é apresentado o desenvolvimento do trabalho através de seus processos metodológicos; na Seção 4 são apresentados os resultados e é realizada uma discussão sobre os mesmos; e, por fim, na Seção 5 é apresentada a conclusão sobre o trabalho.

2. Formulação do Problema

[Langevin et al. 1990] apresenta a seguinte definição matemática para o PCV:

$$min. \sum_{i} \sum_{j} c_{ij} x_{ij} \tag{1}$$

$$s.a. \sum_{i} x_{ij} = 1, \quad j = 1, ..., n, \tag{2}$$

$$\sum_{j} x_{ij} = 1, \quad i = 1, ..., n, \tag{3}$$

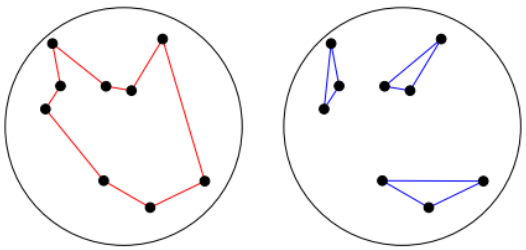
$$u_i - u_j + n x_{ij} \leq n - 1 \quad \forall j \neq 1, i \neq j, \tag{4}$$

$$0 \leq x_{ij} \leq 1, \quad \forall i, j, \tag{5}$$

$$x_{ij} \in \mathbb{Z}, \quad \forall i, j. \tag{6}$$

Iniciando a explicação da definição do problema com a Equação 1, ela define a função objetivo do problema, que consiste em minimizar a distância total percorrida. Já as equações 2 e 3 estabelecem que cada cidade deverá ser visitada uma única vez. A Equação 4, por sua vez, evita a ocorrência das subrotas (*subtours*). A Figura 1 exemplifica de maneira gráfica este fenômeno.

Figura 1. Exemplo de *subtour*. Veja que, quando ocorrem subrotas, o caminho gerado não é contínuo, mas sim é um conjunto de múltiplos ciclos que não se comunicam. Fonte: [Stack Exchange 2022]



Por fim, as equações 5 e 6 asseguram que x_{ij} ou assumirá o valor 1 - indicando que houve o deslocamento da cidade i para a cidade j - ou assumirá o valor 0 - indicando que não houve o deslocamento da cidade i para a cidade j.

3. Desenvolvimento

Esta seção apresenta o processo metodológico para execução deste trabalho. Em mais detalhes, a Seção 3.1 apresenta os conjuntos de dados utilizados; a Seção 3.2 apresenta o formato adotado para representação da solução; a Seção 3.3 apresenta o operador de cruzamento adotado; a Seção 3.4 apresenta o operador de mutação adotado; a Seção 3.5 apresenta o operador de seleção adotado; a Seção 3.6 apresenta o algoritmo genético utilizado; e a Seção 3.7 apresenta o processo de seleção de hiper-parâmetros. O código utilizado para experimentação está disponível no *GitHub*[1].

3.1. Base de Dados

Os dados utilizados foram extraídos da TSPLIB[2], mais especificamente foram utilizadas duas instâncias simétricas (*i.e.* $d_{ij} = d_{ji}$), a *Brazil58* (Br58) e a *Brg180*; e uma assimétrica (*i.e.* d_{ij} não necessariamente é igual a d_{ji}), a *Rbg443*. A Tabela 1 apresenta informações sobre o tamanho e o valor ótimo de cada instância.

[1]Disponível em https://github.com/arthurnfmc/TSPGenetic
[2]Disponível em http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/index.html

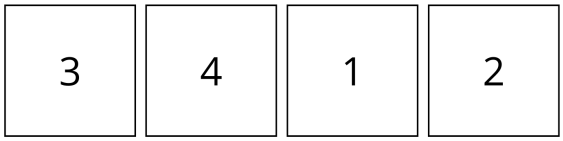
Tabela 1. Informações sobre os *datasets* utilizados.

Nome	N. de Cidades	Valor Ótimo da FO
Brazil58	58	25395
Brg180	180	1950
Rbg443	443	2720

3.2. Representação da Solução

A solução do problema foi representada como uma permutação das n cidades. Desta forma, o espaço de busca restringe-se ao espaço de soluções viáveis, evitando a necessidade de codificação direta das restrições ou mesmo a punição de inviabilidades na FO. A Figura 2 exemplifica essa forma de representação.

Figura 2. Exemplo de representação da solução. Observe que ela é uma permutação dos valores de 1 a n.



3.3. Operador de Cruzamento

O operador de cruzamento utilizado é o de *crossover* ordenado. [Alsedà 2024] descreve o processo conforme os seguintes passos sequenciais:

1. Selecione aleatoriamente uma *substring* de um pai.
2. Copie esta *substring* ao filho.
3. Delete as cidades que já estão na *substring* do segundo pai.
4. Preencha as cidades nas posições não fixadas da esquerda para direita para produzir um filho.

Este operador foi selecionado pois não introduz inviabilidades na solução e é capaz de guiar o algoritmo na exploração de um grande espaço de busca. A Figura 3 ilustra o processo descrito.

Figura 3. Exemplo visual do *crossover* ordenado. Fonte: [Alsedà 2024]

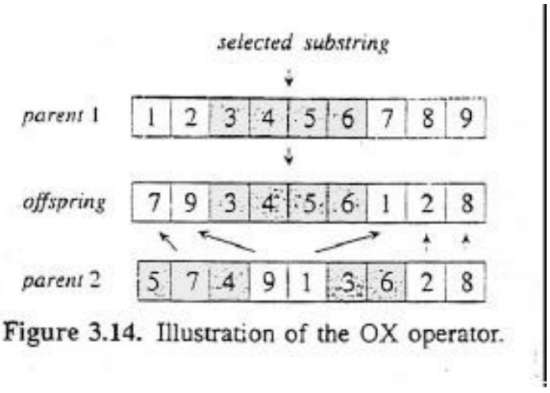


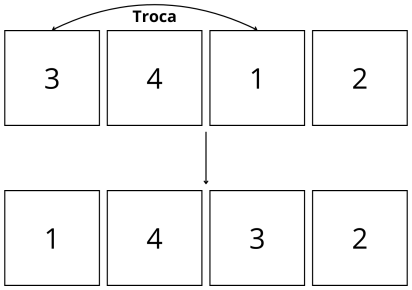
Figure 3.14. Illustration of the OX operator.

3.4. Operador de Mutação

O operador de mutação adotado foi o de troca de cidades. Ou seja, na ocorrência de uma mutação, ele seleciona duas cidades i e j e troca suas ordens de visitação. Dado que irá ocorrer uma mutação, a probabilidade de troca de um gene é de 0,2.

A Figura 4 exemplifica visualmente este operador.

Figura 4. Exemplo do operador de mutação.

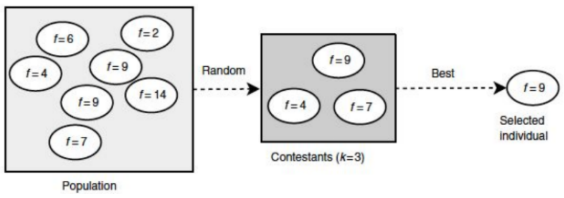


Este operador foi selecionado pois não introduz inviabilidades na solução e é capaz de guiar o algoritmo na exploração de um grande espaço de busca.

3.5. Operador de Seleção

O operador de seleção escolhido foi um torneio de tamanho 3. Isto é, dado um conjunto de indivíduos, são selecionados aleatoriamente três e, destes três, o indivíduo que apresentar a maior aptidão sobrevive. A Figura 5 exemplifica tal processo.

Figura 5. Exemplificação de um torneio $k = 3$. Fonte: [de Souza 2024].



Este operador foi selecionado pois não introduz inviabilidades na solução e é capaz de guiar o algoritmo na exploração de um grande espaço de busca.

3.6. Algoritmo Utilizado

A estratégia evolutiva adotada foi a $\mu + \lambda$, que consiste em uma estratégia evolutiva onde a seleção da geração $n + 1$ é feita com base nos pais e filhos da geração n. O Algoritmo 1 apresenta em minúcias o processo lógico do mesmo.

O algoritmo foi adotado devido aos bons resultados apresentados no trabalho de [Filho et al. 2017], onde os autores trabalharam com o processo de otimização do calendário esportivo do campeonato de futebol brasileiro.

Fonte: [Filho et al. 2017]

Algoritmo 1: Algoritmo ES($\mu + \lambda$)

```
Entrada: μ, λ, maxGen
Saída  : melhorInd
gen ← 0;
P ← geraPopInicial ();
melhorInd ← atualizaMelhor (P);
enquanto gen < maxGen faça
    D ← {} ; // Descendentes
    para cada indivíduo i ∈ P faça
        para λ/μ faça
            j ← criaDescendente (i, p_m);
            avaliarSolucao (j);
            inserirSolucao (D, j);
        fim
    fim
    definirSobreviventes (P, D, μ, λ) ; // ES (μ + λ)
    melhorInd ← atualizaMelhor (P);
    gen ← gen + 1;
fim
retorne melhorInd
```

3.7. Seleção de Hiper-Parâmetros

A seleção dos hiper-parâmetros foi realizada em pares. Isto é, dado um conjunto $W = \{(v_i, v_j), ...\}$, onde v_i é um valor para o hiper-parâmetro i e v_j é um valor para o hiper-parâmetro j, será selecionada "a melhor combinação" $(v_i, v_j) \in W$. Para determinar o que é a "melhor combinação" estabeleceu-se um teste que consiste em três execuções do algoritmo proposto. A configuração de hiper-parâmetros que apresentar a melhor FO na maioria das execuções será a selecionada.

Explicado o processo, os hiper-parâmetros do algoritmo foram organizados em pares da seguintes maneira:

- μ e λ: $W = \{(300, 200), (300, 300), (300, 500)\}$
- Probabilidade de *Crossover* e Probabilidade de Mutação: $W = \{(0,7; 0,3), (0,6; 0,4), (0,5; 0,5), (0,4; 0,6), (0,3; 0,7)\}$

Os testes foram conduzidos utilizando a base *brazil58*. As seções 3.7.1 e 3.7.2 apresentam os resultados para os testes com μ e λ e Probabilidade de *Crossover* e Probabilidade de Mutação, respectivamente.

3.7.1. μ e λ

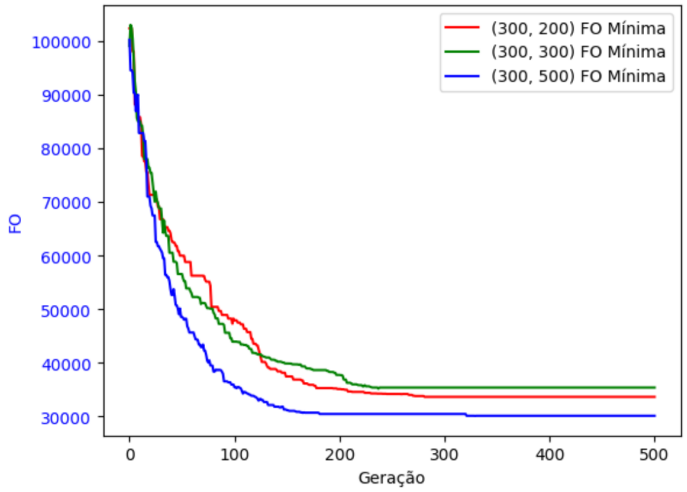
Os resultados para os testes μ e λ são apresentados a seguir nas figuras 6, 7 e 8.

Com base nos resultados obtidos, selecionou-se que $\mu = 300$ e $\lambda = 500$, pois estes valores proporcionaram os melhores resultados duas de três vezes.

3.7.2. Probabilidade de *Crossover* e Probabilidade de Mutação

Os resultados para os testes probabilidade de *crossover* e probabilidade de mutação são apresentados a seguir nas figuras 9, 10 e 11.

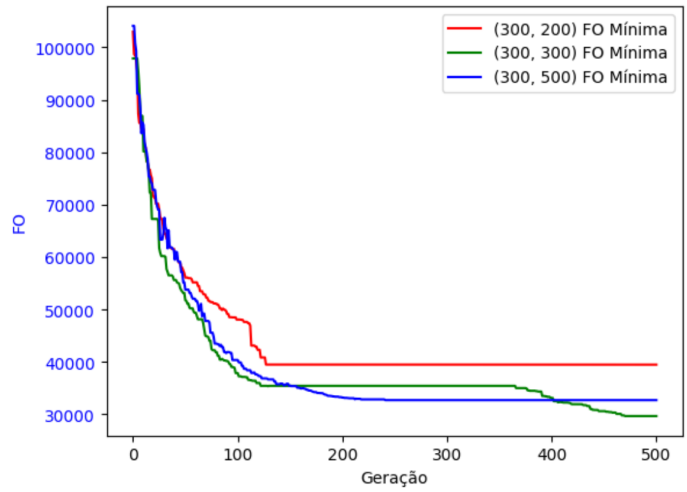
Figura 6. Execução $n = 1$ para μ e λ.




```
Melhores fitness para cada par:
(300, 200) -> 33638
(300, 300) -> 35184
(300, 500) -> 30049

Melhor par:
(300, 500)
```

Figura 7. Execução $n = 2$ para μ e λ.




```
Melhores fitness para cada par:
(300, 200) -> 39486
(300, 300) -> 29681
(300, 500) -> 32723

Melhor par:
(300, 300)
```

Com base nos resultados obtidos, selecionou-se que Prob. de *Crossover*$= 0,5$ e Prob. de Mutação$= 0,5$, pois estes valores proporcionaram os melhores resultados duas de três vezes.

Figura 8. Execução $n = 3$ para μ e λ.

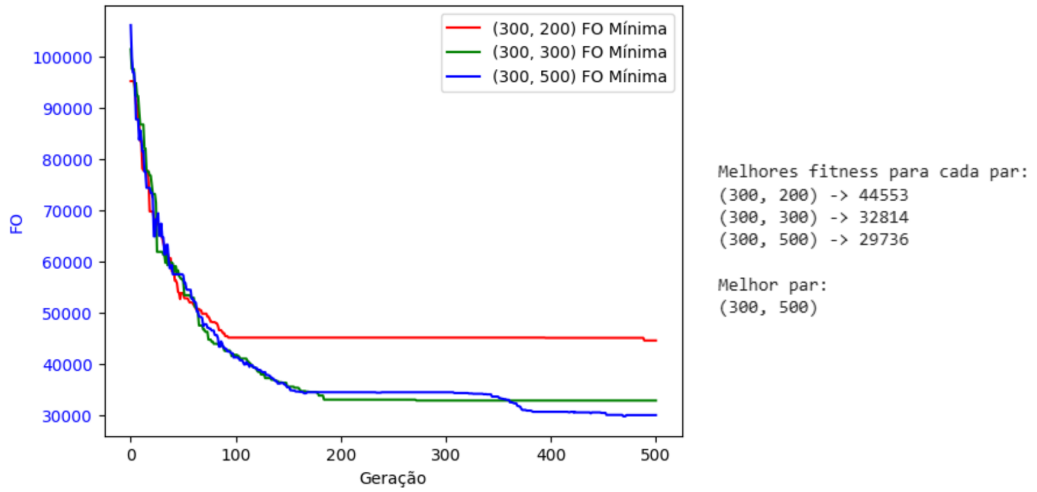
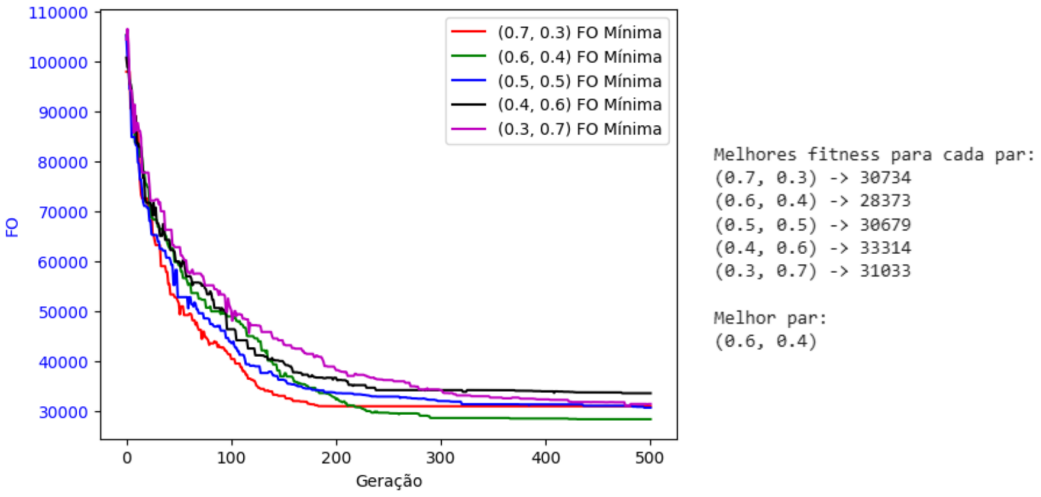


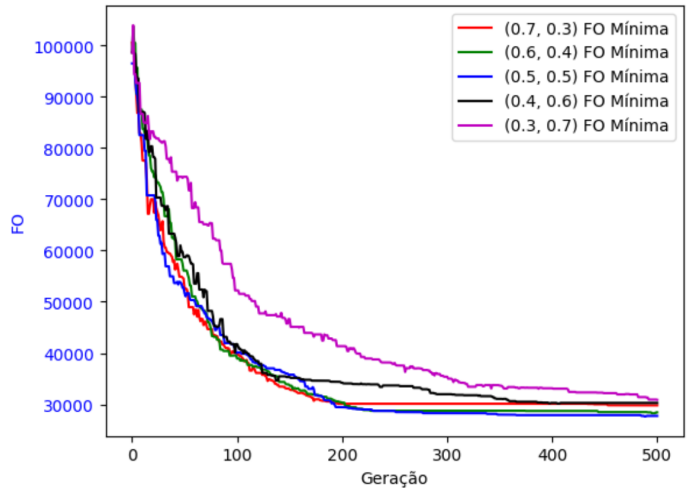
Figura 9. Execução $n = 1$ para Probabilidade de *Crossover* e Probabilidade de Mutação.



4. Resultados e Discussão

Posto todo processo metodológico descrito anteriormente, seguiu-se para a execução dos experimentos. Para relevância estatística dos resultados, cada experimento foi executado $n = 33$ vezes. Foram reportadas a média e desvio padrão dos resultados, além da pior e melhor FO obtidas. Ademais, também foi mensurado o tempo gasto para execução dos

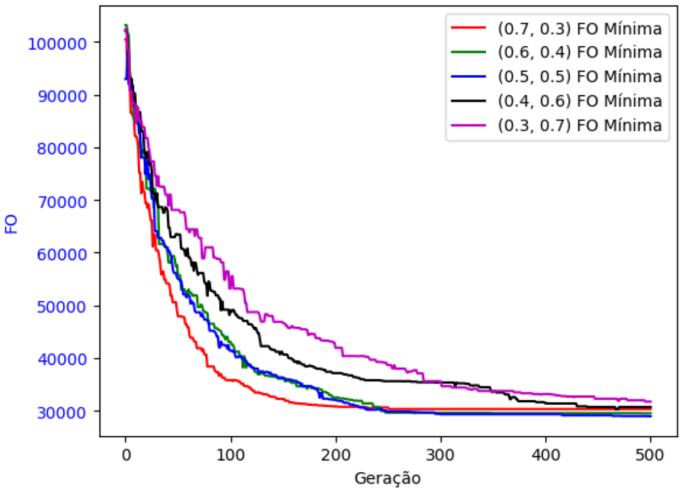
Figura 10. Execução $n = 2$ para Probabilidade de *Crossover* e Probabilidade de Mutação.




```
Melhores fitness para cada par:
(0.7, 0.3) -> 29805
(0.6, 0.4) -> 28266
(0.5, 0.5) -> 27585
(0.4, 0.6) -> 29842
(0.3, 0.7) -> 30883

Melhor par:
(0.5, 0.5)
```

Figura 11. Execução $n = 3$ para Probabilidade de *Crossover* e Probabilidade de Mutação.




```
Melhores fitness para cada par:
(0.7, 0.3) -> 30006
(0.6, 0.4) -> 29384
(0.5, 0.5) -> 28971
(0.4, 0.6) -> 30308
(0.3, 0.7) -> 31741

Melhor par:
(0.5, 0.5)
```

experimentos.

E para a comparação com a literatura estabeleceu-se a métrica GAP, a qual é definida como

$$GAP = \frac{f' - f^*}{f^*} * 100, \tag{7}$$

onde f' é a melhor FO encontrada pelo algoritmo testado dentre todas as execuções e f^* é a FO ótima para o *dataset* utilizado. Essa métrica indica o quão distante a solução encontrada está do ótimo para o problema e, portanto, quanto menor o GAP melhor é a FO. A Tabela 2 apresenta valores obtidos para todas instâncias experimentais, e as figuras de 12 a 14 apresentam os *boxplots* dos resultados. Tenha atenção para a escala dos *boxplots* para evitar comparações errôneas entre as bases.

Tabela 2. Resultados experimentais para cada base de dados em termos de FO e tempo gasto.

Nome	FO Máx.	FO Min.	FO	GAP	Tempo (s) Máx.	Tempo (s) Min.	Tempo (s)
Brazil58	31.639,00	26.319,00	29.329,61 ± 1.170,08	3,64%	32,34	29,94	30,72 ± 0,62
Brg180	102.340,00	32.880,00	54.515,15 ± 13.516,24	1.586,15%	87,18	75,94	78,65 ± 3,08
Rbg443	4.969,00	4.742,00	4.871,85 ± 53,27	74,34%	187,73	184,08	185,84 ± 0,95

Figura 12. *Boxplot* para a *Brazil58*.

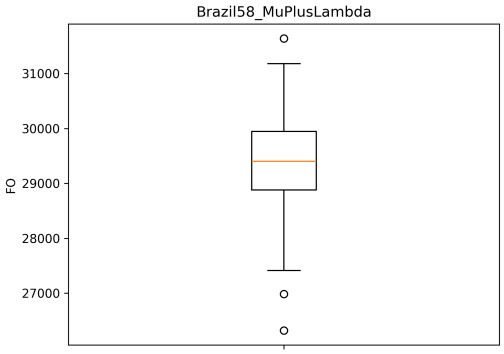


Figura 13. *Boxplot* para a *Brg180*.

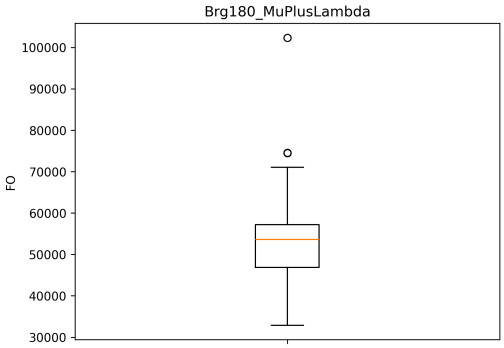
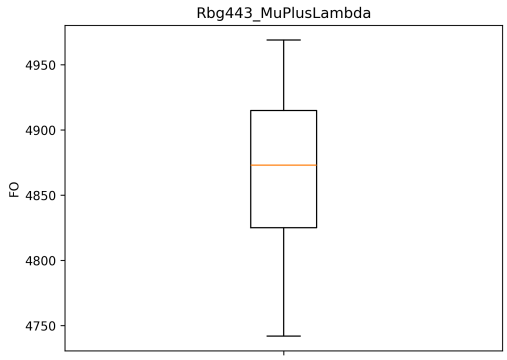


Figura 14. *Boxplot* para a *Rbg443*.



Iniciando a análise com enfoque sobre a métrica GAP, é possível afirmar para a *Brazil58* que os resultados obtidos através do algoritmo $\mu + \lambda$ aproximaram-se consideravelmente do ótimo, haja visto que a métrica assentou-se na faixa de 3,64%.

Por outro lado, trazendo o enfoque da análise para as outras duas bases, percebe-se uma piora considerável na métrica GAP, indicando que o algoritmo proposto não consegue aproximar-se tão bem do ótimo - especialmente na base *Brg180*, onde a métrica excedeu 1500%. Tal fato pode ser explicado pelo aumento considerável da complexidade destas bases em relação à distribuição dos valores da matriz de pesos: conforme pode ser visto nas imagens 15, 16 e 17, que ilustram a matriz de pesos de cada instância numa relação onde quanto mais vermelho maior é a distância de i para j, a distribuição da *Brazil58*, além de mais simples em termos de dimensionalidade, não forma padrões tão complexos quantos ambas as outras.

Figura 15. Matriz de pesos da *Brazil58*.

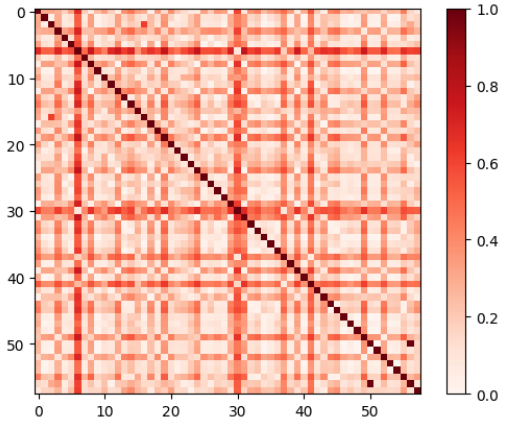


Figura 16. Matriz de pesos da *Brg180*.

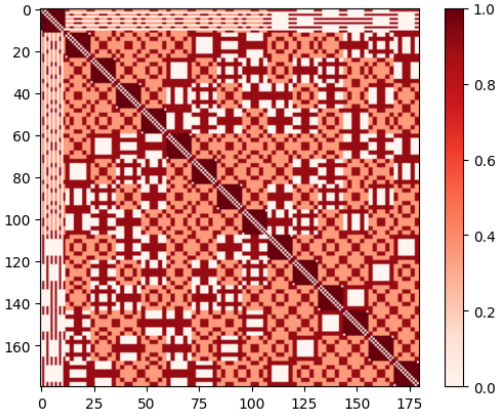
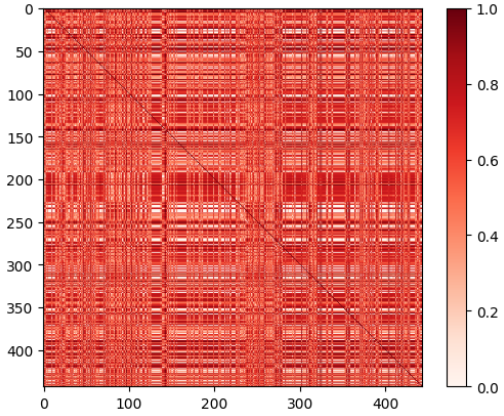


Figura 17. Matriz de pesos da *Rbg443*.



Ainda falando das matrizes de pesos, é possível observar também que, apesar de mais complexa em termos de dimensionalidade, a *Rbg443* estabelece padrões de distribuição mais simples que a *Brg180*. Isso justifica o fato de que, apesar de ser maior, o algoritmo proposto aproximou-se mais do ótimo na *Rbg443* do que na *Brg180* (mais especificamente, a primeira obteve um GAP de 74,34% enquanto a segunda atingiu 1586,15%).

Avançando para a análise de estabilidade dos algoritmos, tendo como base o desvio padrão percentual relativo à média, isto é $D_{perc} = \frac{Desv.\ Pad.}{Media}$, observou-se que: **1.** o

melhor resultado foi encontrado em *Rbg443*, apresentando-se como 1,09%; **2.** o segundo melhor foi em *Brazil58*, atingindo o valor de 3,99%; **3.** e o pior valor foi encontrado na *Brg180*, com um aumento de mais de 20% no mesmo em relação às bases anteriores, atingindo o valor de 24,79%. Tal fato é, portanto, um indicativo de que bases com distribuições complexas na matriz de pesos podem gerar instabilidades no algoritmo ao longo das execuções.

Ainda sobre esta questão de estabilidade, a Figura 12 revela a detecção de três *outliers* durante a execução da *Brazil58*, enquanto a Figura 13 revela a detecção de dois *outliers*. Apesar de revelar certa instabilidade dos algoritmos, observa-se que, para a *Brazil58*, dois de três *outliers* são "surpresas positivas", haja vista que estão abaixo do limite inferior e o TSP é um problema de minimização. Por outro lado, para a *Brg180*, ambos *outliers* são "surpresas negativas", já que ambas excedem o limite superior (sendo que uma até mesmo ultrapassa 100.000 de FO).

Por fim, em análise da questão do tempo de execução, é possível afirmar que a proposta genética é consideravelmente estável, havendo pouca diferença entre os tempos gastos para cada execução. Como pode ser visto na tabela anterior, para a *Brazil58* e a *Rbg443* o desvio padrão em ambos os casos é de menos de um segundo (respectivamente 2,01% e 0,51%). E mesmo para a *Brg180*, que consiste no pior caso, o desvio padrão é de apenas 3 segundos, isto é, 3,91%.

5. Conclusão

Este trabalho conduziu uma experimentação prática sobre o Problema do Caixeiro Viajante. Conduziu-se sua definição formal e explicação na Seção 2 e, em seguida, o processo metodológico foi descrito na Seção 3. Apresentaram-se as bases de dados utilizadas - *Brazil58*, *Brg180* e *Rbg443* - e os elementos constituintes da Estratégia Evolutiva (EE) $\mu + \lambda$ adotada. Os resultados, os quais são apresentados na Seção 4, revelaram o potencial da EE para solução do problema. Através da métrica GAP, observou-se a aproximação considerável do ótimo na *Brazil58*. Entretanto, para as outras bases os resultados não foram tão próximos assim, fato explicado pela complexidade da distribuição da matriz de pesos destes casos. Ademais, a EE avaliada demonstrou-se estável em termos de tempo gasto para todas instâncias; e, em termos de FO, demonstrou-se estável para a *Brazil58* e a *Rbg443*, entretanto para a *Brg180* notou-se um incremento de mais de 20% no desvio padrão relativo.

Por fim, para trabalhos futuros, pode-se realizar a experimentação com variações multiobjetivo do PCV, que podem introduzir dinâmicas novas como a maximização de lucro, a existência de janelas de entrega e a movimentação de cidades (*Kinetic-Based TSP*) [Ilavarasi and Joseph 2014].

Referências

Alsedà, L. (2024). Genetic operations. Disponível em https://mat.uab.cat/ alseda/MasterOpt/GeneticOperations.pdf . Acesso em março de 2025.

de Souza, F. S. H. (2024). Aptidão, seleção e gerenciamento de populações. Disponível nos slides presentes no Moodle. Slides traduzidos do livro Eiben & Smith. Acesso em março de 2025.

Filho, A. J. M., de Oliveira, F. B., and Alexandre, R. F. (2017). Um método baseado em estratégia evolutiva para a resoluç ao do problema de agendamento em competiçoes esportivas. *XLIX Simpósio Brasileiro de Pesquisa Operacional*.

Gavish, B. and Graves, S. C. (1978). The travelling salesman problem and related problems.

Goyal, S. (2010). A survey on travelling salesman problem. In *Midwest instruction and computing symposium*, pages 1–9.

Ilavarasi, K. and Joseph, K. S. (2014). Variants of travelling salesman problem: A survey. In *International conference on information communication and embedded systems (ICICES2014)*, pages 1–7. IEEE.

Langevin, A., Soumis, F., and Desrosiers, J. (1990). Classification of travelling salesman problem formulations. *Operations Research Letters*, 9(2):127–132.

Stack Exchange (2022). Interpretation of a subtour elimination constraint in the tsp. Disponível em https://or.stackexchange.com/questions/8120/interpretation-of-a-subtour-elimination-constraint-in-the-tsp . Acesso em março de 2025.