Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

Algoritmos evolutivos para solução do Problema do Caixeiro Viajante

Prof. Eduardo do Valle Simões

Thiago Ribeiro Goveia 10835942

São Carlos
Janeiro de 2022

Introdução

O problema do caixeiro viajante é amplamente conhecido e estudado em diversas frentes de pesquisa. Possuindo amplas aplicações, esse problema é classificado como NP difícil

[1], o que significa que seu tempo de solução é polinomial e só pode ser resolvido não deterministicamente. Algoritmos evolutivos são ótimos candidatos para a obtenção de soluções que, quando não ótimas, se aproximam razoavelmente da solução ótima, além de representarem uma ótima oportunidade para o estudo desses problemas.

Problema do caixeiro viajante

O problema do caixeiro viajante consiste basicamente de um grafo ponderado que deve ser percorrido por completo com cada nó sendo visitado uma vez. A solução do problema do caixeiro viajante, então, se trata do menor ciclo Hamiltoniano possível para um dado grafo.

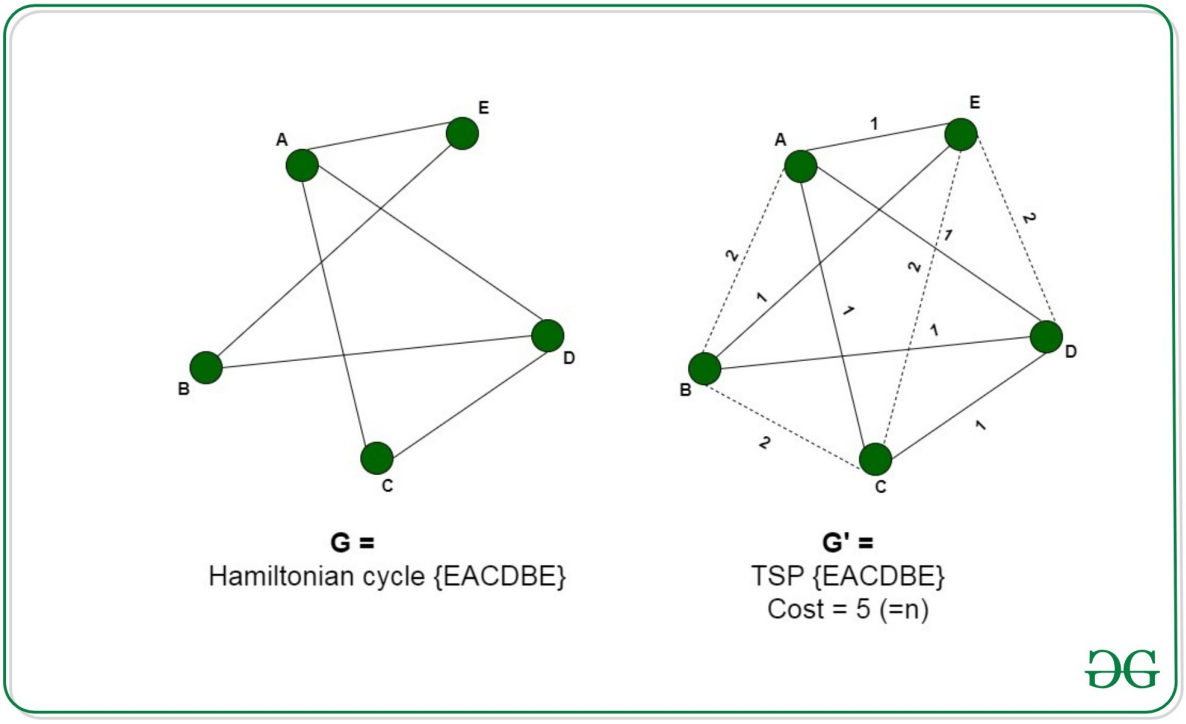

Imagem 1: Exemplo de um ciclo Hamiltoniano de um grafo (Extraído de [2])

As condições para a existência de um ciclo hamiltoniano de um grafo não serão abordadas aqui, sendo suficiente considerar que um grafo completo não direcional sempre terá pelo menos um ciclo hamiltoniano.

Lidando com grafos não completos

Grafos não completos representam um desafio pois nem toda aresta existe para ser percorrida, o que significa que a existência de um caminho Hamiltoniano não é garantida. Para lidar com este problema podemos abandonar a restrição de visitar cada nó do grafo apenas uma vez, permitindo caminhos reversos. O grafo pode ser completado aplicando-se algoritmos de determinação da menor distância entre dois nós, como o algoritmo de Dijkstra [2]. Ao preservar o menor caminho entre dois vértices podemos criar um novo caminho a partir do resultado da evolução.

Algoritmos evolutivos

Algoritmos evolutivos são bons candidatos para a solução de problemas de otimização, por sua capacidade de explorar o universo de soluções de forma probabilística, ajustando-se a cada iteração para se aproximar de uma solução ótima. A exploração do problema do caixeiro viajante demanda a aplicação de algumas técnicas de manipulação do caminho percorrido. A população de AGs é definida por um grupo de caminhos possíveis, que é avaliado quanto ao peso total das arestas percorridas, as técnicas de evolução são aplicadas a cada AG para a obtenção da próxima geração. As técnicas de manipulação de caminho utilizadas neste trabalho são descritas a seguir:
Para fins de compreensão dessa próxima seção, entende-se por eliminação, o ato de embaralhar aleatoriamente a ordem dos nós de um caminho.

Representação de um caminho

Um caminho pode ser representado por um vetor de números, cada um representando um nó do grafo.

$$(1\,4\,3\,5\,2\,6), (1\,2\,4\,3\,5\,6)$$

Imagem 2: Exemplos de caminhos de um grafo de 6 nós. (Extraído de [3])

Métodos de seleção

Dois métodos de seleção foram aplicados: Elitismo e Torneio de dois, cada um descrito a seguir.

Elitismo

O elitismo é realizado, calculando-se a distância total percorrida pelos AGs em cada caminho, selecionando-se a menor delas, que representa o AG escolhido para reprodução.

Torneio de dois

O torneio de dois é realizado escolhendo dois AGs ao acaso, calculando seus respectivos caminhos e tomando o melhor dos dois para reprodução.

Reprodução Sexuada

A reprodução sexuada dos AGs se mostrou um tópico de estudo interessante pois existem várias formas de combinar caminhos de dois AGs de forma a obter um caminho completamente novo, dependendo da técnica de crossover utilizada, diferentes graus de semelhança com os pais podem ser obtidas. A bibliografia oferece um grande acervo de técnicas de crossover, duas das quais foram implementadas neste trabalho. Todas as técnicas abordadas foram implementadas de forma que não existam elementos duplicados num caminho.

Crossover de posição alternada

Proposto por Larrañaga em 1996, esse crossover pega elementos dos pais alternadamente até que seja preenchido o filho.

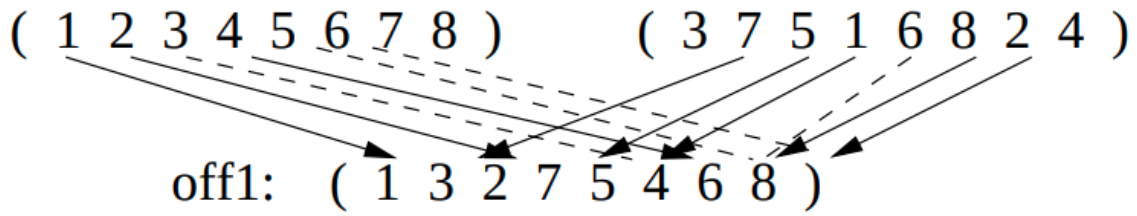

Imagem 3: Representação do crossover de posição alternada (Extraído de [3])

Crossover de preservação máxima

Proposto por Muhlenbein em 1988 [ ] este algoritmo seleciona um sub-caminho de um dos pais e adiciona ao começo do filho, preenchendo o restante com elementos do outro pai quando possível

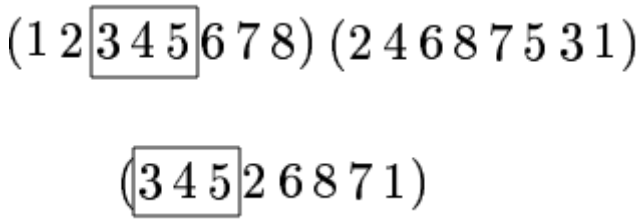

Imagem 4: Representação de preservação máxima

Mutações

Três formas de mutação foram implementadas, todas elas são baseadas na troca de posição de elementos dentro do vetor.

Mutação por deslocamento

Nesta mutação seleciona-se um trecho de determinado tamanho dentro do vetor, que é trocado de posição dentro deste mesmo vetor. A taxa de mutação determinou o tamanho do trecho selecionado.

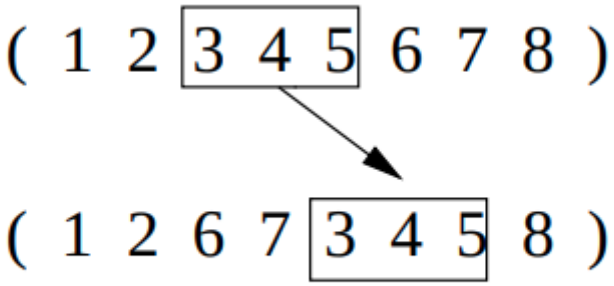

Imagem 5: Representação da mutação por deslocamento (Extraído de [3])

Mutação por troca

Nesta mutação dois ou mais elementos são trocados de lugar dentro do vetor. A taxa de mutação determina quantas vezes essa mutação é aplicada em um caminho.

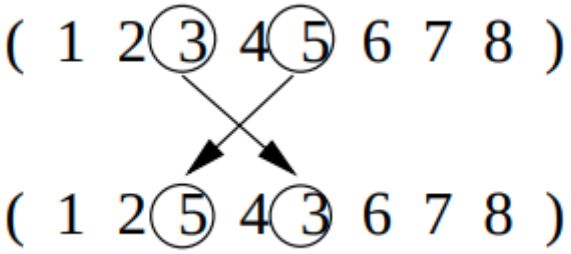

Imagem 6: Representação da mutação por troca (Extraído de [3])

Mutação por inserção

Nesta mutação um elemento é removido do vetor e inserido em uma nova posição. A taxa de mutação determina quantas vezes essa mutação é aplicada em um caminho.

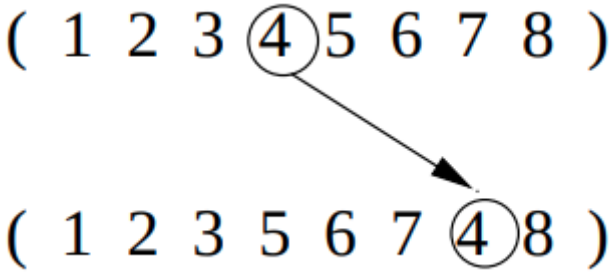

Imagem 7: Representação da mutação por inserção (Extraído de [3])

Reprodução Assexuada

Também foi abordada a possibilidade de que nenhum crossover seja realizado, utilizando-se assim apenas técnicas de mutação para diferenciação.

Métodos de eliminação

Foram implementados dois métodos de eliminação, predação e genocídio.

Predação

A predação pode ser atingida pegando o AG de menor escore, eliminando-o.

Genocídio

O genocídio pode ser atingido eliminando todos os AGs de uma geração.

Seleção das técnicas de evolução

A seleção da melhor técnica de evolução para um dado grafo se mostrou uma tarefa difícil, uma vez que cada grafo é único, sendo improvável que uma única técnica seja ótima para todos. Levando isso em consideração, foi desenvolvido um segundo algoritmo evolutivo que evolui para selecionar as melhores técnicas para um grafo.

Segundo AG

O segundo AG é representado por um vetor de 10 posições dividido em 4 genes.

$$(1\,0\,|\,1\,0\,0\,0\,|\,1\,0\,0\,|\,1\,0)$$

Imagem 8: Vetor de seleção

Cada elemento pode valer 1 ou zero, representando que está ligado ou desligado e dentro de cada fenótipo apenas um elemento pode valer 1, o significado de cada fenótipo, na ordem, e elementos é definido a seguir:

- Gene de seleção - Pelo menos um valor igual 1
    - Ag[0]: Elitismo
    - Ag[1]: Torneio
- Gene de crossover - Pelo menos um valor igual 1
    - Ag[2]: Crossover de posição alternada
    - Ag[3]: Crossover baseado em posição
    - Ag[4]: Sem crossover
- Gene de mutação - Pelo menos um valor igual 1
    - Ag[5]: Mutação por deslocamento
    - Ag[6]: Mutação por troca
    - Ag[7]: Mutação por inserção
- Gene de eliminação - Ambos podem valer 0
    - Ag[8]: Predação
    - Ag[9]: Genocídio

A taxa de mutação dos processos de mutação também pode ser alterada pelo AG ao ser incrementada ou decrementada segundo os seguintes critérios:

- Incremento: Se o escore do ag está dentro de um threshold comparado ao melhor escore
- Decremento: Se o escore do ag está fora de um threshold comparado ao melhor escore

Reprodução

A reprodução desse AG é feita de forma assexuada, realizando mutação dos genes. A mutação é atingida sorteando um cromossomo para ser ligado, seguido de atualização da taxa de mutação.

Avaliação

O escore de um AG é determinado rodando 300 gerações de evolução dos AGs de caminho, o melhor escore dessas gerações é tomado como o escore do AG controlador.

Resultados

O algoritmo foi executado para um grafo completo de 1000 nós nos modos, com e sem técnicas de eliminação. Os escores do melhor ag de cada geração estão demonstrados no gráfico. Na Tabela 1 encontram-se o escore e a geração de ambos os modos.

Modo	Escore	Geração
Com eliminação	5215	2120757
Sem eliminação	3864	604903

Tabela 1: Resultados obtidos para os AGs nos modos com e sem eliminação

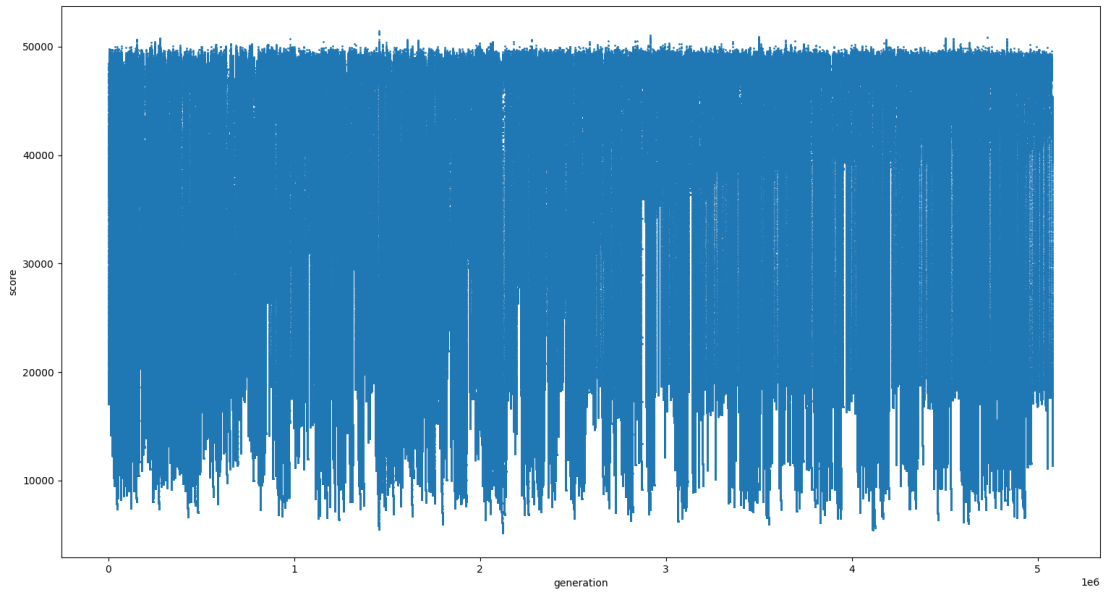

Imagem 9: G com técnicas de eliminação sem zoom.

Este AG foi evoluído por mais de 5 milhões de gerações e não mostrou tendência de otimização do resultado, observando o grafo mais de perto podemos ver que esse comportamento é causado pela eliminação muito frequente do melhor de todos.

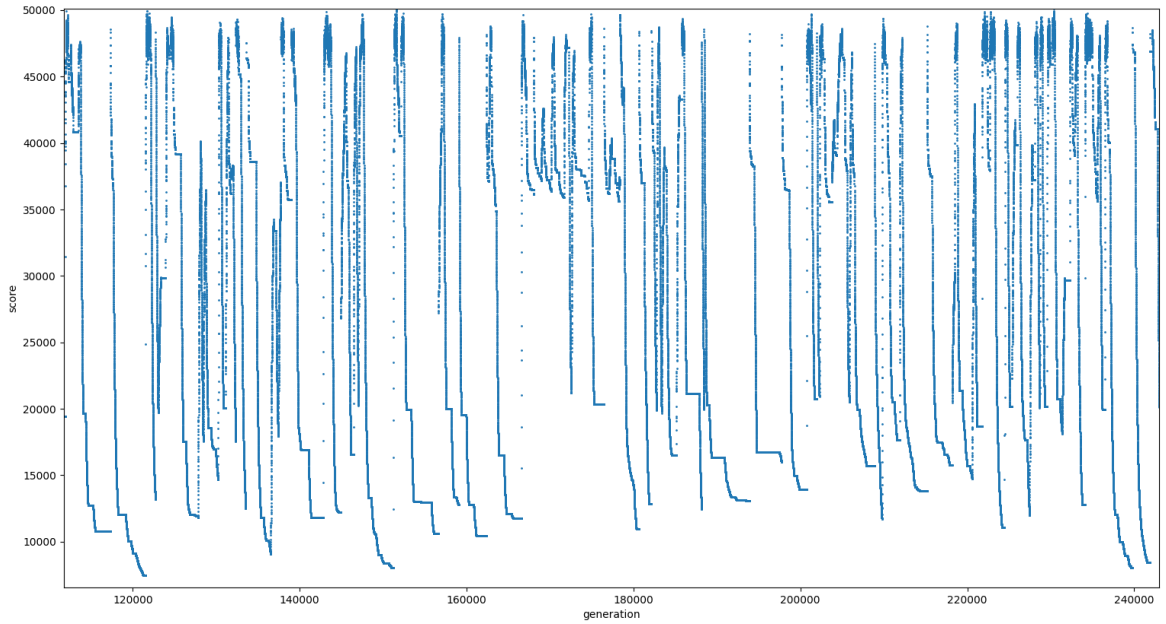

Imagem 10: AG com técnicas de eliminação com zoom.

É possível ver em uma seção magnificada do gráfico que existe a tendência de queda do escore que é interrompida sucessivamente. Foi teorizado que esse comportamento é causado pela ocorrência muito frequente de genocídio e/ou torneio de dois. Para confirmar a teoria, o algoritmo foi rodado novamente com esses métodos desativados.

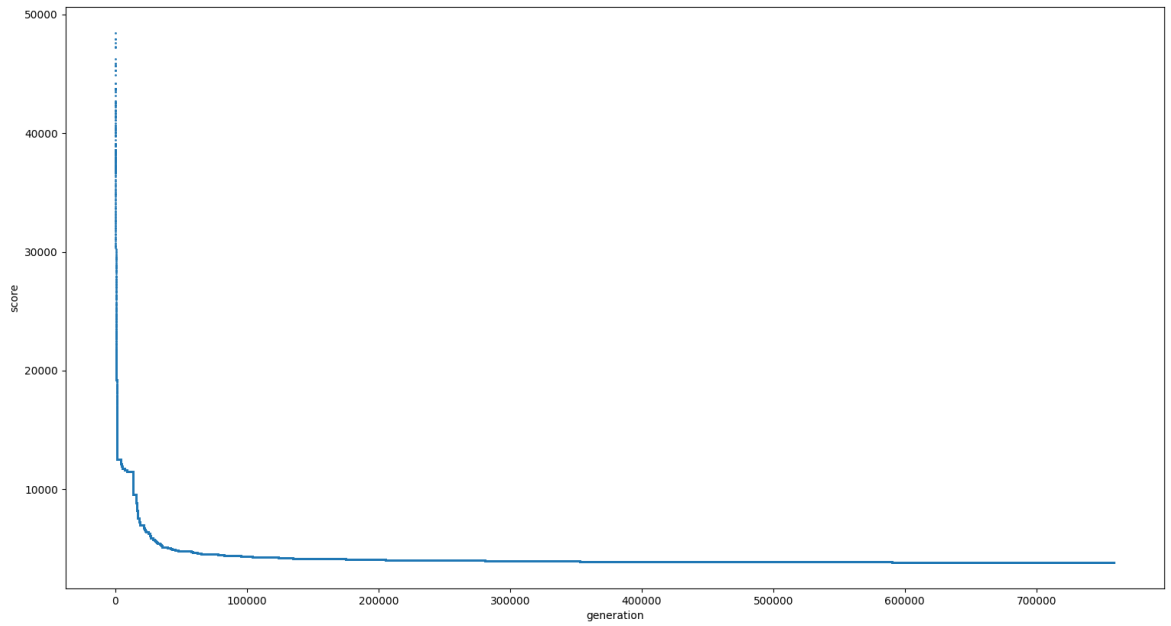

Imagem 11: AG sem técnicas de eliminação;

Os resultados observados confirmaram a suposição anterior visto que, em uma fração das gerações, foi obtido um escore melhor que o AG anterior, sendo possível também observar a tendência de queda do escore.

Avaliação do AG controlador

Os escores obtidos do ag controlador são demonstrados a seguir:

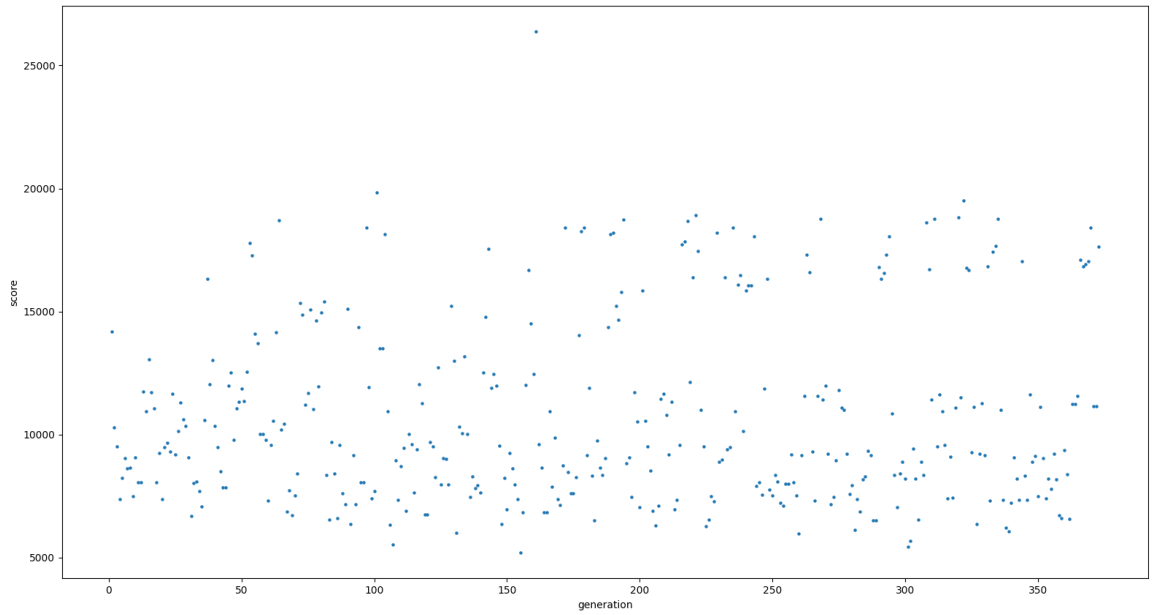

Imagem 12: AG controlador com técnicas de eliminação.

Assim como o outro AG, é fácil notar que o uso de genocidio e torneio de dois preveniram que este AG minimizasse seu escore.

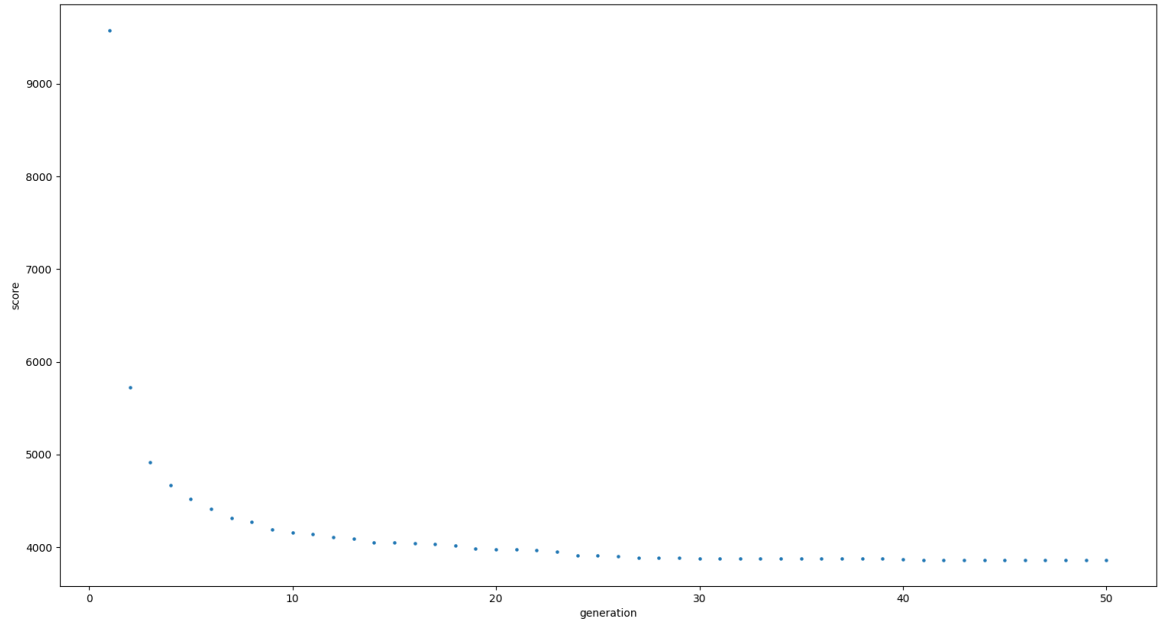

Imagem 13: AG controlador sem técnicas de eliminação.

Com genocídio e tornei desligados, assim como anteriormente, fica fácil observar a tendência de minimização do escore.

Prevalência das técnicas de evolução:

Utilizando o Ag sem técnicas de eliminação, foi possível analisar os a dominância de cada cromossomo, esses valores podem ser utilizados para avaliar a eficácia de cada método de evolução. Em ambos os genes de evolução houve predominância de um dos cromossomos.

Técnica	% de ocorrência entre os melhores
Crossover de posição alternada	1.4%
Crossover de preservação máxima	1.4%
Sem crossover	97.2%
Mutação por deslocamento	90.2%
Mutação por troca	1.4%
Mutação por inserção	8.4%

A análise do AG controlador sugere que técnicas de crossover são ineficientes para este grafo, uma hipótese que justifica tal ocorrência está no fato de que a distância percorrida pode variar muito com modificações entre elementos e as técnicas de crossover tendem a reorganizar completamente o caminho, mantendo pouco das características dos pais, agindo mais como técnica de eliminação do que de evolução.
A técnica de mutação de deslocamento dominou os resultados, talvez porque esta preserve trechos inteiros do AG, permitindo a exploração do caminho ótimo de forma mais controlada.

Erros cometidos e melhorias possíveis

Algumas das dificuldades encontradas durante o desenvolvimento estão listadas a seguir

- Tentativa inicial de resetar a população a cada troca de AG controlador
- Genocídio muito frequente
- Remover acidentalmente o melhor de todos da população
- Segmentation fault (Core dumped)

O algoritmo atual foi desenvolvido para provar a possibilidade de um AG para seleção de técnicas de evolução de um AG otimizador do problema do caixeiro viajante, portanto a otimização do código foi frequentemente colocada em segundo plano, focando atingir o objetivo. Como resultado, grafos grandes podem ser inviabilizados por limitações de performance. Portanto, algumas melhorias são sugeridas, entre elas:

- Tabelar resultados intermediários recorrentes para evitar avaliações repetidas
- Modificar código para a execução da avaliação de cada AG caixeiro em paralelo
- Criar uma população de AGs caixeiros para cada AG da população de AGs controladores, permitindo a evolução destes em paralelo, permitindo populações maiores de AG caixeiros.
- Minimizar cópias de vetores
- Minimizar travessia de vetores.

Conclusões

Dos resultado observados podemos concluir que o AG controlador não conseguiu aprender a usar o genocídio e o torneio de dois, o uso frequente destas técnicas introduziu muito ruído nos resultados, impossibilitando que a seleção atingisse valores ótimos. Podemos concluir que a perda do melhor de todos introduziu um impacto que não pode ser recuperado antes da realização de novas eliminações. A execução do algoritmo sem técnicas de eliminação apresentou melhores resultados, sugerindo que, para este problema, a mutação é suficiente para escapar dos mínimos locais e permite que o resultado convirja para um valor ótimo.

Referências

[1]https://en.wikipedia.org/wiki/NP-hardness
[2]https://www.geeksforgeeks.org/proof-that-traveling-salesman-problem-is-np-hard/
[3]https://www.dca.fee.unicamp.br/~gomide/courses/EA072/artigos/Genetic_Algorithm_TSPR_eview_Larranaga_1999.pdf