Comparação entre Algoritmo Genético e Algoritmo da Colônia de Formigas no Problema do Caixeiro Viajante

[1]Hugo Fernandes

[1]Depto de Computação e Tecnologia - Universidade Federal do Rio Grande do Norte (UFRN)
Caixa Postal: 59.300-000 - Caicó - RN - Brazil

[1] hugo.medeiros.fernandes@gmail.com

Abstract. *This report seeks to compare two heuristic methods, genetic algorithm and ant colony algorithm, in solving the problem of the traveling salesman.*
Resumo. Este relatório busca comparar dois métodos heurísticos, algoritmo genético e algoritmo da colônia de formigas, na resolução do problema do caixeiro viajante.

Keyworks: Genetic Algorithm, Ant Colony Algorithm, Traveling Salesman Problem, Python

1. O problema do Caixeiro Viajante

O Problema do Caixeiro Viajante (*PCV*) é um problema que tenta determinar a menor rota para percorrer uma série de cidades (visitando uma única vez cada uma delas), retornando à cidade de origem.

É um problema de otimização NP-difícil inspirado na necessidade dos vendedores em realizar entregas em diversos locais (cidades) percorrendo o menor caminho possível, reduzindo o tempo necessário para a viagem e os possíveis custos com transporte e combustível.

O espaço de busca do problema aumenta exponencialmente dependendo de n, o número de cidades:

$$(n - 1)!$$

2. Dados usados na Comparação

Foram gerados três arquivos, *'out1.txt' 'out2.txt' 'out3.txt'*, com coordenadas X e Y aleatórias, onde cada par ordenado corresponde ao local de uma cidade. Em todos os exemplos foram geradas 30 cidades.

3. Os Métodos Heurísticos

3.1. Algoritmos Genéticos (*Genetic Algorithm*)

Os algoritmos genéticos são uma técnica de busca que se inspira em conceitos naturais da biologia evolutiva.

Aplicando essa técnica ao problema do caixeiro viajante, buscamos em uma população de possíveis soluções, encontrar a melhor solução entre elas. Neste trabalho, foi convencionado manter a população em 100 soluções por cada geração.

Nos algoritmos genéticos, o processo evolutivo ocorre similar ao natural. Uma população inicial surge, e desse ponto em diante, ela começa a se reproduzir, gerando novas gerações. Da mesma forma que o processo natural, esporadicamente, indivíduos nascem com mutações, que podem te trazer vantagens ou desvantagens. Vale lembrar, que todo esse procedimento segue o princípio da seleção natural, onde indivíduos melhor adaptados tem mais vantagens competitivas.

Muito do que foi dito no parágrafo anterior, pode ser interpretado como analogias, no nosso problema, população é um conjunto de soluções, reprodução (*crossover*) é o cruzamento entre duas dessas soluções a fim de gerar outra, mutações são pequenas inversões na rota das cidades.

Alguns parâmetros sobre a execução do algoritmo genético, devem ser mencionados, como por exemplo, as taxas de crossover e mutação aplicadas. O crossover foi fixado em 50%, assim o cruzamento entre duas soluções, aproveitaria metade da informação das duas. Já a taxa de mutação foi fixada em 10%, assim a cada nova solução gerada, ela teria uma taxa de 10% de surgir com alguma mutação. Outro ponto que vale ser dito, é a proporção da população que é renovada em comparação com a anterior. Para isso, foi reaproveitada a taxa de crossover, assim metade da população seria renovada a cada geração.

Sobre os métodos de avaliação e seleção de uma solução, foi aplicada a distância euclidiana para avaliar cada indivíduo da população. Já sobre o método de seleção, a partir da avaliação, ou seja, quanto menor sua distância, ele teria mais chances de ser escolhido, o método utilizado aqui, é conhecido como método da roleta.

Por fim, o algoritmo executou os exemplos por 100 e 1000 gerações.

3.2. Algoritmo da Colônia de Formigas (*Ant Colony Algorithm*)

Os algoritmos de colônia de formigas, também conhecidos como *ACO* (*Ant colony optimization*), é uma técnica de otimização inspirada no comportamento das formigas durante sua buscam por comida.

O procedimento é bastante simples, um número *n* de formigas saem da colônia em busca de alimento. Quando alguma o encontra, começa a liberar um feromônio pelo caminho que fez. Formigas próximas, são atraídas por essa rota de feromônio, consequentemente, o reforçando. Depois de algum tempo, as formigas tendem a convergir em um caminho cheio de feromônio.

A lógica principal da técnica, é que esse feromônio possui em taxa de evaporação, fazendo com que caminhos mais longos e/ou que não possuam muitas formigas liberando feromonios, tendem a serem esquecidos pela evaporação natural da toxina ao longo do tempo.

Nos exemplos deste trabalho, foram fixados alguns parâmetros, como a taxa de evaporação do feromônio em 50%, ou seja, a cada iteração os feromônios liberados tem 50% de chance de serem evaporados. Outro parâmetro que vale ser mencionado, é a quantidade de formigas, que foi adotado como o mesmo número de cidades, no caso dos exemplos, 30 formigas.

Por fim, o algoritmo executou os exemplos por 100 e 1000 iterações.

4. Resultados

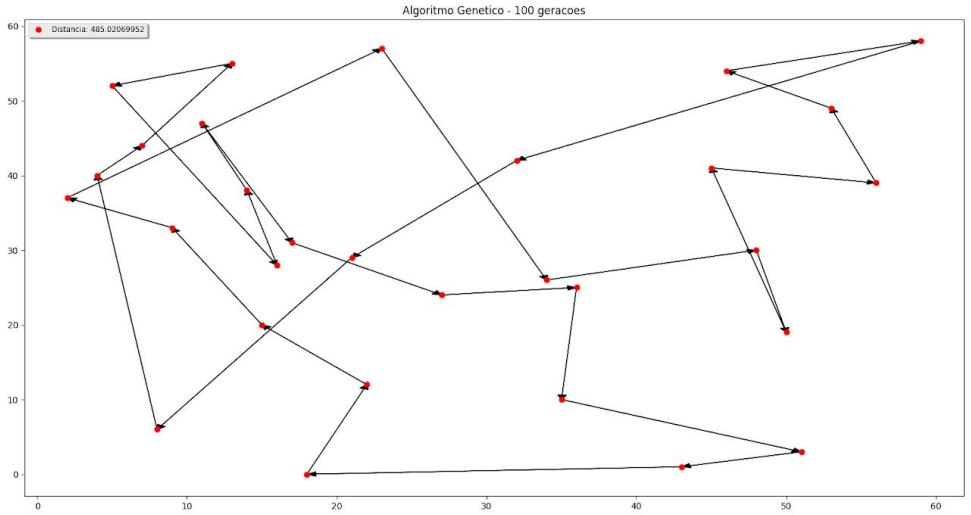

Figura 1: Algoritmo Genético = 100 gerações - *'out1.txt'*

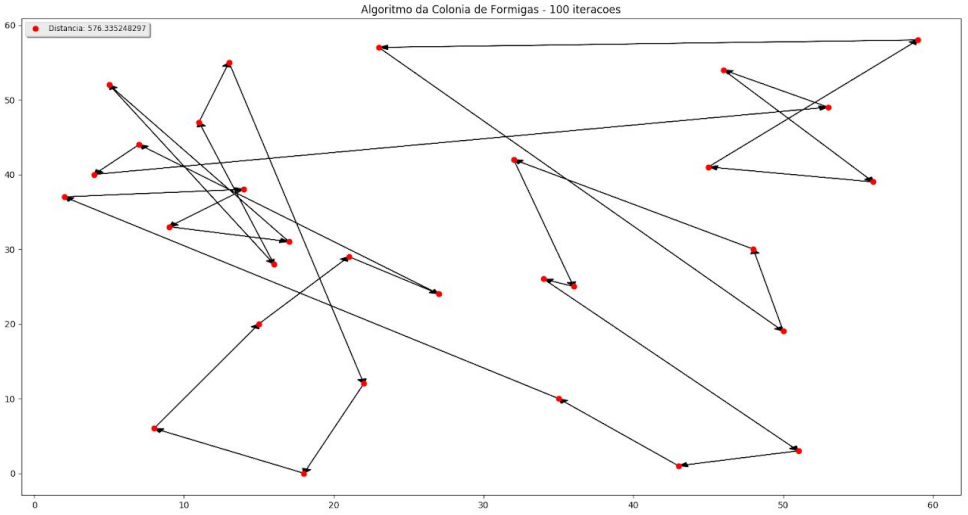

Figura 2: Algoritmo da Colônia de Formigas = 100 gerações - *'out1.txt'*

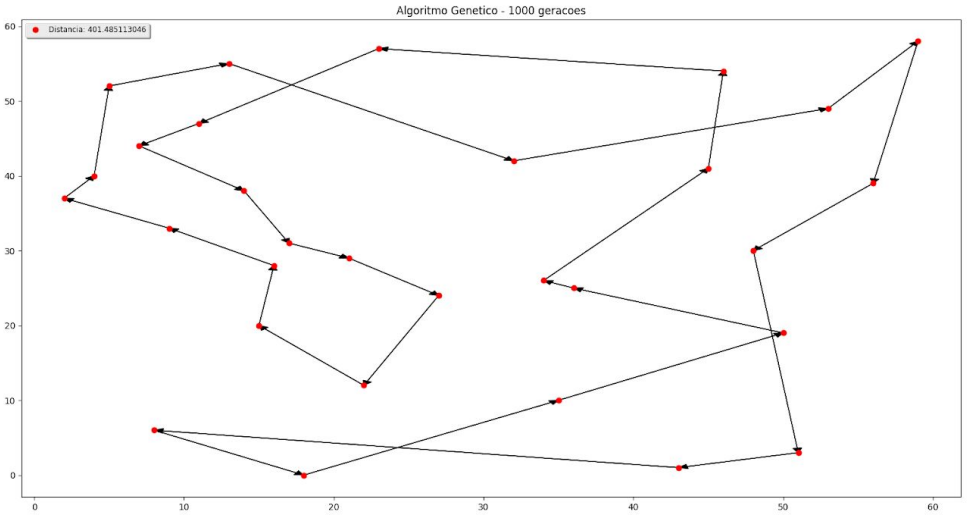

Figura 3: Algoritmo Genético = 1000 gerações - *'out1.txt'*

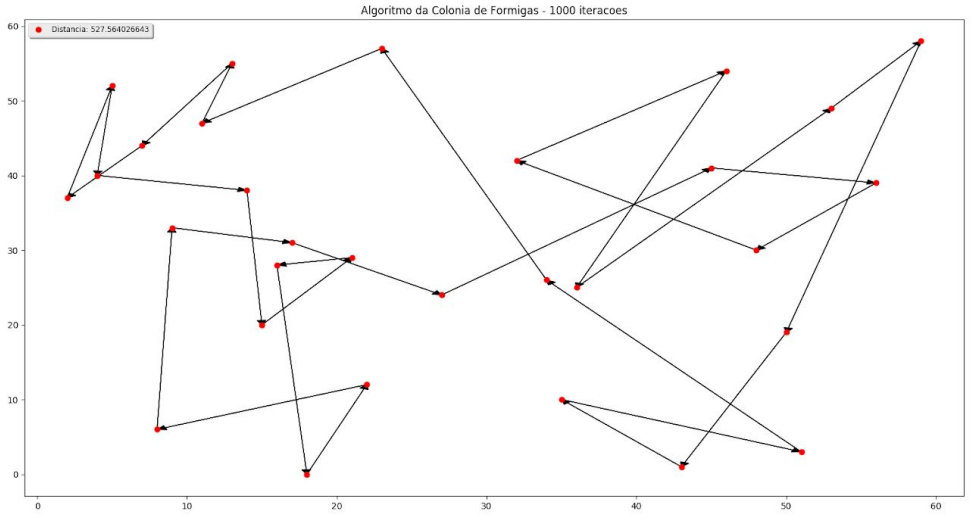

Figura 4: Algoritmo da Colônia de Formigas = 1000 gerações - *'out1.txt'*

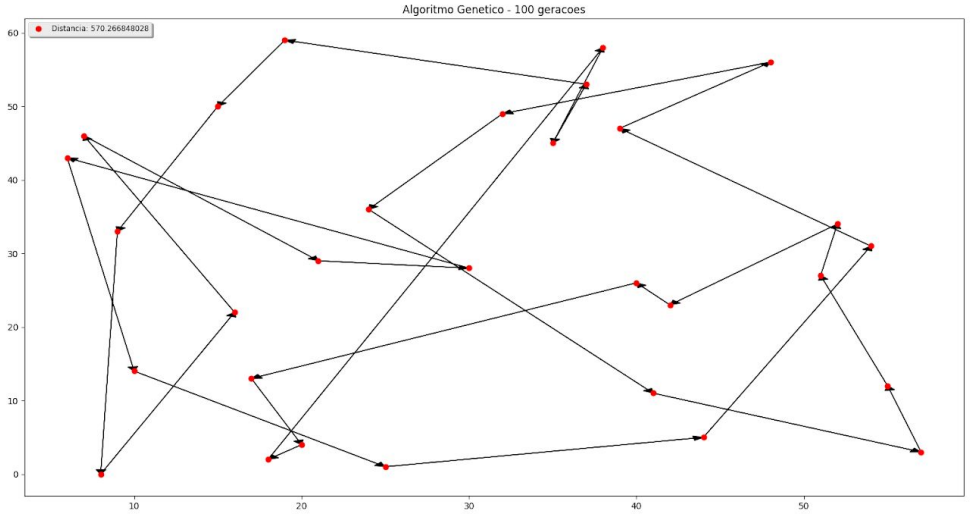

Figura 5: Algoritmo Genético = 100 gerações - *'out2.txt'*

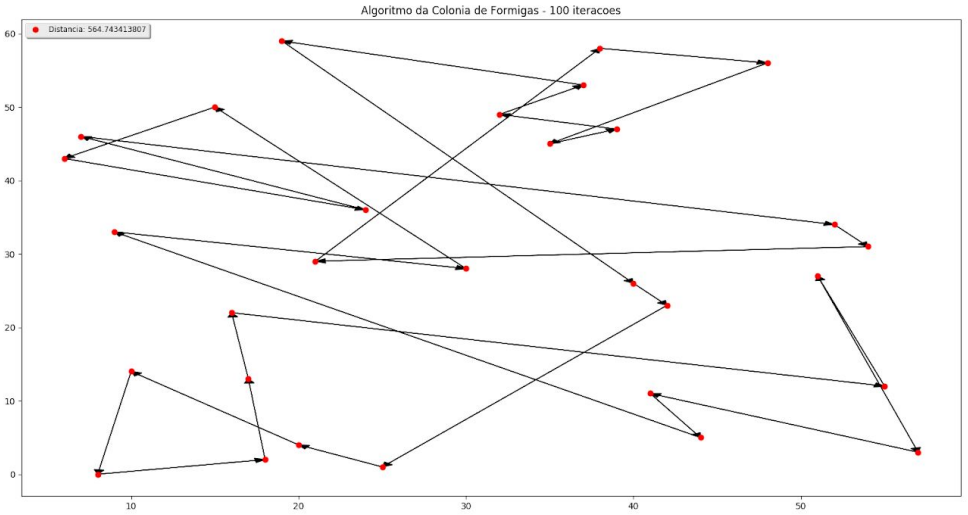

Figura 6: Algoritmo da Colônia de Formigas = 100 gerações - *'out2.txt'*

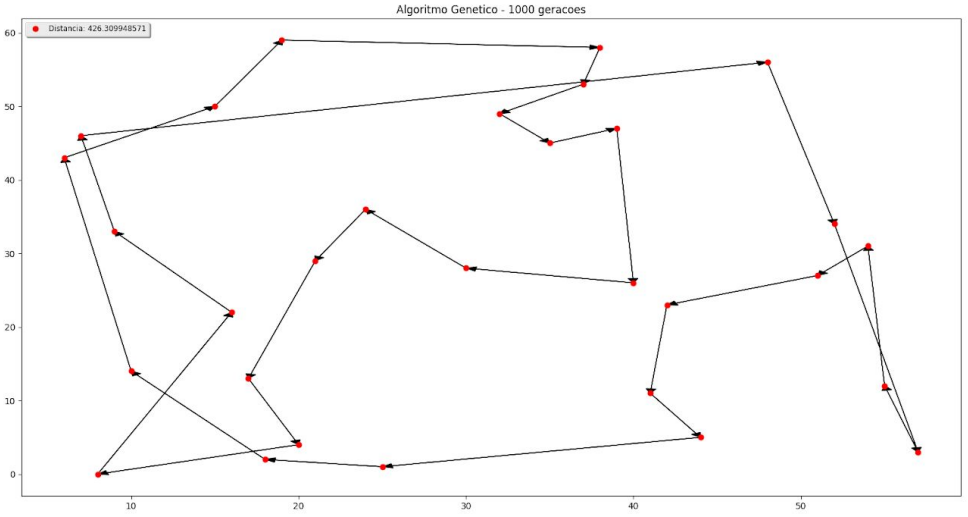

Figura 7: Algoritmo Genético = 1000 gerações - *'out2.txt'*

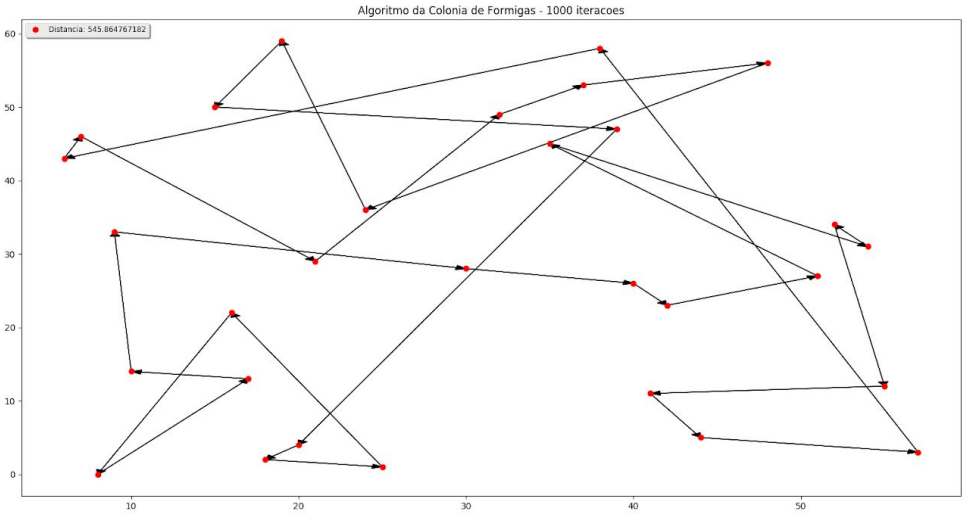

Figura 8: Algoritmo da Colônia de Formigas = 1000 gerações - *'out2.txt'*

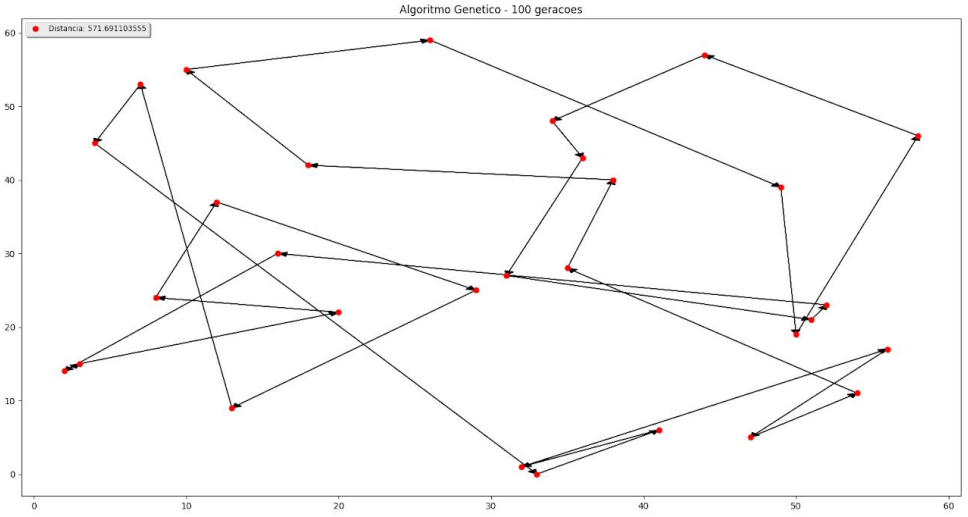

Figura 9: Algoritmo Genético = 100 gerações - *'out3.txt'*

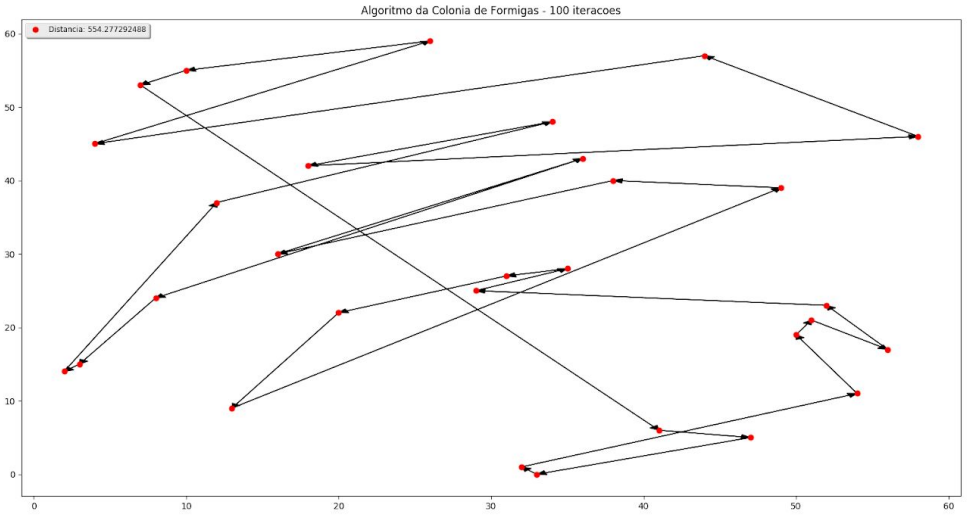

Figura 10: Algoritmo da Colônia de Formigas = 100 gerações - *'out3.txt'*

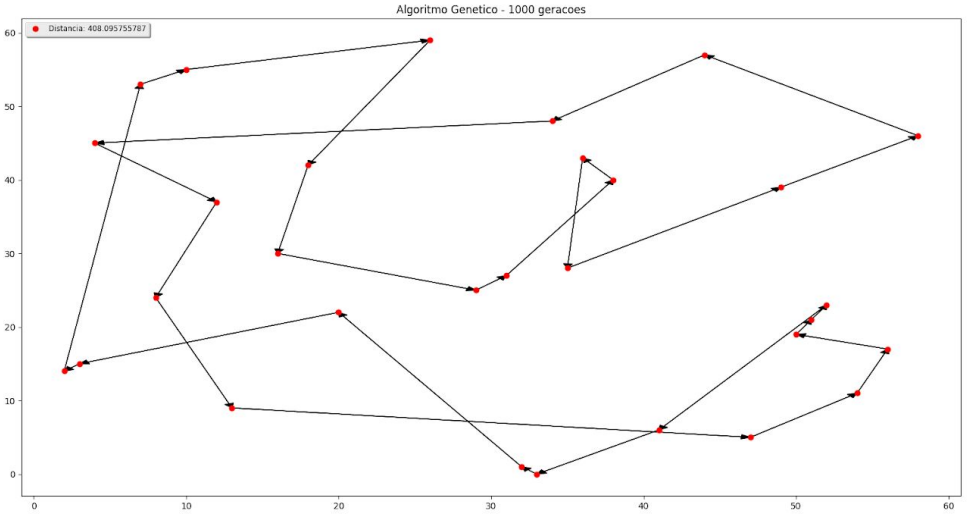

Figura 11: Algoritmo Genético = 1000 gerações - *'out3.txt'*

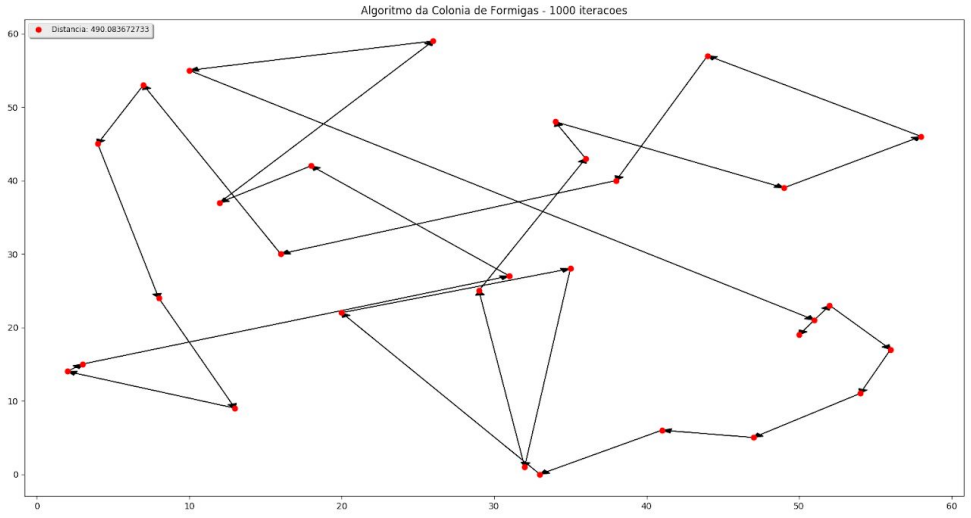

Figura 12: Algoritmo da Colônia de Formigas = 1000 gerações - *'out3.txt'*

5. Considerações Final

DISTÂNCIAS	*'out1.txt'*	*'out2.txt'*	*'out3.txt'*
GA = 100 gerações	485,02	570,26	571,69
ACO = 100 iterações	576,33	564,74	554,27
GA = 1000 gerações	401,48	426,30	408,09
ACO = 1000 iterações	527,56	545,86	490,08

Ao analisar os resultados pode-se ver uma certa vantagem para o algoritmo genético, no entanto, dois pontos devem ser relatados. Primeiramente, em todas as vezes que foi executado o algoritmo da colônia de formigas, ele nunca conseguiu um resultado melhor que o algoritmo genético. Foi alterado parâmetros, procura por bugs no código, mais nada mudou. Acredito que somente com mais testes ou avaliando com diferentes problemas, que se possa chegar a alguma conclusão a respeito da eficácia das técnicas. Em segundo ponto, foi visível a diferença de tempo de execução entre as duas técnicas. A colônia de formigas demorou

tranquilamente algumas dezenas de vezes a mais que o algoritmo genético. Embora não foi utilizado nenhum método de verificação do tempo de execução, mais essa diferença foi notada. Novamente, talvez com ajustes ou reescrita no código, talvez sua eficiência seja melhorada.

6. Referências

[1] https://pt.wikipedia.org/wiki/Problema_do_caixeiro-viajante
[2] **LINDEN**, R. **Algoritmos Genéticos**, 2a edição, Ed. Brasport, Rio de Janeiro, 2008
[3] https://pt.wikipedia.org/wiki/Col%C3%B4nia_de_formigas_(otimiza%C3%A7%C3%A3o)
[4] https://github.com/marcoscastro/tsp_aco