Aplicação de algoritmos de busca na resolução do Problema do Caixeiro Viajante

João Paulo Vargas da Fonseca

Departamento Acadêmico de Engenharia Eletrônica

Universidade Tecnológica Federal do Paraná (UTFPR) – Curitiba, PR -Brazil

jfonseca@alunos.utfpr.edu.br

**Abstract.** This paper tries to implement and compare two local search algorithms, simulated annealing and genetic algorithm, to find a good solution to the travelling salesman problem. A tour problem where the solution is a closed link path between a group of cities, and the goal is to find the shortest distance path.

**Resumo.** Este artigo busca implementar e comparar duas técnicas de busca local, a têmpera simulada e algoritmo genético, para buscar uma boa solução para o problema do caixeiro viajante. Um problema de tour onde a solução é o caminho em enlace fechado de um grupo de cidades, e a meta é encontrar o caminho com o menor percurso.

1. Problema do Caixeiro Viajante

O problema do caixeiro viajante é um problema simples, consistente de encontrar o menor percurso dentre um conjunto de cidades, partindo de uma cidade origem onde cada cidade deve ser visitada apenas uma vez, e após voltar à origem. Porém, ele é complexo computacionalmente, da ordem de O(n!), sendo n o número de cidades. Por isso utilizam-se algoritmos de busca, que têm como objetivo otimizar o custo computacional, como os algoritmos têmpera simulada e o algoritmo genético, para encontrar soluções boas, mas sem otimalidade, que é a garantia de se obter a melhor solução, no caso, obter o caminho com a menor distância possível.

1. Fundamentação Teórica

O algoritmo da têmpera simulada implementa uma busca em estados vizinhos ao do atual estado, sendo uma melhoria ao algoritmo de subida de encosta pois, enquanto este acessa um estado inicial aleatório e implementa uma ida somente a estados vizinhos que tenham alguma característica melhor, que é medida através de uma função, aquele permite a ida a estados piores com uma certa probabilidade que decai com o tempo, ou número de iterações ou temperatura. Sendo esta somente uma forma de inserir aqueles no cálculo de probabilidade, que, quando chega a zero, transforma o algoritmo em uma subida de encosta, parando quando não houver melhores estados vizinhos que o atual.

Já o algoritmo genético tenta implementar algo totalmente diferente, pois se assemelha ao processo de evolução natural para se chegar a um estado melhor que o atual. Ele se utiliza de uma população, ou seja, um número arbitrário de indivíduos (estados inicialmente gerados aleatoriamente), com poucos escolhidos de acordo com seu valor de fitness, uma função avaliativa do próprio estado, para reproduzir, combinando genes dos pais, e gerar a próxima geração, sendo que há uma possibilidade de cada novo indivíduo gerado sofra alguma mutação. E a condição de parada normalmente é uma convergência dos genes da população.

1. Metodologia

O ambiente considerado é um plano, com cidades puntiformes. Desta forma, uma cidade pode ser representada por um número único e uma posição em um plano cartesiano. E um caminho pode ser facilmente representado por um conjunto de números únicos, referentes às cidades na ordem em que devem ser percorridas pelo caixeiro antes de retornar à inicial.

Como é possível ter um número n de cidades, cada ambiente será gerado aleatoriamente com as n cidades e armazenado, até para ter uma referência de comparação entre os algoritmos de têmpera simulada e o algoritmo genético.

Computacionalmente, cada ambiente será uma matriz n x 2, sendo o primeiro índice o número único da cidade e o segundo referente à posição x (1) e y (2), e cada conjunto de cidades será armazenado em um arquivo texto (“.txt”). O caminho armazenado em um vetor de tamanho n, sendo o índice responsável pela posição da cidade no caminho, e seu armazenamento será o número único da cidade. E como a comparação de um caminho a outro é feita pela distância total, que é a soma da distância entre as cidades adjacentes no caminho, será armazenado em uma matriz n x n a distância entre duas cidades i e j, prevendo com que o mesmo cálculo seja feito milhares de vezes pelo custo de mais armazenamento.

A implementação será feita utilizando do Matlab R2021a.

* 1. Têmpera Simulada

No têmpera simulada, vizinhos são considerados como estados cujas cidades adjacentes ou consecutivas no caminho, pois esta é a forma de ter o menor número possível de vizinhos.

O valor de um estado é considerado como a soma das distâncias dentre cidades em seu caminho. E a probabilidade do estado atual ir a um estado de valor maior (pior) é dado por (1), desta forma é mais provável ir a estados menos piores do que muito piores.

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

E como a temperatura está se relacionando com a diferença do valor entre estados, foi definido que ela teria valor inicial como a média da matriz de distâncias entre as cidades. E ao longo das iterações, a temperatura é dada por (2), chegando à temperatura final somente quando chegar ao número máximo de iterações, inicialmente deixada como , mas passível de modificação.

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

* 1. Algoritmo Genético

No algoritmo genético foi determinado que se utilizaria uma população com 1000 indivídos, uma probabilidade de mutação de 1%, sendo que ela consiste de trocar duas cidades adjacentes aleatórias de lugar e, se possível se tornar um estado com valor melhor, medida pela função de fitness, sendo a mesma utilizada na têmpera simulada, a soma das distâncias entre as cidades adjacentes no caminho.

A escolha de pais que cruzarão é feita através de funções densidade de probabilidade (pdf), sendo aplicadas à população quando esta estiver ordenada de forma crescente, permitindo que a nova geração seja diversificada, sem elitismo. E as funções utilizadas para tal são as (3) e (4), sendo o parâmetro o número de indivíduos na população e a posição do indivíduo no conjunto ordenado.

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

É importante enfatizar que o número de pais escolhidos é dado somente pela pdf, então quanto mais plana, menos elitista e mais favorável a um maior número de pais que farão parte da próxima geração. A única condição de histórico ou “elitismo” que há é salvar o melhor da geração para a próxima.

O processo de cruzamento é feito por pmx, ou partially mapped crossover, com pontos fixos em (inteiro) (número de cidades)/3 e duas vezes este valor. Sendo que os elementos do 1º pai que estão dentre estes índices são copiados diretamente ao filho, e os elementos do 2º pai que estão no mesmo intervalo, mas não estão no filho são correlacionados com elementos do 1º pai que estão na mesma posição, encontra-se o elemento correlacionado no 2° pai e o coloca na mesma posição no filho. Caso esta posição esteja ocupada, repete-se a correlação até não ter posição ocupada. E, quando terminado para todos os elementos dentro do intervalo, as posições não ocupadas do filho são diretamente preenchidas pelos elementos do 2º pai.

Quando 95% da população tiver os mesmos genes, o algoritmo é dito como completo e retorna o melhor indivíduo.

1. Resultados e Discussões
   1. Têmpera Simulada

Analisando as figuras 2 e 3 do apêndice A, percebe-se que começa muito agitado o algoritmo, pois a temperatura está alta, logo há muita variação em seu custo, tendendo a se abaixar junto com a temperatura, porém, logo antes já se encontra a melhor solução para aquela execução, como visto pelo gráfico laranja. Mesmo assim, tenta procurar algo melhor, mas pode resultar também em um estado pior, como na figura 4, onde o melhor estado encontrado é o mesmo da figura 3, mas o resultado não. Por isso é melhor ter um estado auxiliar que armazena o melhor estado já encontrado, para tal não ser tratado como um mínimo local, quando na verdade é o global.

Verificando o quanto o algoritmo melhora em relação ao estado inicial, que é aleatório, é obtida a figura 5 do apêndice A para vários números de cidades, mas o resultado é aproximadamente constante se contar que há aleatoriedade também.

* 1. Algoritmo Genético

Testando no mapa de 15 cidades e utilizado o mesmo número de iterações, ou seja, 1000 gerações de uma população de 1000 indivíduos, e com a pdf (3), é obtido resultados indicados pelas figuras 6 e 7 do apêndice A.

O caminho é bem parecido com aquele obtido pela têmpera simulada, mas acaba tendo um valor um pouco pior, até porque o algoritmo não convergiu nestas 1000 gerações. Muito provavelmente causado pela pdf usada tornar um pouco mais de 50% da geração atual como pais e os transpor para a próxima.

Mas quando converge, com valores variando de acordo com a figura 8, os resultados são parecidos com os das figuras 2 ou 6 para n = 15. E a média é um pouco abaixo de 2000, indicando que para valores diferentes de cidades, fica constante a convergência, o que mais ou menos acontece com maiores números de cidades se diminuir a população para 100 ou 200. Indicando também que a convergência no número de gerações obedece quase que uma relação linear com o número de indivíduos na população.

Verificando uma melhoria da população final em relação à inicial, sem limite de gerações, é obtida a figura 11 do apêndice A e, em uma comparação com a figura 5, verifica-se que em termos de população, a melhoria obtida não é tão expressiva quanto o outro algoritmo, e com um número de iterações, em média, dobradas. Porém, se pego o pior da primeira geração para comparar ao invés da média, ambos os algoritmos devem produzir resultados parecidos ou um pouco melhores para o genético pois, como no exemplo de 15 cidades, o resultado obtido não é muito diferente.

1. Conclusão

O algoritmo genético não se saiu tão bem quanto o da têmpera simulada neste trabalho.

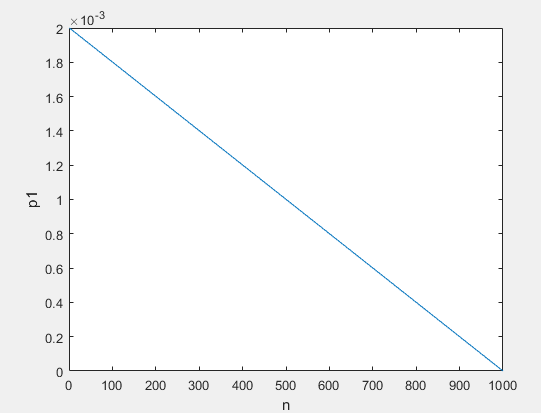
A opção de não ser elitista se provou cara e não tão eficiente se comparada com a têmpera simulada, obtendo resultados parecidos, mas que poderiam ser melhores caso uma outra função densidade de probabilidade fosse empenhada, reduzindo o número de parentes e forçando a população se convergir mais rapidamente. Mas no geral, os melhores indivíduos de cada geração devem se parecer muito com a iteração correspondente do têmpera, além de que ambos são melhores que algoritmos como o guloso, subida de encosta ou técnicas de análise combinatória.

Referências

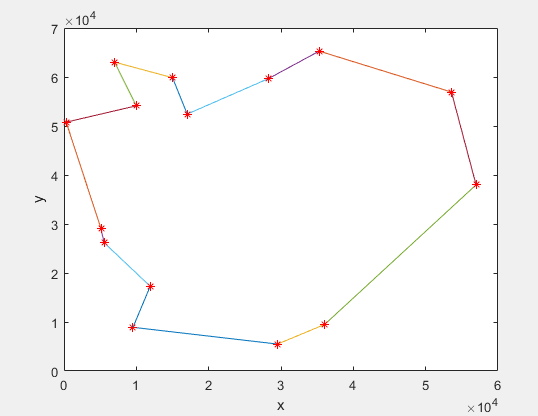
V. Singh and S. Choudhary, "Genetic algorithm for Traveling Salesman Problem: Using modified Partially-Mapped Crossover operator," 2009 International Multimedia, Signal Processing and Communication Technologies, 2009, pp. 20-23, doi: 10.1109/MSPCT.2009.5164164.

Russel, S. e Norvig, P. (2004). Inteligência Artificial. 2ª ed, páginas 69, 109-117. Elsevier Editora Ltda.

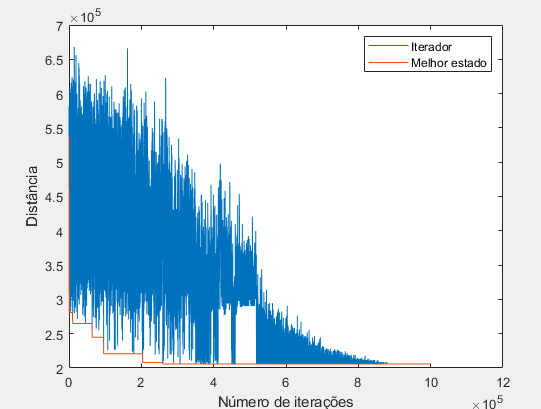
**Apêndice A - Figuras**



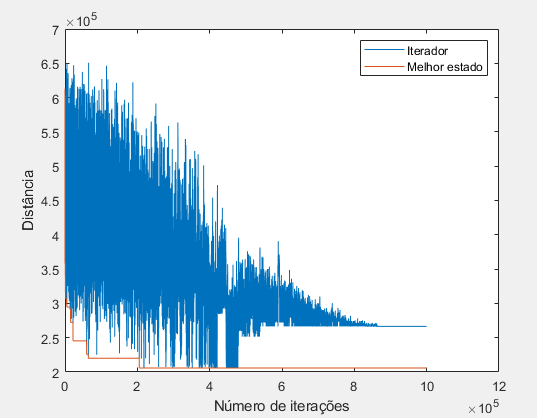
**Figura 1. Gráfico de (3), com N = 1000**



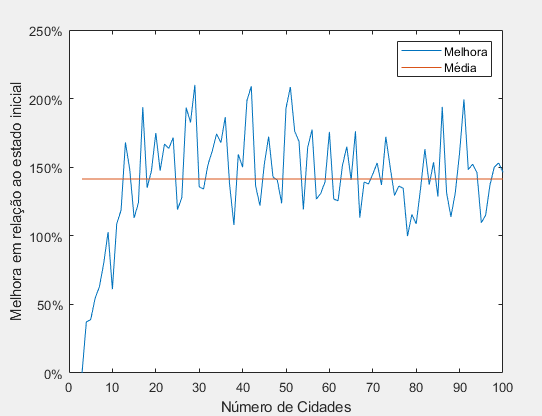
**Figura 2. Caminho obtido pelo algoritmo de Têmpera Simulada com n = 15**

****

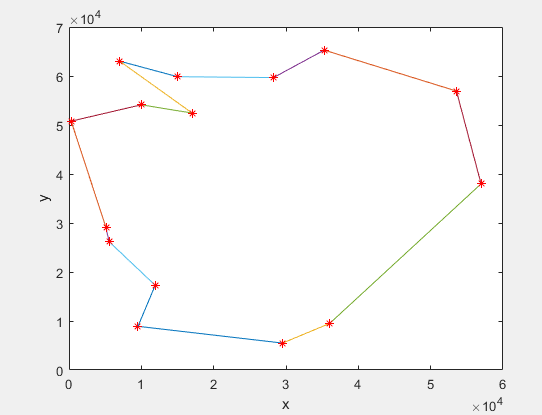
**Figura 3. Histórico de custo do Têmpera Simulada com n = 15**

****

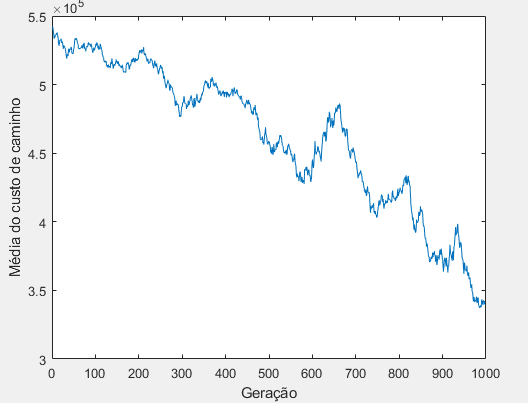
**Figura 4. Histórico de custo do Têmpera Simulada com n = 15**

****

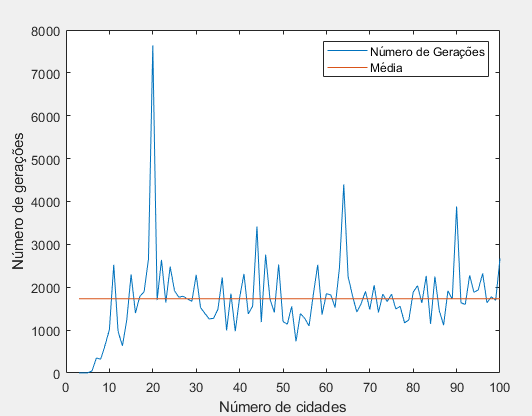
**Figura 5. Porcentagem de melhora no caminho para o Têmpera Simulada**

****

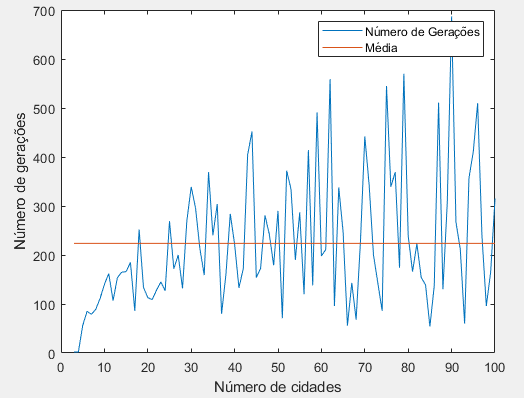
**Figura 6. Caminho obtido pelo Algoritmo Genético com n = 15**

****

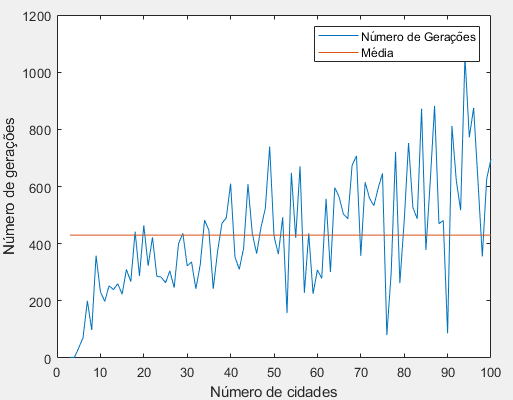
**Figura 7. Média do custo de caminho pelo Algoritmo Genético com n = 15**

****

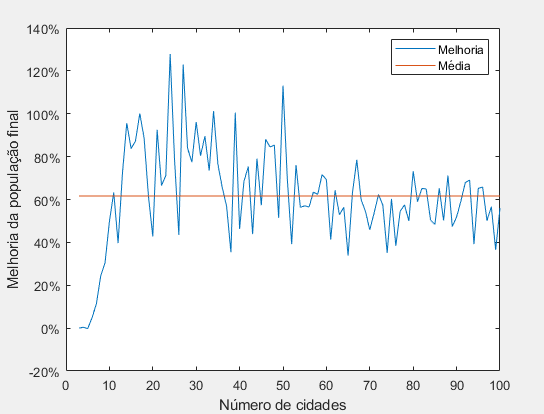
**Figura 8. Número de gerações por número de cidades do Algoritmo Genético para N = 1000**

****

**Figura 9. Número de gerações por número de cidades para N = 100**

****

**Figura 10. Número de gerações por número de cidades para N = 200**

****

**Figura 11. Melhoria da população final em relação à inicial para N = 1000**

**Apêndice B – Links úteis**

O código com a implementação dos algoritmos pode ser encontrado em