Universidade Estadual de Maringá
Departamento de Informática
Professor: Ademir Aparecido Constantino
Disciplina: Modelagem e Otimização Algorítmica - 6903
Aluno: Gabriel T. H. dos Santos. **RA:** 107774
Aluno: Thiago Issao Yasunaka **RA:** 103069

ANÁLISE DA IMPLEMENTAÇÃO DO ALGORITMO GENÉTICO COMBINADO COM BUSCA LOCAL PARA O PROBLEMA DO CAIXEIRO VIAJANTE

Maringá, Paraná
2021

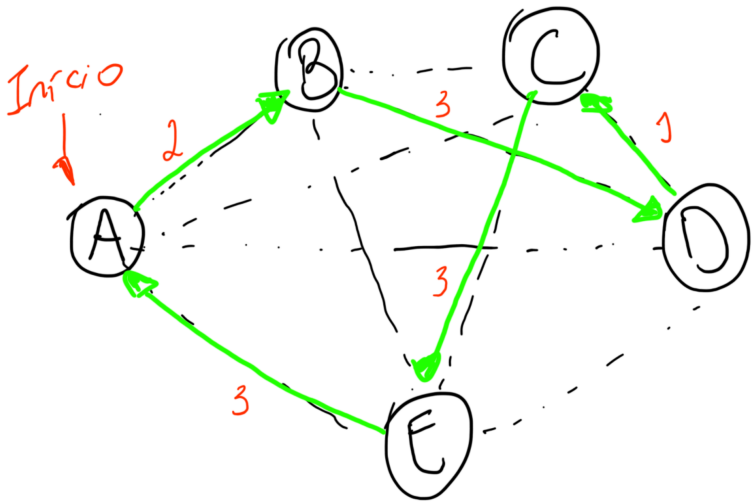
Introdução

Com o avanço da computação que está ocorrendo nas últimas décadas, foram surgindo diversos paradigmas para solucionar problemas computacionais e uma dessas soluções possui como base a teoria da evolução, nomeada de algoritmos genéticos. A análise que será descrita neste trabalho consiste em apresentar resultados referentes ao problema do caixeiro viajante utilizando princípios de algoritmos genéticos com a busca local.

Descrição do Problema

O problema do caixeiro viajante é um problema muito conhecido na área de otimização computacional.
Suponha-se que um viajante escolha uma cidade inicial, seu objetivo será passar por todas as cidades desejadas exatamente uma única vez e retornar para a cidade inicial, percorrendo a menor distância possível. A figura abaixo mostra um exemplo hipotético:

Problema do Caixeiro Viajante:



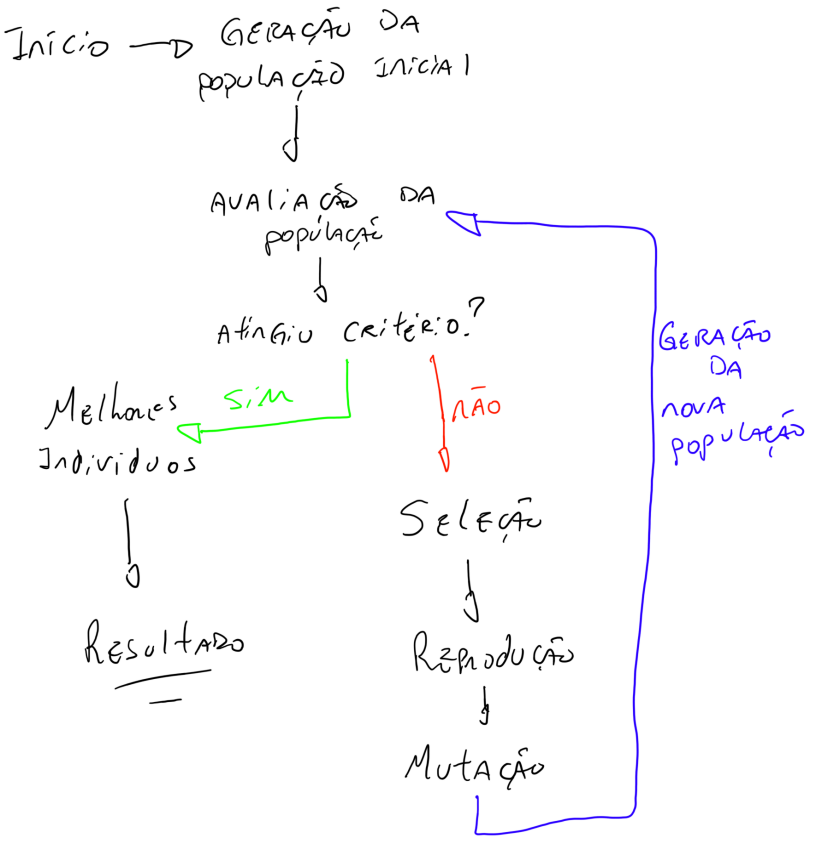
Na figura, é possível observar que o vértice A é o ponto de partida, depois do ponto A, são percorridos os vértices B, D, C, E e finalizando novamente com o ponto A, respectivamente. Supondo que este é o melhor caminho, então a solução para este exemplo seria `dAB + dBD + dDC + dCE + dEA = 12.`
O problema do caixeiro viajante é um problema simples, porém a sua solução possui complexidade fatorial caso sua resolução for de forma exata. Para contornar esta complexidade custosa, uma das soluções seria a implementação do problema do caixeiro viajante utilizando o conceito de algoritmos genéticos, porém utilizando algoritmos genéticos

não é garantido a melhor solução, e sim, a melhor solução encontrada de acordo com diversos parâmetros.

Descrição do Algoritmo

Como o nome sugere, algoritmos genéticos fazem uma analogia com a teoria da evolução, na qual a partir de uma população inicial, é possível gerar descendentes com características semelhantes com a de seus pais, e além disso, ter algumas mutações em seus genes durante a sua criação. A figura abaixo mostra um exemplo do que seria um algoritmo genético genérico:

Algoritmo Genético:



O fluxo acima ocorre da seguinte forma:

- **Início:** Foram instanciadas todas as variáveis que foram utilizadas de forma global, um exemplo é o vetor contendo todas as cidades que foram importadas do arquivo de cidades disponibilizadas, possuindo três critérios: Nome do vértice, coordenada x e coordenada y do plano cartesiano.

- **Geração da população inicial:** Para o problema do caixeiro viajante, foram geradas duas soluções genéricas iniciais. Para gerar de forma aleatória, foi sorteada um

valor entre zero e a quantidade de cidades, a partir disso foram construídas as duas primeiras soluções para a população.

- **Avaliação da População:** A avaliação da população é determinada pela menor distância de uma solução, isto é, o menor caminho encontrado até então pelo algoritmo genético. Para isto, a população é ordenada utilizando o atributo *distance* de uma solução.

- **Atingiu Critério:** O critério utilizado para parar a execução do algoritmo é a quantidade de soluções geradas pelo algoritmo. Este valor irá variar para cada teste realizado.

- **Seleção:** Seleciona as duas melhores soluções obtidas até então, ou seja, as duas soluções com as menores distâncias encontradas. De acordo com a documentação disponibilizada, este tipo de seleção é denominada **seleção por classificação,** pois a população é ordenada de acordo com o seu valor de aptidão, isto é, a distância percorrida pelo caixeiro viajante.

- **Mutação:** Após o *crossover,* cada novo indivíduo gerado passa por um processo de mutação, isto é, alguns vértices do grafo são trocados, nesta etapa a distância é definida como zero para todas as novas instâncias, pois ela será calculada apenas na avaliação da população. A quantidade de vértices que serão alteradas é definida na execução do algoritmo.

- **Geração da Nova População:** Na geração da nova população ocorre a inserção das novas soluções encontradas a partir do *crossover* e mutação, ou seja, a cada iteração são inseridas dois novos descendentes para a população.

- **Melhores Indivíduos:** Os melhores indivíduos são selecionados a partir de uma ordenação ascendente observando a distância percorrida pelo caixeiro viajante.

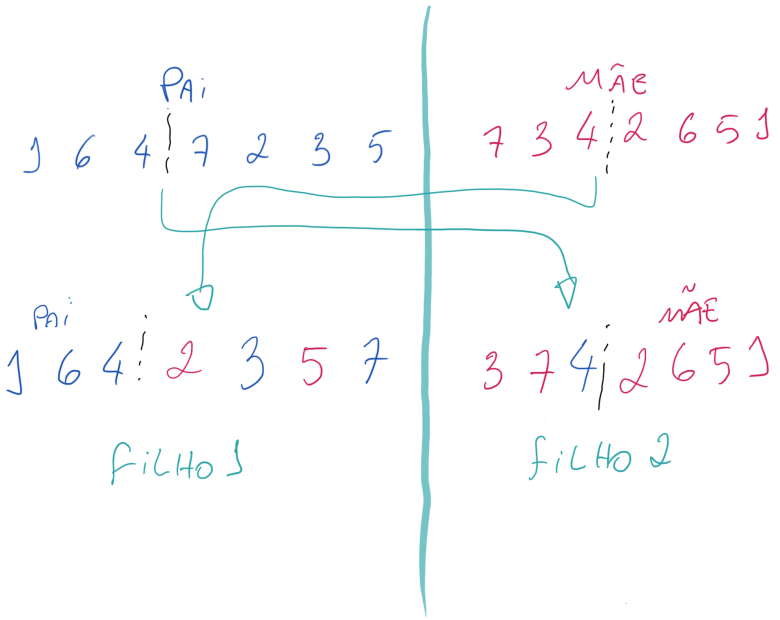
- **Resultados:** O resultado do algoritmo apresenta a menor distância encontrada, junto com o caminho encontrado e o tempo de execução do algoritmo.

- **Reprodução ou *Crossover*:** Logo após a seleção, é possível fazer a reprodução de indivíduos filhos dos pais selecionados. A técnica utilizada é descrita abaixo:
    - A seleção disponibiliza o que será o pai e a mãe dos filhos gerados na reprodução.
    - Divide-se pela metade o caminho (solução) do pai e o caminho (solução) da mãe.
    - A primeira metade do caminho do filho 1 será idêntica à primeira metade do caminho do pai.
    - A segunda metade do caminho do filho 2 será idêntica à segunda metade do caminho da mãe.
    - Para o filho 1, o *crossover* ocorre na segunda metade do seu caminho, ou seja, ele recebe todos os genes da segunda metade do caminho da mãe que **não seja repetida** com os genes já inseridos (isto se deve a regra do caixeiro viajante). Se um gene for repetido, será inserido o primeiro gene

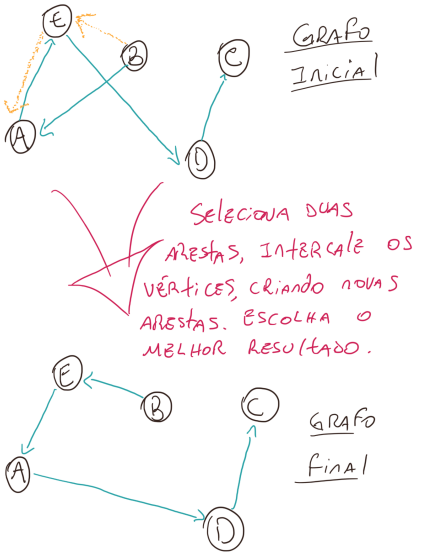
encontrado do pai que ainda não foi inserido no caminho. A mesma lógica é feita para o filho 2.

A figura abaixo explicar o *crossover* de forma visual:

Representação do *crossover* para o problema do caixeiro viajante:



Além de todo o fluxo exibido acima, existe uma outra etapa que tende a melhorar o resultado para o caixeiro viajante. A busca local baseada no algoritmo 2-opt é uma das várias opções disponíveis na literatura para otimizar o algoritmo genético para o caixeiro viajante. Uma figura para exemplificar o funcionamento do algoritmo 2-opt com a busca local é mostrado abaixo:



A busca local escolhe aleatoriamente uma aresta do grafo e a partir dela ela seleciona uma outra aresta, com isso ocorre uma intercalação (mostrada na figura acima) entre as arestas a fim de apresentar melhores resultados, no caso do caixeiro viajante um melhor resultado é representado por uma distância percorrida menor. Após esta intercalação ocorre uma comparação de resultados, se o novo resultado obtido for melhor do que a anterior, então os vértices recebem novos vizinhos, isto é, ocorre a troca entre as arestas.

Especificações Técnicas

Para os testes realizados, foi utilizado uma máquina com as seguintes configurações:

- Processador: Intel® Core™ i7-1165G7 (2.8 GHz até 4.7 GHz, cache de 12MB, quad-core, 11ª geração)
- Memória: 16GB (2x8GB), DDR4, 3200MHz
- Gráfico: Mesa Intel® Xe Graphics (TGL GT2)
- Sistema Operacional: Ubuntu 20.10, 64 bits.

Resultados e Conclusões

Foram realizados diversos testes a fim de mostrar uma melhoria na distância percorrida pelo caixeiro viajante conforme fosse sendo aumentado a quantidade de descendentes gerados. O Gráfico a seguir tenta justificar isso com valores:

- **Caso analisado:** kroA100
- **Taxa de mutação:** 1%

Tabela da melhor distância encontrada x Tamanho da população (SEM BUSCA LOCAL)

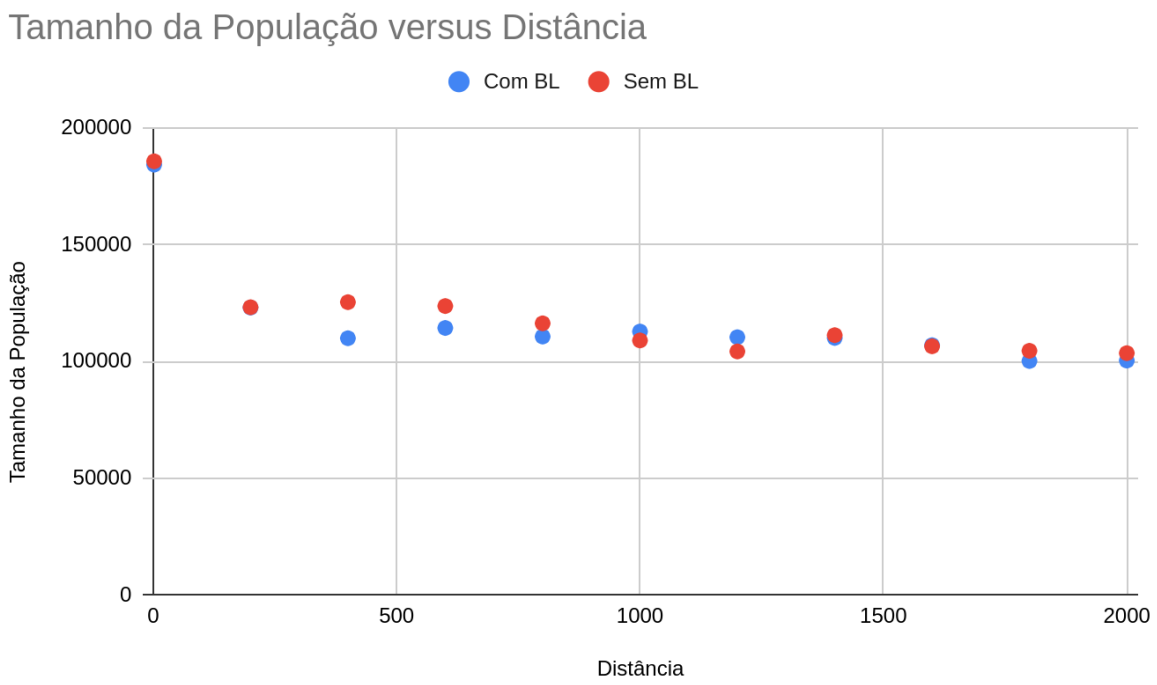
Distância	Tamanho da População
185580	2
123185	200
125350	400
123696	600
116295	800
108988	1000
104284	1200
111244	1400
106484	1600
104566	1800
103553	2000

Tabela da melhor distância encontrada x Tamanho da população (COM BUSCA LOCAL)

Caso	Mutation %
kroA100	1
Distância	**Tamanho da População**
184148	2
123036	200
109930	400
114326	600
110649	800
112800	1000
110362	1200
110060	1400
106957	1600
100198	1800
100387	2000

A partir das tabelas com os valores encontrados durante os testes, é possível gerar o seguinte gráfico:

Gráfico para o caso kroA100 - Tamanho da População x Distância



Analisando o gráfico acima, é possível validar a melhora nos resultados obtidos conforme é aumentado o critério de parada (tamanho da população). Porém, se comparado com a

melhor solução obtida pela literatura (O melhor resultado encontrado segundo o documento disponibilizado foi de 21282) ainda há uma diferença significativa que dá para ser bastante otimizada, pois o melhor resultado obtido foi de 103553. A partir destes dados, o GAP entre a melhor solução encontrada com a melhor solução da literatura, isto é, o desvio relativo, será maior do que 350%.
Foram testados dois tipos de situações, utilizando a busca local e não utilizando a busca local. Com a busca local o resultado obtido, de modo geral, é melhor do que sem a busca local, porém para este caso de teste, o resultado melhorado não obteve um resultado tão acima do esperado.
Por que um GAP tão alto ocorreu? Existem alguns fatores que podem influenciar muito um resultado de um algoritmo genético, sendo eles:

- Durante o crossover, quando um novo descendente está sendo criado, existem infinitas formas para se fazer o cruzamento entre os genes do pai e os genes da mãe, dessa forma, uma determinada heurística escolhida para fazer esta ação pode acabar impactando muito o resultado final da operação.
- A taxa de mutação pré-determinada pode impactar diretamente o resultado final da otimização do problema do caixeiro viajante, pois seus genes são alterados baseado nesta taxa.
- Critério de Parada. No caso desta base de testes, o critério de parada foi o tamanho da população. Observando o gŕafico acima, é possível observar que conforme é aumentado a distância, a tendência de uma melhora nos resultados é explícita. Baseado nisso, o critério de parada possui uma influência grande na obtenção de melhores resultados.

Além do caso detalhado testado no parágrafo acima, foram realizados diversos outros testes para diversos outros casos, porém estes menos aprofundados. A tabela a seguir mostra as soluções encontradas para a execução de diversos tipos de conjuntos.

Caso	**Ms**	**Alg**	**Gap_1 (%)**	**Alg_BL**	**Gap_2 (%)**	**Tam_Pop**	**Tx_Mut (%)**
a280	2579	2850,61	10,53	2841,24	10,17	1000	1,43
gr666	294358	5504,8	-	5424,41	-	1000	0,90
kroA100	21282	104566	391,34	100198	370,81	1800	1
pla33810	66048945	-	-	-	-	-	-
pla85900	142382641	-	-	-	-	-	-
pr1002	259045	339879	31,20	337397	30,25	1000	1,00
tsp225	3916	10010	155,62	9876,06	152,20	1000	0,89

De acordo com os dados obtidos na tabela acima, é possível concluir que para alguns casos os resultados foram próximos do melhor valor já encontrado na literatura, estes casos foram a280 e pr1002.
Para os casos kroA100 (Este caso possui uma análise mais detalhada no tópico acima) e tsp225 os resultados obtidos possuíram um GAP acima de 100%, um valor bem distante dos melhores resultados.

Um caso indeterminado foi para o gr666, o resultado foi muito melhor do que o esperado, portanto, para este caso específico, fica a necessidade de uma revisão com a literatura.
Para os casos pla33810 e pla85900 não foi possível obter um resultado, o computador que realizou a execução para estes casos de testes obteve um estouro de memória .
A busca local implementada no projeto melhorou os resultados, porém estes resultados não foram extremamente relevantes, as melhoras ficaram em torno de 1% para cada caso testado.

Legenda:

- **Caso:** nome do caso testado.
- **MS:** melhor solução conhecida da literatura.
- **Alg:** melhor solução obtida com o algoritmo.
- **Alg_BL:** melhor solução obtida com o algoritmo com a busca local.
- **Gap_1%:** parâmetro de desvio relativo para Alg. (Gap_1% = [(Alg - MS) / MS] * 100.
- **Gap_2%:** parâmetro de desvio padrão para Alg_BL. (Cálculo igual ao Gap_1% porém para o Alg_BL).
- **T_Alg (sec):** tempo de execução para o Alg, em segundos.
- **T_Alg_BL (sec):** tempo de execução para o Alg_BL, em segundos.

Referências

CASTILHO, Marcos. **Problema do caixeiro viajante**. Curitiba, 2012. 11 slides, color. Disponível em: https://www.inf.ufpr.br/marcos/ci242/aula4.pdf. Acesso em: 08 maio 2021.

WILHELM, Volmir Eugênio. **Problema do Caixeiro Viajante**. Curitiba, 2020. 17 slides, color. Disponível em: https://docs.ufpr.br/~volmir/PO_II_12_TSP.pdf. Acesso em: 08 maio 2021.

CONSTANTINO, Ademir Aparecido. **Algoritmos Genéticos + busca local**. Maringá, 2021. 3 slides, color. Disponível em: https://moodlep.uem.br/pluginfile.php/171013/mod_resource/content/1/AG-BL.pdf. Acesso em: 08 maio 2021.

CONSTANTINO, Ademir Aparecido. **Heurísticas Para Otimização Combinatória**. Maringá, 2021. 66 slides, color. Disponível em: https://moodlep.uem.br/pluginfile.php/171008/mod_resource/content/2/Aula2_Introducao_Otimizacao_Comb%20-%203%20.pdf. Acesso em: 08 maio 2021.

CONSTANTINO, Ademir Aparecido. **Algoritmos Genéticos Aplicados à Otimização.** Maringá, 2021. 29 slides, color. Disponível em: https://moodlep.uem.br/pluginfile.php/171012/mod_resource/content/2/AG-TPO_3.pdf. Acesso em: 08 maio 2021.

CARO, Gianni A. di. **local-search-and-ILS**. Qatar, 2021. 21 slides, color. Disponível em: https://web2.qatar.cmu.edu/~gdicaro/15382/additional/local-search-and-ILS.pdf. Acesso em: 15 maio 2021.

OLIVEIRA, José Fernando. **Heuristics and Local Search**. Porto, 2009. Color. Disponível em: https://paginas.fe.up.pt/~mac/ensino/docs/OR/CombinatorialOptimizationHeuristicsLocalSearch.pdf. Acesso em: 15 maio 2021.