**Implementação de grafo bidirecional em C, e uso da estrutura em algoritmos de menor caminho**

## Nascimento, D.T. Prado, G. Rocha, S. Terceiro, P.

## Ciência da Computação – Instituto de Ensino Superior de Brasília (IESB)

## Brasília, Brasil

1. **Apresentação Shortest-Path Problem (SPP)**

O problema do menor caminho é bem antigo. Por hora é isso.

* 1. **Motivação para busca de solução**

Temos muitas razões para buscar a solução deste problema. Logisticamente falando, uma diminuição do tamanho das rotas de entrega de um serviço, por exemplo, pode proporcionar uma enorme economia para uma empresa. Dar outros exemplos.

1. **Apresentação Algoritmo de Dijkstra**
   1. **Explicação base BFS**

O Breadth First Search é um algoritmo que sistematicamente explora cada vértice *v* de um grafo, partindo de um vértice inicial *s* buscando identificar todos os que sejam alcançáveis a partir de *s*. Ao varrer todos os vértices o algoritmo cria uma *Breadth First Tree*, que elenca todo v alcançável e sua respectiva distância da origem. Nessa árvore o caminho é necessariamente o menor caminho possível entre s e v.

* + 1. **Motivação para implementação**

É basicamente para resolver o SPP. Ver se vale a pena manter essa parte.

* + 1. **Implementação**

O BFS assume necessariamente que os pesos entre os vértices são todos iguais, ou seja: ele é um algoritmo que conta o menor número de saltos entre um vértice e outro.Uma das características do BFS é que, como o nome sugere, ele vai seguindo sempre a fronteira dos vértices. Isso quer dizer que ele encontra todos os vértices a P passos da origem antes de encontrar qualquer vértice a P+1 passos da origem.

O funcionamento do algoritmo se dá na seguinte forma:

Antes de iniciar suas iterações ele cria uma representação do grafo com atributos adicionais em cada vértice: cor, anterior e distância. A cor pode ser branca, cinza ou preta, representando respectivamente um vértice que ainda não foi visitado, cinza para um vértice que já foi visitado mas ainda não teve todos os seus consequentes explorados, e preto para um vértice que já teve todos os seus consequentes visitados.

Em seguida, ele procede a iterar sobre cada vértice com o auxílio de uma fila.

|  |
| --- |
| para todo vértice v != s:  v.cor = 'branco';  v.anterior = NULL;  v.distancia = NULL; s.cor = 'cinza'; s.distancia = 0; s.anterior = NULL; fila F; F.push(s); enquanto !F.empty:  atual = F.pop()  para todo v adjacente a atual:  se v.cor == 'branco':  v.cor = 'cinza';  v.anterior = atual;  v.distancia = atual.distancia + 1;  F.push(v);  v.cor = 'preto'; |

* 1. **Explicação base Dijkstra**
     1. **Explicação diferença Dijkstra vs. BFS**

O Algoritmo de Dijkstra pode ser considerado uma evolução do BFS. Enquanto todos os vértices de um grafo para o BFS devem ter o mesmo peso, Dijkstra permite pesos distintos não-nulos.

* + 1. **Implementação Dijkstra**

O algoritmo começa instanciando algumas estruturas auxiliares, a começar por um conjunto S de vértices para os quais os valores de menor caminho partindo de *s* já foram pré-determinados. O algoritmo repetidamente seleciona *v* pertencente ao conjunto de vértices do grafo, V, e não pertencente a S, que tem o menor valor de *shortest path*, adiciona *v*  a *S*, e relaxa todos os vértices saindo de *v*.

|  |
| --- |
| inicializa BFS(); conjunto S;  fila F; enquanto F.empty() != 0:  atual = F.pop\_min();  S.adiciona(atual)  para v adjacente a atual:  relaxa(atual, v, peso(atual,v)) |

relaxa(atual,v,peso(atual,v)), nesse contexto, é atualizar todos os valores de pesos dos vértices adjacentes a v, caso haja diminuição do valor total da distância à *s* passando por atual.

1. **Apresentação Algoritmo Genético**
   1. **Explicação Base do Algoritmo**

O algoritmo genético se baseia na Teoria da Evolução Biológica e se utiliza da premissa de que ao interpolar diferentes soluções propostas a um problema, selecionando os melhores atributos de cada uma delas a cada iteração, o algoritmo converge a uma solução ótima. Isso é feito mediante os conceitos de variabilidade, seleção, cruzamento e mutação.

* 1. **Casos de uso**

O algoritmo genético é utilizado em diversos setores, sendo assim, é vasto o número de aplicações que o mesmo pode oferecer. Entre os principais casos pode-se ressaltar a sua utilização em: análise de DNA, problemas de otimização, processamento de imagens, machine learning, e o TSP (Traveling Salesman Problem) que se assemelha com o algoritmo descrito nesta análise.

* 1. **Implementação**

O algoritmo define alguns termos acessórios para a sua execução. Como não é um algoritmo concebido especificamente para resolver o SPP, devemos fazer algumas adequações. No algoritmo, gene é definido como cada atributo de uma solução proposta. O conjunto de genes que compõe uma solução é denominado cromossomo, que seguindo o conceito de caminhos, deve ter o número máximo de vértices do grafo como quantidade máxima de genes. Cada solução composta, que é definida por um cromossomo, é chamada de indivíduo, e o conjunto de indivíduos propostos é chamado de população. Além disso, é definida uma *fitness function*, uma métrica para verificar quais dos indivíduos melhor solucionam o problema.

O algoritmo inicialmente cria N indivíduos viáveis, seguindo as regras de negócio. No nosso caso, gera caminhos aleatórios, não-cíclicos e não retrógrados, levando de *s* a *v.* Essa população, em seguida, é submetida aos processos de seleção baseada no *fitness*, cruzamento e mutação. É importante ressaltar que o modelo do algoritmo proposto apresenta tamanho fixo de população e que ocorre uma taxa de sobreposição de gerações denominada Taxa de Elitismo que é explicada mais à frente.

A seleção consiste na transformação do *fitness* individual em uma porcentagem do geral da população de forma que a probabilidade de seleção de progenitores para cruzamento ocorre seguindo essa porcentagem. Dessa forma que os indivíduos com maior valor de *fitness* acabam possuindo maior chance de deixar seus genes para as gerações seguintes.

Após a fase de seleção ocorre a fase de cruzamento entre dois progenitores, onde são identificados os genes comuns aos indivíduos e, aleatoriamente, escolhido um desses genes como linha de corte para que ocorra o Crossing Over, sendo esse o processo de permuta gênica em que se baseia o cruzamento. Sendo assim, é realizada a troca das porções entre os cromossomos nessa linha de corte, gerando duas proles que são submetidas a uma função de correção para que seja retirado, caso haja, loops.

A mutação é uma alteração que ocorre em uma posição aleatória do genoma do indivíduo e se torna extremamente necessária no contexto para a manutenção da variabilidade genética da população, trabalhando de forma contrária ao fenômeno de Deriva Genética, que pode ser adequado ao conceito da perda de genes ao longo das gerações . Para minimizar o efeito da Deriva Genética foram considerados dois casos de mutação:

Mutação na prole: Ao ocorrer o cruzamento, as duas proles geradas se tornam suscetíveis ao fenômeno de mutação.

Mutação no indivíduo: Foi incorporado um percentual de indivíduos que sofrem mutação ao final de uma geração.

Apesar de a mutação ocorrer aleatória e pontualmente no contexto natural, para que as regras de negócio fossem conservadas o processo de mutação altera uma sequência de genes, sendo um ponto do cromossomo aleatoriamente escolhido para a mudança. Definido o ponto de mutação, é feita a inicialização de um novo caminho até o destino determinado tendo o ponto de mutação como origem.

Ao final do processo de cruzamento é feita a comparação entre as duas proles geradas, sendo selecionada a com maior valor de *fitness* para ocupar uma posição dentro da próxima geração.

Afim de serem preservados os indivíduos com maior *fitness* foi incorporada uma taxa de Elitismo, que faz com que uma quantidade pré-determinada dos melhores indivíduos permaneça na geração seguinte.

Assim, a nova população P(g) definida por uma geração g é definida pela substituição dos indivíduos menos qualificados de P(g-1) quando comparados com os gerados no processo de cruzamento.

* + 1. **Adequação dos termos para o TSP**

Para nosso fim, os genes de cada cromossomo serão os vértices do grafo. Cada cromossomo e, portanto cada indivíduo, será um caminho proposto entre *s* e o destino *t*. A população em sua geração é, assim, um conjunto de caminhos entre *s* e *t* gerados descriteriosamente.

1. **Eficiência Computacional**
   1. **Motivação para cálculo da eficiência**

O cálculo da eficiência computacional é fundamental na escolha do melhor algoritmo para resolução de um problema. A depender da natureza e quantidade dos dados, dois algoritmos possíveis podem alterar entre muito e pouco eficientes. Isso é importante tanto no meio acadêmico quanto no mercado, onde a otimização de recursos computacionais é sempre um dos principais objetivos para minimizar custos. No meio comercial, ainda, esse tipo de análise é essencial para aumentar a competitividade de uma solução e atender os SLAs propostos.

* 1. **Demonstração da eficiência**
     1. **Dijkstra**
     2. **Genético**

Para realizar a avaliação do Algoritmo Genético proposto e sua maior eficácia foram realizadas diversas execuções com parâmetros pré-definidos, sendo a análise destes feitas por meio da avaliação, ao longo das gerações, da média do peso dos indivíduos da população (Azul), média do peso do indivíduo com menor peso da população (Laranja) e média do peso do indivíduo com maior peso da população na geração avaliada (Cinza).

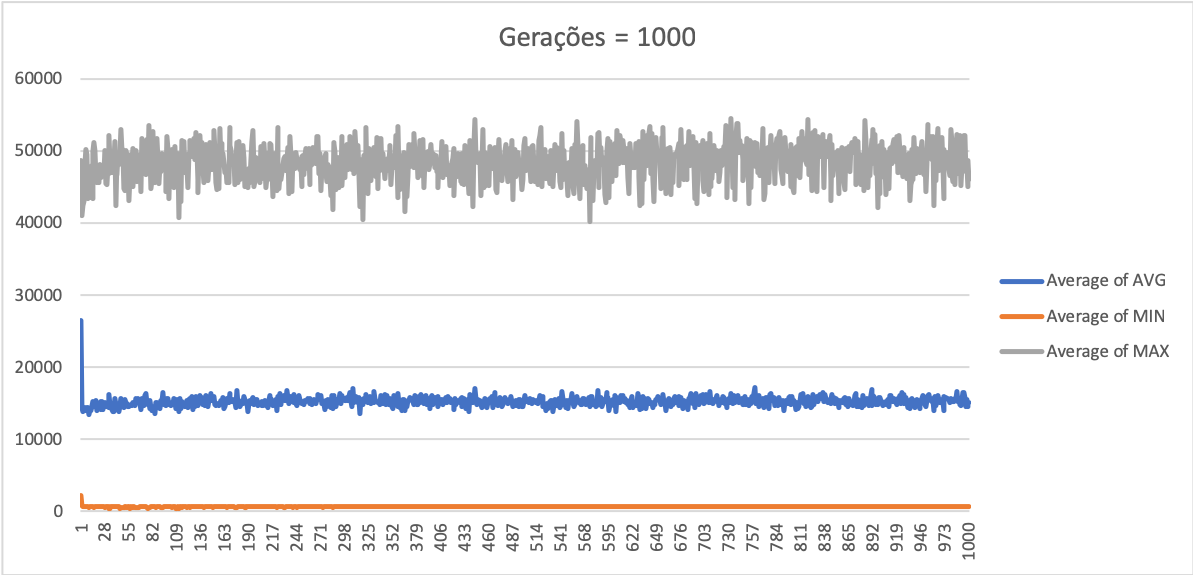
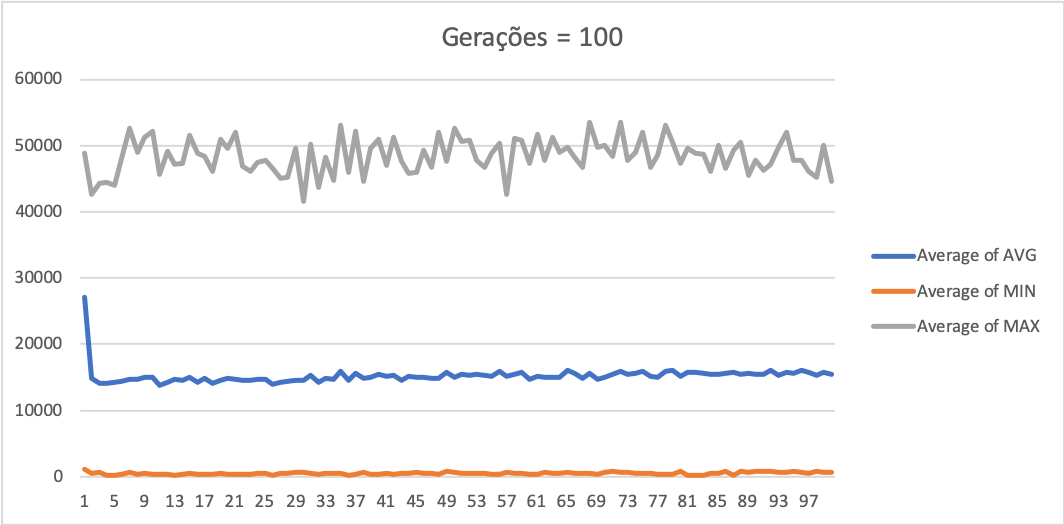
Embora o *fitness* seja o parâmetro utilizado para avaliação da adaptação do indivíduo na população a análise dos somatórios dos pesos entre as ligações que formam os caminhos é igualmente válida para serem feitas análises de qualidade, visto que o peso de um caminho e o peso total da população são utilizados para realizar o cálculo do *fitness*. O único cuidado a ser tomado é que, embora o *fitness* seja diretamente proporcional à adequação do indivíduo na população, o peso se torna indiretamente proporcional. Sendo assim, indivíduos com menor peso são considerados melhores adaptado ou mais adequados.

Foram avaliadas dois valores para cada parâmetros em questão, como mostrado a seguir, sendo os testes realizados e analisados com mesma quantidade de execuções e os distribuídos entre três grafos de 1000 vértices, porém com densidade de conexões distintas:

* Quantidade de Gerações: 100 e 1000;
* Tamanho da População: 20 e 80;
* Taxa de Mutação nos Cruzamentos: 1% e 10%
* Taxa de Mutação nos Indivíduos Consolidados: 1% e 15%
* Taxa de Elitismo: 1% e 10%.
  + - 1. **Quantidade de Gerações**

Não houve grande diferença entre as curvas geradas quando avaliado a quantidade de gerações nos valores determinados, porém é possível a identificação de um ponto de estabilização da curva para quantidade de gerações próximas de 8.

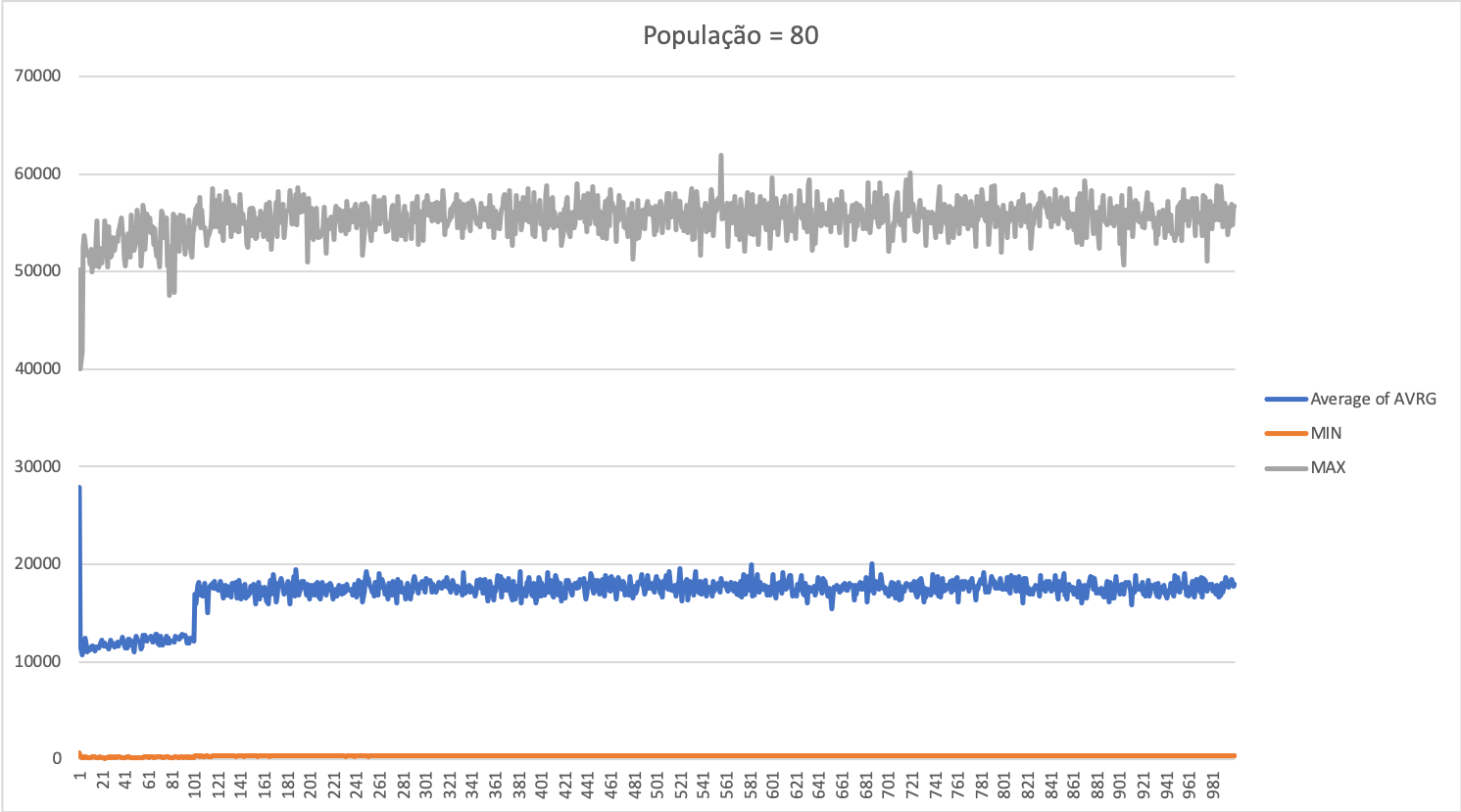
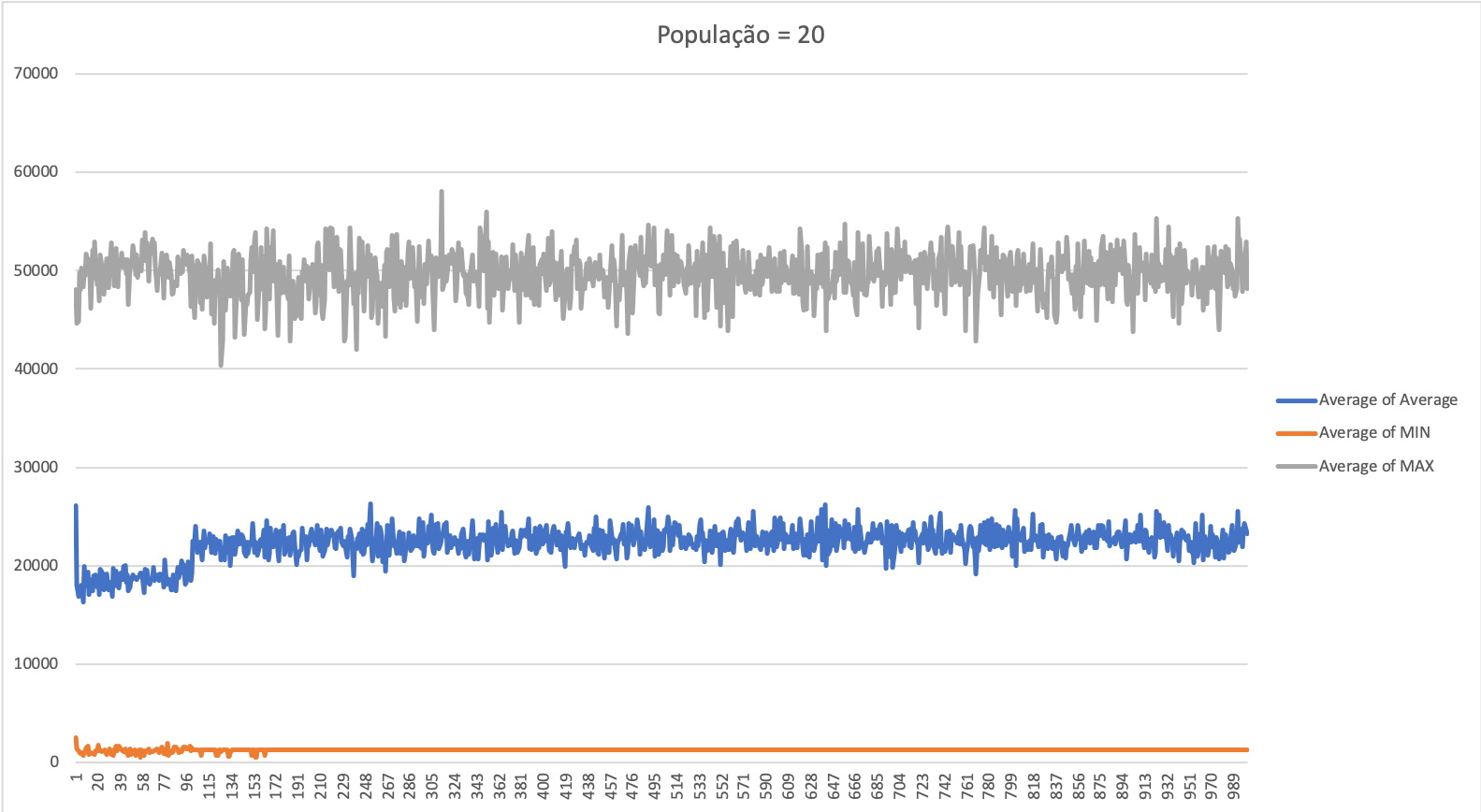
Isso, em consonância com a avaliação a curva de peso médio entre os indivíduos com menor peso, indica que o aumento no número de gerações não traz, sozinha, melhora significativa na população ou na obtenção do melhor indivíduo, embora também não cause impactos negativos. Porém se torna importante avaliar que uma maior quantidade de gerações acarreta em um maior esforço computacional e também demanda mais tempo de execução.

****

* + - 1. **Tamanho da População**

Em relação ao tamanho da população, embora as curvas se comportem da mesma forma, é possível notar que a curva de peso médio dos indivíduos é deslocada para baixo no gráfico de população mais elevada. É relevante ressaltar que essa consequência ocorre em consonância com a manutenção do peso médio dos indivíduos com maior pode indicar uma melhora nos caminhos, em geral.

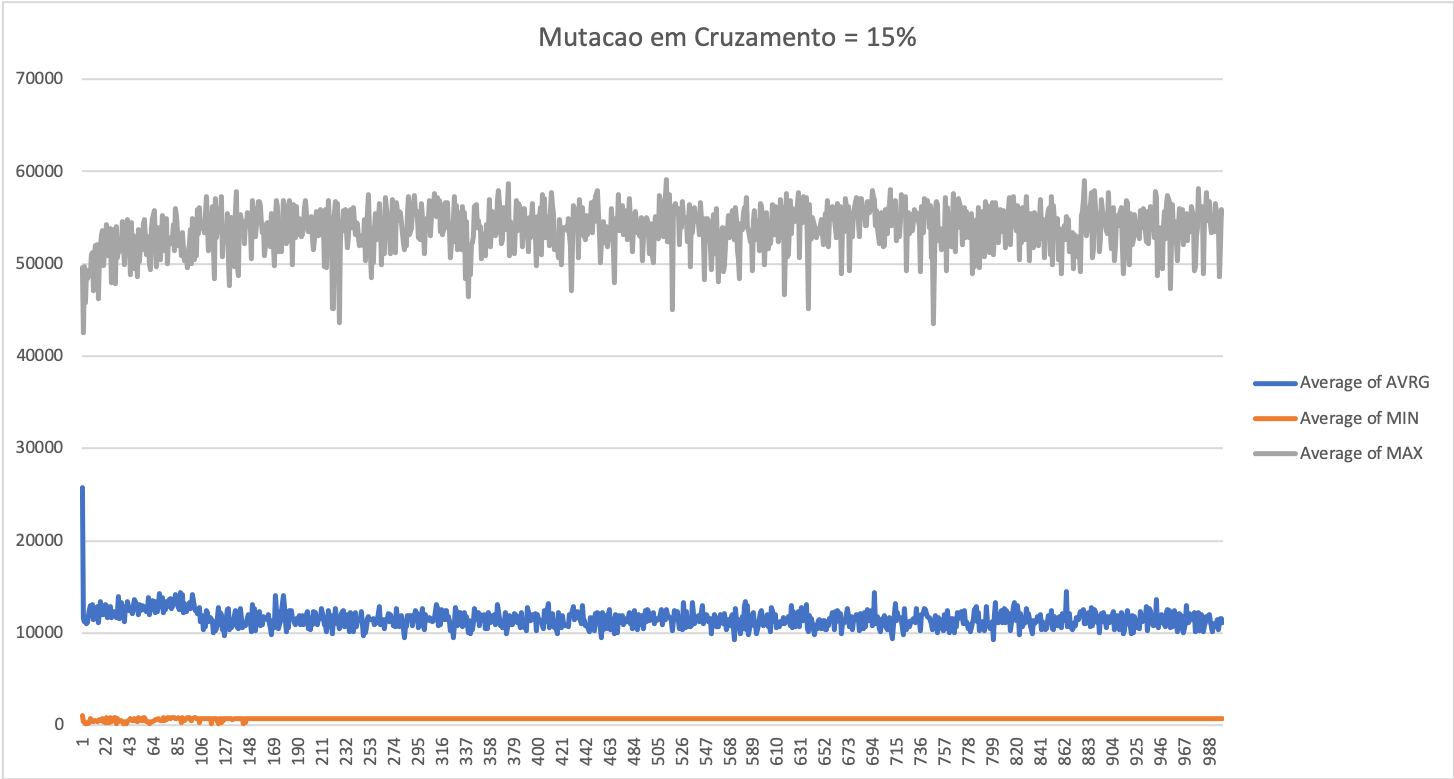
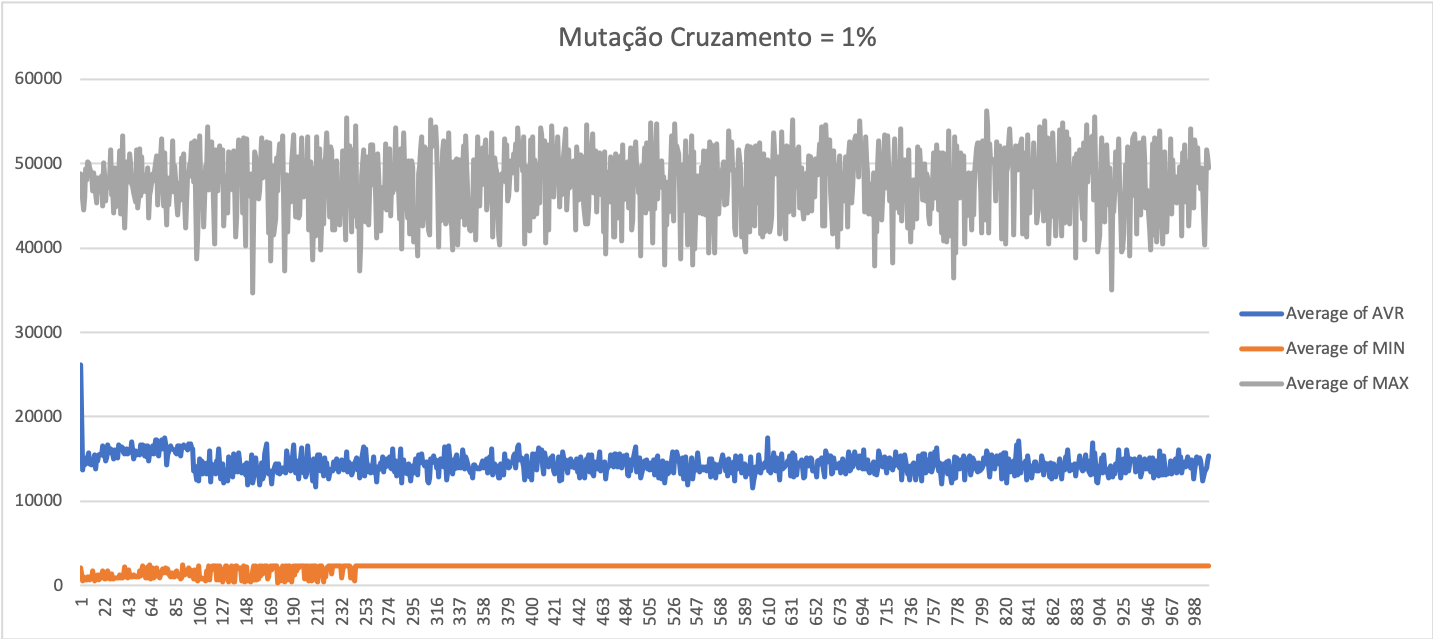
Assim como a Quantidade de Gerações, o Tamanho da População também acarreta em maior esforço computacional, além de implicar em maior ocupação da memória principal por ser necessário a alocação de mais caminhos dentro do mesmo ciclo de execução de geração.

****

* + - 1. **Taxa de Mutação nos Cruzamentos**

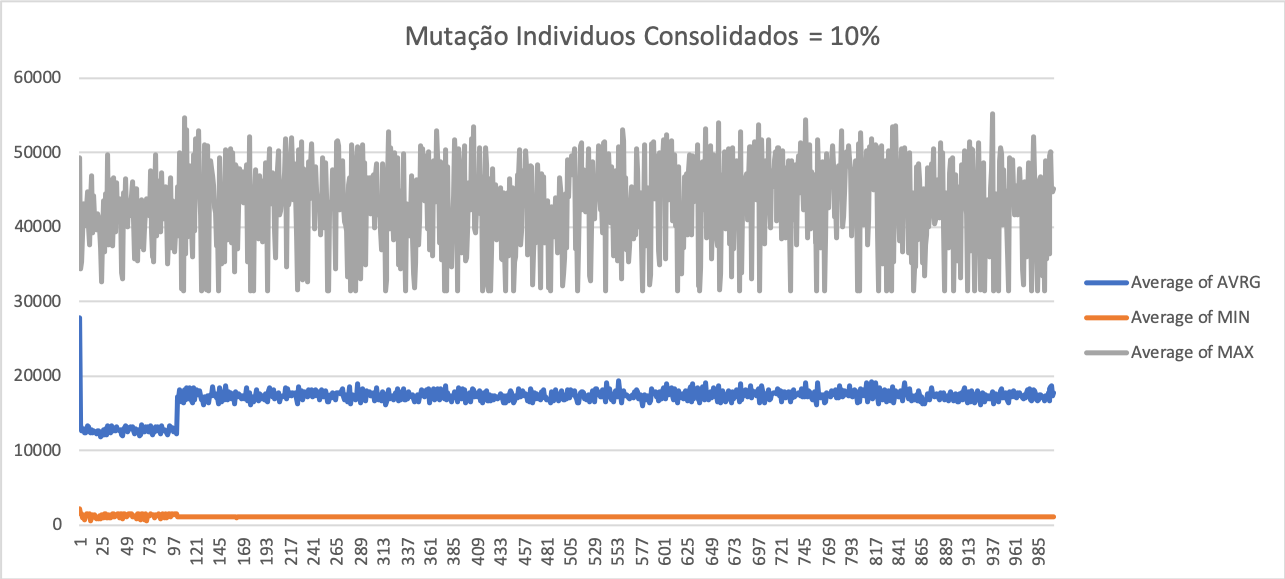
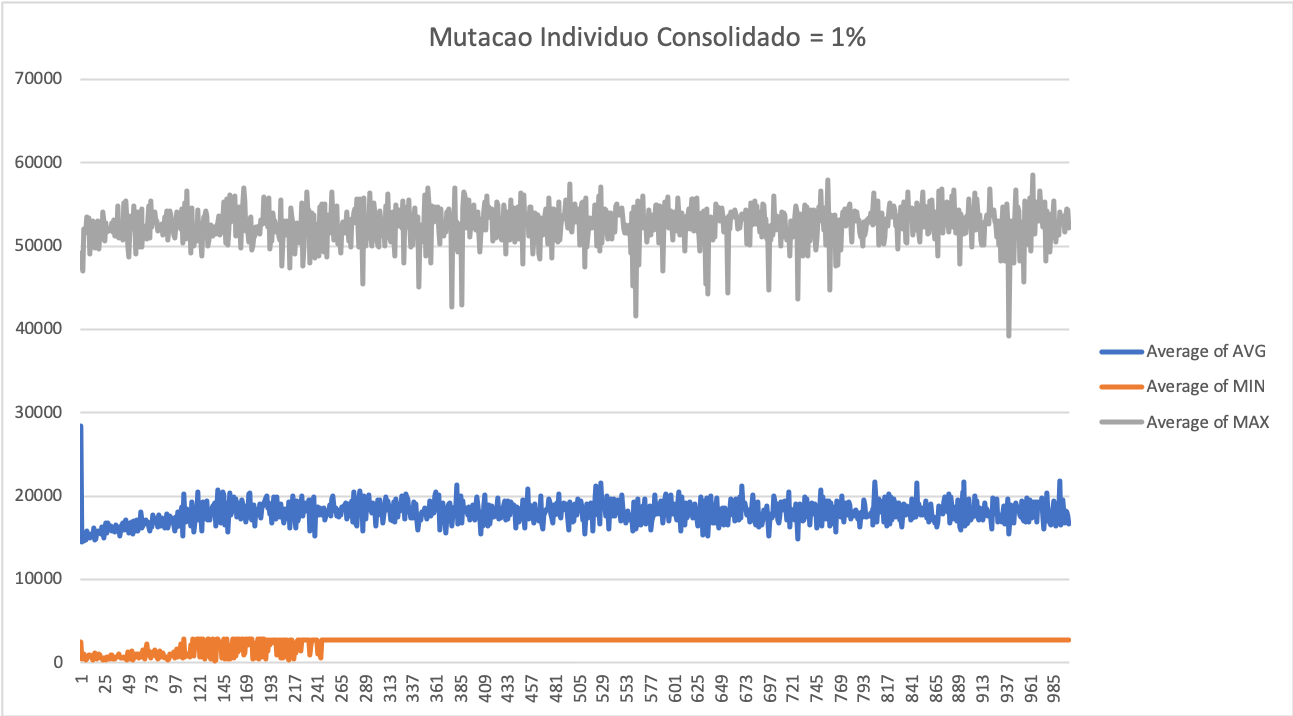
A Taxa de Mutação é um fator de extrema importância para a resolução do problema, visto que dá a possibilidade de aparição de genes que não estavam representados na população.

O aumento na taxa de mutação dentro dos cruzamentos acabou por contribuir para a melhora do resultado do algoritmo, não apenas por diminuir a média de peso da população, quando comparado a um estado com baixa mutação, mas também por diminuir e manter a média dos pesos mínimos.

****

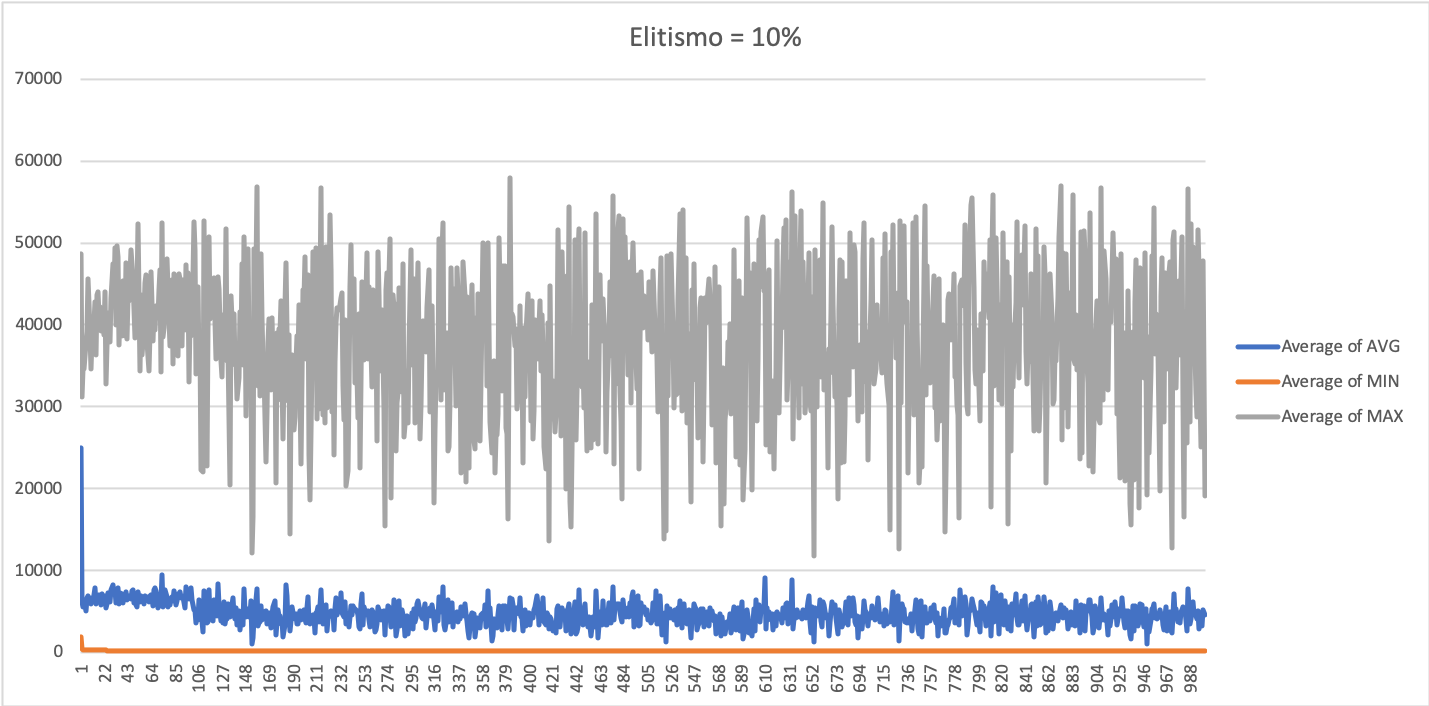
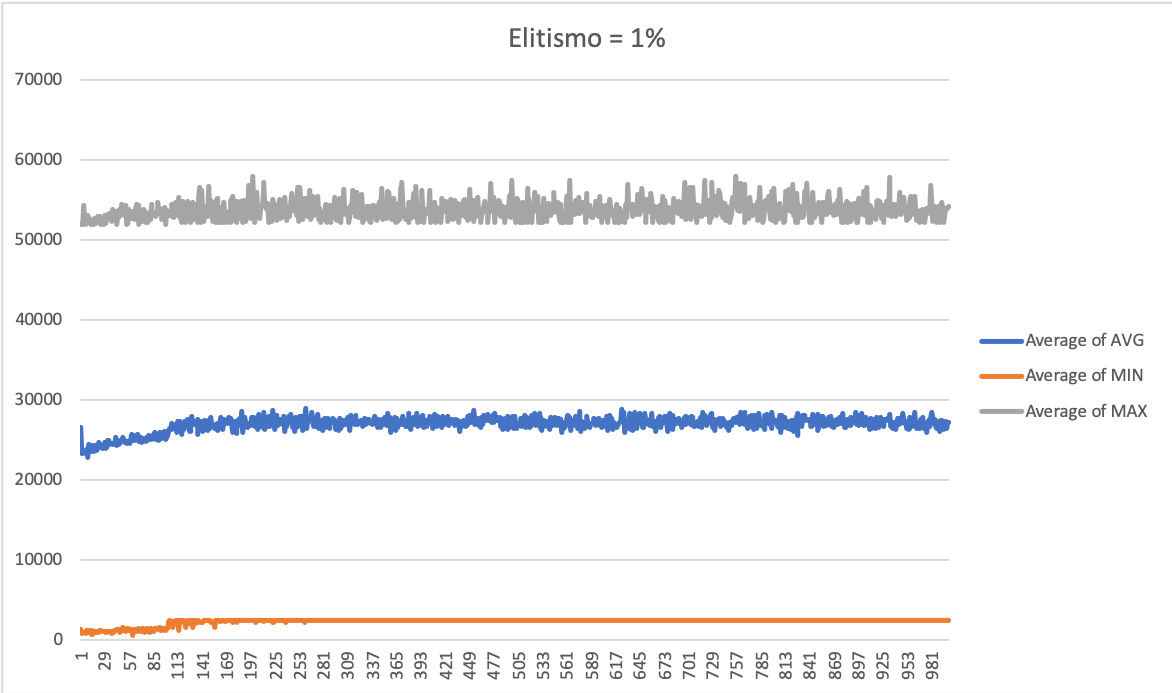
* + - 1. **Taxa de Mutação nos Indivíduos Consolidados**

Assim como a taxa de Mutação em Cruzamentos, a Taxa de Mutação em indivíduos consolidados traz um fator benéfico para a qualidade da população, fazendo com que seja possível o melhoramento de indivíduos que não sejam tão adaptados.

****

* + - 1. **Taxa de Elitismo**

A Taxa de Elitismo apresentou o resultado mais expressivo entre os parâmetros, essa taxa, que auxilia na perseverança de um indivíduo considerado apto para uma população específica, acaba por selecionar uma porcentagem dos indivíduos considerados adaptados e os mantém para a geração futura, sendo esse um mecanismo fundamental para se manter um progresso dentro de um algoritmo que caminha de forma aleatória.

****

* 1. **Comparações e expectativas**

1. **Experimentação**
   1. **Descrição da proposta experimental**
      1. **~~Definição do conceito de disponibilidade aplicada ao TSP~~**
         1. **~~Demonstração de implementação da disponibilidade~~**
   2. **Grafos sem peso**
      1. **Sem aplicação de disponibilidade**
         1. **Geração aleatória de grafo**
            1. **Código de geração aleatória de grafo**
         2. **Teste dos três algoritmos**
         3. **Resultados**
   3. **Grafos com pesos**
      1. **Sem aplicação de disponibilidade**
         1. **Teste dos três algoritmos**
         2. **Extrapolação: Grafo aleatoriamente gerado**
            1. **Código de geração do grafo**
         3. **Resultados**
2. **Discussão dos Resultados**
   1. **Genético**

O Algoritmo Genético tem a eficácia e custo computacional determinados pelos parâmetros descritos, mas também pelo acaso, visto que a cada execução são criados indivíduos aleatoriamente e a cada geração existem fatores e funções que podem ocorrer, ou não. Portanto não é de se surpreender que a cada execução, mesmo que sejam mantidos os parâmetros, haja a possibilidade de se obter saídas e resultados diferentes.

Torna-se evidente, então, que para esse tipo de algoritmo ser aproveitado dentro de um contexto não pode haver a necessidade de obtenção do melhor resultado global, ou de um único resultado absoluto para todas as execuções, sendo uma opção extremamente adequada e interessante para situações que permitam um resultado mais flexível por trazer boas respostas diferentes e situações em que não exista um algoritmo preciso para sua resolução.

* 1. **Comparação de eficiência para grafos sem peso**
  2. **Comparação de eficiência para grafos com peso**
  3. **Sugestão de melhor grafo para TSP**