Validação de implementação dos Algoritmos de Dijkstra e Genético para solução do Problema do Menor Caminho aplicado à malha metroviária de Nova Iorque

Nascimento, Diogo; Prado, Gabriel; Rocha, Samuel; Haluch, Pedro.
Ciência da Computação – Instituto de Ensino Superior de Brasília (IESB)

Abstract: This publication's purpose is to assess the applicability of both Dijkstra's and Genetic Algorithms in solving the Shortest Path Problem (SPP) when applied to the subway system of New York City. The city was chosen given its largest complexity when compared to any other subway system in the world. This means that, by extension, a plausible solution to the specific system may be applied to any other ones worldwide. All algorithms were implemented in the C programming language, providing a better understanding and control of the inner workings of both algorithms. We conclude that Dijkstra's algorithm is a good solution to the SPP, with caution warranted regarding computational costs. The Genetic Algorithm, however, is not suited given its non-deterministic nature and the time constraints imposed by the usual implicit SLA expected from this type of platform.

Resumo: A proposta desta publicação é determinar a aplicabilidade de dois algoritmos, Dijkstra e Genético, para solucionar o SPP aplicado à linha metroviária de Nova Iorque. Foi escolhida essa cidade por ter a malha mais densa dentre os sistemas desse tipo no mundo fazendo, portanto, com que os resultados encontrados sejam aplicáveis também aos outros casos menos complexos. A fim de melhor conhecer e controlar o comportamento dos algoritmos, toda a implementação foi feita na linguagem C. Conclui-se que a implementação de Dijkstra é uma solução boa, ponderando-se o custo computacional. A do Algoritmo Genético não, devido à sua natureza não-determinística e limitações de tempo de execução para respeitar o SLA implícito desse tipo de plataforma.

1. Introdução Teórica

1.1. Motivação

O Problema do Menor Caminho (SPP, do inglês *Shortest Path Problem*) é um dos principais exercícios de aplicação de algoritmos em Teoria de Grafos. Com proposta relativamente simples consegue ser aplicado a diversos desafios reais, dentre os quais um dos mais comuns é o de encontrar a melhor forma de se locomover por uma cidade (exemplo de plataformas como Waze e Google Maps), rotas de Drones (Mirzaeinia et. al, 2001), ou ainda decidir qual a melhor combinação de estações pelas quais passar para ir de um local a outro usando o sistema metroviário (Derekenaris et. al, 2000; Soares et al, 2014).

1.2. Representação de Grafos como vetores de listas ligadas

Um grafo é um conjunto de vértices V, unido a um conjunto de arestas A, que conectam pares de vértices distintos, com no máximo uma única aresta conectando qualquer par de vértices.(Sedgewick, 2002). Esse formato é ideal para a representação de pontos de parada e pesos de transição entre esses pontos. Além dos exemplos com que trabalharemos, pode-se usar grafos para representar redes de computadores e suas latências, redes telefônicas e a força de seus sinais, circuitos e suas caraterísticas, e muitos outros.

Das duas formas comuns de representação das adjacências em grafos, vetores e listas (Cormen et al, 1999), decidimos pela última pela elasticidade de uso da memória durante a execução do algoritmo.

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	1	0	0	1	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	0	0	0	0	0	0	0
4	0	0	0	1	0	1	1	0	0	0	0	0	0
5	1	0	0	1	1	0	0	0	0	0	0	0	0
6	1	0	0	0	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	1
10	0	0	0	0	0	0	0	0	0	1	0	0	0
11	0	0	0	0	0	0	0	0	0	1	0	0	1
12	0	0	0	0	0	0	0	0	0	1	0	1	0

```
0:  1 2 5 6
1:  0
2:  0
3:  4 5
4:  3 5 6
5:  0 3 4
6:  0 4
7:  8
8:  7
9:  10 11 12
10:  9
11:  9 12
12:  9 11
```

Figura 1: À esquerda representação de grafo como matriz de adjacência e à direita representação como lista de adjacência.

Enquanto a representação vetorial acelera o processo de consulta das conexões entre os vértices, a necessidade de instanciação a priori de um vetor de tamanho V^2 inviabiliza a escalabilidade dos experimentos.

1.3. Algoritmo de Dijkstra

Criado pelo holandês Edsger Dijkstra, o algoritmo propõe a solução do problema do caminho mais curto, podendo ser aplicado num grafo orientado (dirigido) ou não orientado, com arestas de peso não-negativo (Dijkstra, 1959)

Uma versão mais simples de Dijkstra, a título de completude, é o algoritmo de busca Breadth First Search (BFS). Este 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 BFS 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 (Tenenbaum, 1995).

A implementação de Dijkstra se dá pelo uso de uma fila com preferência pelo mínimo, o que quer dizer que o primeiro a sair será sempre o menor valor da fila, não importando a ordem em que os elementos entraram. Inicializam-se todos os vértices como estando a uma distância infinita da origem s, e a partir de s visitam-se todos os vértices adjacentes totalizando a distância necessária para chegar até eles. Por fim, esses vértices com suas distâncias são adicionados à fila, e a partir daí repete-se o processo sempre usando o menor elemento da fila, até que se chegue ao destino t. Se nesse processo um mesmo vértice for visitado mais de uma vez, prevalece o vértice de origem pelo qual o caminho é menor. Esse processo é chamado de relaxamento dos vértices.

Dijkstra, assim como o BFS, tem a propriedade de sempre definir o menor caminho entre a origem e todos os vértices pelos quais passa se tiver passado por todos os seus adjacentes, por sempre substituir um caminho maior a um vértice v por um menor que seja descoberto (Sedgewick, 1998).

1.4. Algoritmo Genético

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.

O algoritmo genético é utilizado em diversos setores (Bilal, 2006), 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 (Ruiz et. al, 2019), problemas de otimização, processamento de imagens, machine learning, e o TSP (Traveling Salesman Problem) que se assemelha com o algoritmo descrito nesta análise. Em sua 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.

A fim 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 (Whitley, 1994).

1.4.1. 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.

2. Representação das estruturas em código

O projeto contou com três implementações complementares de código. O primeiro gera aleatoriamente um grafo de acordo com os parâmetros fornecidos, e o grava em um arquivo .txt em formato pré-estabelecido. O grafo do metrô foi montado seguindo os mesmos padrões a fim de assegurar que o formato do arquivo não interfira na mensuração de resultados. A segunda implementação, que usa Dijkstra, lê esse arquivo e constrói em memória o grafo representado, partindo então à aplicação do algoritmo. O mesmo é feito pela terceira, que usa o algoritmo Genético. Esse último representa de forma distinta o grafo em memória, para respeitar as peculiaridades do algoritmo. Todo o código pode ser encontrado no repositório do GitHub https://github.com/phaluch/projeto_integrador.

2.1. Grafo

O código de geração do grafo cria primeiro um ponteiro para um vetor de ponteiros de cabeçalhos. Para cada um, o valor é inicializado com o índice do vetor para representar o vértice. Depois disso é preenchido com os links mínimos, ligando os vértices do início ao fim um a um, para garantir que todos os vértices estejam conectados. Dessa forma, também, garante-se que a priori o primeiro e último vértices estejam o mais distante possível um do outro. Depois disso itera-se sobre todos os vértices adicionando aleatoriamente links entre quaisquer 2 vértices até atingir-se a quantidade de links que respeita a densidade selecionada.

2.2. Dijkstra

Para a implementação de Dijkstra aproveitamos o formato de grafo utilizado pelo código gerador, e criamos apenas um novo tipo de operação de push, o pushMinimo. Ao declarar uma lista normalmente e usar apenas essa função, criou-se sem necessidade de muito mais código a lista com prioridade pro mínimo necessária. Essa função antes de adicionar um novo elemento na lista ligada confere se já existe esse vértice na lista. Se houver, o remove e compara os dois selecionando o de menor caminho. Depois disso re-insere o elemento na lista, imediatamente antes do um elemento que tiver valor igual ou maior do que si, ou ao final da lista caso seja o maior.

2.3. Genético

Para a implementação do Algoritmo Genético foi aproveitada a estrutura de lista ligada pré-definida e foram definidos parâmetros globais, como tamanho da população, quantidade de gerações, taxa de elitismo, taxa de mutação em cruzamento, taxa de mutação em indivíduos consolidados, ponto de origem e ponto de destino. Após esse momento é criada uma população de quantidade de indivíduos pré-definida, sendo os caminhos que

compõem os indivíduos dessa população criados a partir da origem até o destino aleatoriamente, utilizando-se sempre de conexões válidas definidas no grafo. Em seguida é feita a pesagem de cada indivíduo, somando-se o peso entre cada conexão que compõem o caminho, calculado o fitness de cada indivíduo, sendo esse uma relação inversa entre o peso do indivíduo e o peso total da população e, baseado na proporção do fitness do indivíduo em relação ao somatório do fitness da população, o indivíduo deste é colocado o número de vezes que representa o esse percentual em um vetor de 100 posições (se seu fitness corresponde a 30% do somatório dos fitness, ele é adicionado 30 vezes no vetor), que é utilizado para orientar a escolha dos pais na hora do cruzamento. Dois pais são escolhidos de maneira aleatória dentro desse vetor e, então é feita a análise de pontos em comum existentes para ambos. Posteriormente essa análise, é feita a cópia dos indivíduos escolhidos como pais e um dos pontos avaliados é selecionado aleatoriamente para que seja realizada a troca das porções entre as cópias, que seria a mudança cruzada do conteúdo dos ponteiros. Após esse evento é aplicada uma taxa correspondente a chance de ocorrer mutação, que escolhe um ponto aleatório dentro do caminho do indivíduo e baseado nas conexões feitas por esse ponto no grafo, selecionado aleatoriamente um outro nó para substituir o nó selecionado, após isso é feita a correção do caminho do cromossomo, de forma que todas as conexões posteriores sejam consideradas válidas e também é aplicada uma função de remoção de loops, que encontra duplicidades dentro do caminho e quando a encontra realiza a ligação direta entre a primeira aparição desse e o nó posterior a segunda aparição. Após essa etapa, um percentual dos melhores indivíduos é selecionado para permanecer na próxima população e um outro percentual é selecionado para sofrer outro processo de mutação com correções(Darrel, 1994).

3. Experimentação

Para comparação da razoabilidade do tempo de demora, calculou-se o tempo necessário pelo site do sistema metroviário de Nova Iorque para a sugestão de uma rota entre extremos da malha de estações. O tempo necessário para geração de resposta foi 3 segundos, dos quais 1 foi subtraído para descontar a latência entre a requisição e resposta do servidor.

3.1. Equipamento

Para todos os testes, a fim de remover a necessidade de normalização de processamento, foi usando o mesmo equipamento a saber:

- ➔ Notebook Samsung Modelo Expert 300E
- ➔ 8Gb Memória RAM DDR4 2133Hz
- ➔ Processador Intel® Core™ i5-7200U CPU @ 2.50GHz × 4
- ➔ Samsung SSD 840 PRO Series 256GB SATA 3.1, 6.0 Gb/s
- ➔ (Irrelevante para os testes) GeForce 920MX/PCIe/SSE2

3.2. Grafo do Metrô

O Metrô de NY pode ser representado por um grafo de 487 vértices e densidade de aproximadamente 0.3. (fonte: site de NY e tal).

3.3. Dijkstra

A análise de resultado do algoritmo de Dijkstra é muito determinística. Pela natureza do algoritmo ele não tem componente aleatório, o que quer dizer que para o mesmo input sempre será gerado o mesmo output. Sendo assim, para a determinação da eficiência do algoritmo ele é aplicado uma vez sobre o grafo.

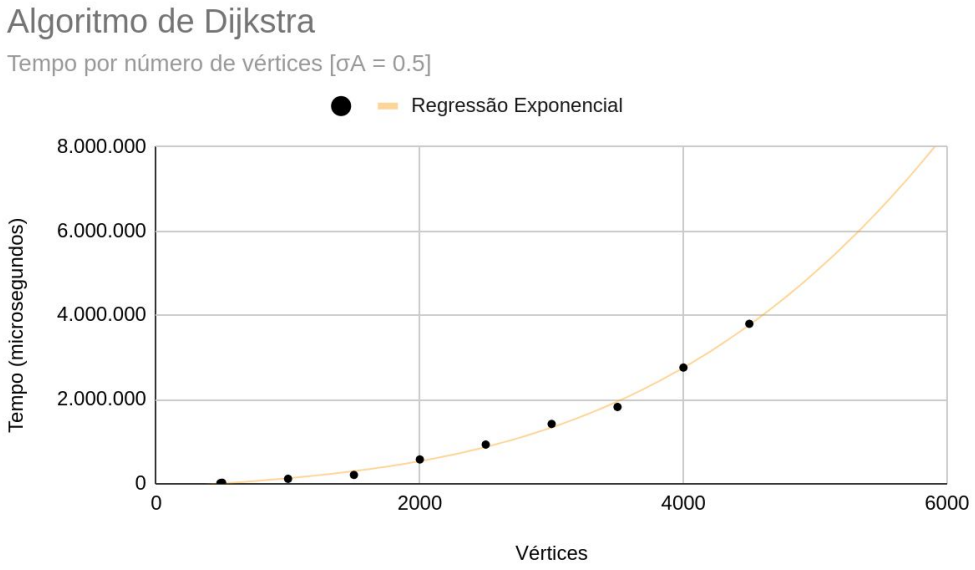

Gráfico 1: Tempo para solução do SPP usando Dijkstra. O metrô de NY é o primeiro ponto do gráfico.

Podemos notar que, para número de vértices inferior a 3000, o algoritmo consegue encontrar o menor caminho em menos de 2 segundos. Outros testes foram feitos com grafos de até 4500 vértices, e depois regressão exponencial foi aplicada para estimar o tempo necessário para a solução de grafos com até 6000 vértices. O caminho mínimo do ponto 1 ao 467 é de 20 unidades de distância.

3.4. 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).

Para cada combinação de parâmetros foram feitas 10 iterações, programadas para serem interrompidas depois de 2 segundos de execução. Então, a média do menor valor dos pesos na última população criada foi calculada. Esse threshold foi escolhido para proporcionar comparação entre o Algoritmo de Dijkstra e o benchmark de tempo de execução do site do MTA.

A execução foi feita usando o grafo representando o metrô de NY, e durante os testes foram variados 4 parâmetros, como mostrado a seguir,

Tamanho da População: 100 a 1000, em passos de 100;

Taxa de Mutação nos Cruzamentos: 10% a 40%, em passos de 10%;

Taxa de Mutação nos Indivíduos Consolidados: 10% a 40%, em passos de 10%;

Taxa de Elitismo: 10% a 40%, em passos de 10%;

Considerando as 10 iterações para cada combinação de parâmetros, foram executados 6400 iterações.

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. Os resultados são apresentados na Tabela 1 abaixo; Nota-se que em geral, o maior contribuinte para uma melhor performance é o número de indivíduos na população, pelo aumento de células de coloração verde em direção à direita. A taxa de mutagênicos, em contrapartida, é a variável que menos altera os valores, dada a distribuição aleatória dos elementos a cada linha. É também observada uma leve aglomeração de melhores valores para o intervalo População (700 -> 1000), Taxa de Mutação (30 -> 40), Taxa de Elitismo (40 -> 30), Taxa de Mutagênicos (Todos os valores). O menor caminho encontrado foi com a combinação 96,3, com os valores de parâmetros (na ordem acima): {900, 40, 20, 10}.

AVERAGE de Menor_Populacao			*Populacao*									
Tx_Mutacao	*Tx_Elitismo*	*Tx_Mutagenicos*	100	200	300	400	500	600	700	800	900	1000
10	10	10	283	230,1	236,6	173,9	174,2	136,6	165,2	120,5	148,7	145,8
10	10	20	370,4	178,6	217,8	219,8	186,4	165,9	148,4	176,9	173	138,7
10	10	30	331,9	234,9	212,5	213,2	186,7	174,9	197,7	185	180,5	148,1
10	10	40	274,8	282,7	259,4	250,8	205,8	184,8	170,5	116,5	196,8	122,2
10	20	10	292,9	216,9	192,8	217,3	177,1	158,9	182,9	138,5	134,6	133,5
10	20	20	276,6	300,2	145,6	153,8	195,2	150,9	134,6	172,6	131,8	137
10	20	30	286,3	197,2	157,9	234	157,6	174,3	161,3	130,4	139,8	130,8
10	20	40	278,3	247,8	228,6	158,1	215,7	217	161,2	125,4	160,5	157,5
10	30	10	377,1	232,9	164,5	192,8	160,9	181	147,8	143,7	141	188

AVERAGE de Menor_Populacao			Populacao									
Tx_Mutacao	Tx_Elitismo	Tx_Mutagenicos	100	200	300	400	500	600	700	800	900	1000
10	30	20	260,9	239,7	221,6	173,4	176,1	156	150,9	134,9	141,6	140,7
10	30	30	234,2	265,3	189,2	255,9	167,1	163,3	179,9	126,1	138,2	121,8
10	30	40	309,7	241,5	127,3	173,7	199,9	210,9	163,7	160,6	144,4	146
10	40	10	315,2	227,7	226	186,2	179,2	197,1	125,7	130,2	155,5	134,8
10	40	20	232,6	267,7	165,7	201	184,3	160	156,9	174,1	177,9	154,6
10	40	30	242,2	284,9	221,4	205,5	204,2	193,2	184,2	192,6	172,7	156,7
10	40	40	225,9	238,4	278,8	159	161,6	161,6	134,3	164,2	134,2	131,4
20	10	10	258,4	204,6	216,8	143,5	144	167,3	170,8	135,2	147,5	137,8
20	10	20	291,8	215,4	242,3	166,2	125,9	157,1	154,7	158,8	159	168,3
20	10	30	233,5	332,4	215,5	143,1	199,4	131,8	124	180,9	121,5	114,2
20	10	40	412,4	206,9	195,6	178,1	198,9	161,4	150,3	119,4	115,9	141,6
20	20	10	273,4	191	230,4	156	156,4	188,4	100	163,7	149,8	132,7
20	20	20	291,6	313,6	209,9	170,2	187	169,3	176,6	155,1	168,6	162,7
20	20	30	344,3	302,4	205,8	219,6	181,5	180,2	180,7	150,4	125,7	123
20	20	40	279,5	226,3	215,1	136,9	208	149,7	167,4	122,6	166,9	135,9
20	30	10	238,5	250	206,1	250,2	172,2	206	124,7	152,4	131,9	166,8
20	30	20	323	217,2	202,6	192,6	201,1	173,1	138	167,5	170,3	158,6
20	30	30	363,7	276,2	212,3	155,3	166,4	158	122,6	167,5	123,1	132,9
20	30	40	238,2	239,1	198,6	148,7	217,3	148,8	141,2	139,3	129,8	161,2
20	40	10	271,2	282,9	228,2	250,9	169,6	148,4	155,9	145,6	133,2	135,4
20	40	20	206,9	239,8	251,7	187,8	136,7	167,7	147	157,8	142,5	145,2
20	40	30	326	229,9	199,2	221,1	165,4	167,5	155,8	145,1	120	149,1
20	40	40	316,3	238,4	224,1	179	183,8	190	138,7	112	125,7	128,5
30	10	10	298,5	200,7	151,8	200,1	202,3	143,3	147,6	130,3	113,1	137,1
30	10	20	318,8	270	208,1	219	153	169,9	169,8	172,7	147,5	145,4
30	10	30	283,9	205,9	173,6	188,9	135,2	163,6	161,5	130,2	139,4	155,5
30	10	40	256,4	261	217,8	134,5	136,8	140,1	181,7	143,1	133,3	152
30	20	10	257,7	191,7	207,9	223,8	145,1	156,8	159,8	104,2	130,2	151,8
30	20	20	400,2	227,5	184,5	139,7	178,1	117,3	155	131,1	142,1	111,3
30	20	30	310,2	230,4	160,6	190,8	128	208,2	167,2	156,8	119,5	143,8
30	20	40	318,9	179,8	192,5	174,5	176,7	142,7	162,5	170,5	149,4	135,3
30	30	10	323,9	242,7	227,6	161	142,5	141,6	152,5	178,8	187,3	135,9
30	30	20	301	212,9	206,7	198,1	196	175,2	139,2	156,3	120,5	148,8
30	30	30	284,2	231,8	171,1	136,9	158,9	133,3	158,7	153,6	124,4	120,5
30	30	40	324,2	217,5	220,4	135,1	187	151,1	125	153	115,1	126,3
30	40	10	326,8	232,9	186,2	165,9	205	179,4	164,4	165,7	145,2	109,6

AVERAGE de Menor_Populacao			Populacao									
Tx_Mutacao	Tx_Elitismo	Tx_Mutagenicos	100	200	300	400	500	600	700	800	900	1000
30	40	20	258	261,1	156,8	246,4	135,1	142,3	144,7	129	147,8	123,7
30	40	30	355,5	156,5	152,7	165,7	175,3	127,4	149,2	108,5	118,7	138,4
30	40	40	258,6	219,6	223,6	238,6	170,7	178,9	124,4	141,5	152,1	130,4
40	10	10	317,9	253,7	279,2	148,6	122,3	159,2	117,1	130,6	108,9	128,7
40	10	20	208,2	224,3	239,6	195,8	151,8	166,1	173,8	125,4	135,7	119,6
40	10	30	295,6	230	197,6	209,7	200	141,3	121,4	195,3	137,5	108,2
40	10	40	245,6	257	204,7	191,5	142,5	172,6	129,7	150,4	137,4	123
40	20	10	329,4	256,9	180,5	136,2	191,9	143,4	149,9	167,3	96,3	148,7
40	20	20	382,3	231,1	160	218,8	157,8	153,1	173,5	158,6	133,6	173,9
40	20	30	264,9	250,5	135,2	155,8	157,7	124,9	163,2	161	149	105,6
40	20	40	254,4	298,1	151,2	169,2	156	137,1	140,5	128,6	133,4	122,3
40	30	10	239	220,9	189,1	200,3	179,5	168,9	150,1	101,7	128,2	141,5
40	30	20	320,2	256,8	166,5	136,4	201,4	148,5	125,7	140,9	112,4	128,3
40	30	30	319,1	265,5	178,7	145,1	120	141,3	144,4	135,2	135,8	138,2
40	30	40	287,7	243,8	182,9	197,1	147	161,7	143,4	147,5	104,6	152,1
40	40	10	309,3	209,1	182	175,3	153	144	163,1	135,5	128,5	158
40	40	20	185	211,3	201,6	182,1	146,6	178,6	162,1	136	147,4	121,3
40	40	30	315,7	297,2	222,9	212,5	161,4	161,5	142,7	125	161,7	161,4
40	40	40	279,8	253,1	211,9	169,5	157,1	194,5	139	142,5	126,7	137,6

Tabela 1: Combinações de parâmetros para Algoritmo Genético. Valores em verde são mais desejados, em vermelho menos desejados. O valor menos desejável encontrado foi 412,3; o mais desejável 96,3.

4. Conclusão

O algoritmo de Dijkstra foi capaz de solucionar o problema específico do Metrô de NY em 0.015s, bem mais do que o necessário para o SLA praticado pelo site do MTA. O Algoritmo Genético, no mesmo período de tempo, apresentou para os melhores parâmetros a distância mínima de 96,3 unidades de distância, valor 481,5% maior do que o mínimo real de 20 unidades de distância. Esse resultado não é surpreendente: pela natureza randômica do Algoritmo Genético espera-se certa variabilidade nos resultados, principalmente com o número relativamente pequeno de gerações a que nos limitamos devido ao limite de tempo imposto.

5. Próximos Passos e Considerações Finais

Pudemos concluir que o Algoritmo de Dijkstra é um bom candidato a ser implementado para resolução não só desse grafo mas outros de densidade similar, e até 4000 vértices. Ponderações devem ser feitas, no entanto, quanto ao custo computacional de sua

implementação. O Algoritmo Genético, por sua vez, não é recomendado para a solução deste problema específico, dada a sua variação inerente, não é aceitável na definição de itinerários. Para definição definitiva de sua usabilidade neste problema precisaríamos experimentar com uma capacidade maior de processamento, a fim de estudar possíveis melhorias nos resultados obtidos chegando a um número maior de indivíduos na população. Isso, no entanto, também aumentaria custos sem necessidade tendo em vista a existência de outros algoritmos mais eficientes.

6. Referências Bibliográficas

Mirzaeinia, A., Shahmoradi, J., Roghanchi, P., & Hassanalian, M. (2019). Autonomous Routing and Power Management of Drones in GPS-Denied Environments through Dijkstra Algorithm. AIAA Propulsion and Energy 2019 Forum. doi:10.2514/6.2019-4462

Edsger W. Dijkstra. (1959) A note on two problems in connexion with graphs. Numerische Mathematik, 1:269-271.

Tenenbaum, Aaron M.; Langsam, Yedidyah; Augenstein, Moshe J. (1995) Estruturas de dados usando C. São Paulo: Pearson Makron Books.

Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. (1999) Introduction to Algorithms. MIT Press Publ, Cambridge Mass, 22 edition.

Sedgewick, Robert. (2002) Algorithms in C, Part 5: Graph Algorithms. Addison-Wesley.

Sedgewick, Robert. (1998) Algorithms in C, Parts 1-4: Fundamentals, Data Structures, Sorting, Searching. Addison-Wesley.

Sérgio A. F. Soares, Brauliro G. Leal, Renato G. B Pereira, Ivonaldo F. S. Junior, Danilo M. B. Silva. (2014) Algoritmo de Dijkstra Aplicado ao Problema do Metrô de Paris.

Whitley, Darrell. A genetic algorithm tutorial, 1994. Statistics and Computing.

Gonen, Bilal. Genetic Algorithm Finding the Shortes Path in Networks, 2011. Conference: The 2011 International Conference on Genetic and Evolutionary Methods

Efrain Ruiz, Valeria Soto-Mendozab, Alvaro Ernesto Ruiz Barbosac, Ricardo Reyesd. 2019. Solving the open vehicle routing problem with capacity and distance constraints with a biased random key genetic algorithm. Journal of Computers & industrial Engineering, 133(2019) 207-219.