Universidade Federal Rural de Pernambuco

Departamento de Estatística e Informática

Bacharelado em Sistemas de Informação

FlyFood

Jéssica Moreira Leal

Recife

Outubro de 2021

1. Introdução

No contexto no qual o serviço de delivery é amplamente requisitado, mas o alto custo da mão-de-obra humana e da extensa espera por pedidos tornaram necessário a implementação do uso de drones no serviço de delivery por uma empresa juntamente com um algoritmo de roteamento eficiente.

Nesse sentido, o objetivo geral do trabalho é propor um algoritmo de roteamento de drones com a finalidade de sanar a necessidade de um algoritmo que trace os menores percursos tendo em vista reduzir os custos e concluir todas as entregas dentro do tempo de duração da bateria do drone. Para a realização do objetivo geral, são necessários cumprir objetivos específicos: Identificar os endereços de entrega e de partida/retorno; gerar todas as sequências de rotas possíveis dado os endereços de entrega, calcular o custo em dronômetros de cada rota e encontrar, por intermédio de comparações, a rota menos custosa.

A justificativa do trabalho é dada pela necessidade de reduzir os custos com a mão-de-obra, adaptar-se ao mercado atual e melhorar a experiência do cliente com a diminuição do tempo de espera pelo produto e redução de avarias.

Dessa forma, a organização do trabalho foi definida em tópicos para melhor fluidez e entendimento do leitor, sendo elas: Introdução do trabalho; referencial teórico, em que haverá uma análise crítica da literatura abordada nesta pesquisa; procedimento, no qual abordará o passo-a-passo da solução desenvolvida; resultados obtidos; conclusão, a fim de analisar se o objetivo inicial foi devidamente cumprido através dos resultados e, por fim, as referências bibliográficas para a criação do trabalho.

2. Referencial Teórico

Nesta seção é apresentado a literatura do trabalho para melhor compreensão dos mesmos.

2.1. Complexidade de Algoritmos

O custo computacional é definido como a quantidade de recursos, tal como tempo de processamento e espaço de memória da máquina, demandado por determinado algoritmo, em função do volume n de dados de entrada.

É importante ressaltar que o custo computacional não é a representação real das unidades de complexidade temporal e computacional, pois tal caso depende de vários fatores além do algoritmo, como hardware da máquina, arquitetura de software, estado atual da máquina, etc. Logo, o custo computacional é representado através da taxa de crescimento do processamento e consumo de memória em função do comprimento da entrada de dados, sendo de complexidade polinomial a medida que cresce linearmente e de complexidade exponencial caso cresça exponencialmente.

2.2. O Problema do Caixeiro Viajante

O problema do Caixeiro Viajante é um clássico problema de otimização combinatória[1], visto que sua importância é definida pelo fácil entendimento, ampla classe de problemas da área de otimização combinatória e extensa aplicabilidade, tal como a utilização do PCV no roteamento de veículos, no problema de coleta de pedidos em depósitos, na fiação de computadores, na cristalografia de Raio-x, etc[2]. O problema do Caixeiro Viajante é baseado no ciclo hamiltoniano, no qual é um circuito no grafo hamiltoniano $G = (V, E)$ que visita todos os vértices somente uma vez a fim de encontrar o caminho menos custoso. Do mesmo modo, o vendedor deseja viajar de uma cidade i para uma cidade $j$sendo o custo $c(i, j) = \{1\ se\ (i,\ j) \in E,\ 0\ caso\ contrário\}$e o custo total $c(P) > 0$ e mínimo. Assim, se há $n$cidades a serem visitadas, então haverá $(n - 1)!$ rotas possíveis sendo uma das cidades o ponto de partida e de retorno. Tanto o problema do caixeiro viajante quanto o problema exposto no trabalho possuem uma relação, ambos buscam visitar

todos os pontos mapeados na rota menos custosa. Logo, ambos se utilizam da mesma teoria para elaborar uma solução coerente com a sua realidade.

2.3. Algoritmos Genéticos

Os algoritmos Genéticos (AGs) foram introduzidos pelo professor John Holland[3] a fim de, baseado em mecanismos evolutivos, traduzir os procedimentos da estrutura genética para uma estrutura algorítmica capaz de ser interpretada por uma máquina, com uma ampla aplicabilidade e de alta ou razoável qualidade. Inicialmente, há uma população de espécimes que são submetidas a uma função matemática (fitness function), determinada pelo desenvolvedor, para analisar o quão adaptadas os espécimes estão em tal contexto provido pelo desenvolvedor do algoritmo e selecionar os melhores espécimes segundo a função matemática. Posteriormente, é realizado o "cross-over" entre os espécimes selecionados e a mutação que gerará novos espécimes. Com isso, é possível repetir o processo até gerar espécimes que satisfaçam suficientemente o contexto no qual foram inseridos.

3. Procedimento

Na construção do algoritmo genético para solucionar o nosso problema, será necessário um conjunto de métricas, algumas probabilísticas, outras pré-estabelecidas. Assim como a representação do cromossomo, que no caso será de tamanho n de cidades e representado por letras, no qual cada letra significa um ponto específico a ser visitado.

A estrutura do algoritmo genético se estabelece da seguinte forma: primeiramente há a geração de uma população de cromossomos (rotas) aleatoriamente, posteriormente essa população será submetida a um cálculo de aptidão, também chamado de função de fitness, o cálculo de aptidão será estabelecido através da função $f(d) = d^{-1}$, sendo d a distância de cada rota. Quanto maior o valor da função de aptidão, melhor é a solução.

Dessa forma, em seguida faremos a seleção dos pais que irão gerar a próxima geração a partir dos seus genes, a técnica de seleção utilizada será a seleção por ranqueamento, no qual a partir do valor de fitness de cada cromossomo é realizado um ranking decrescente. O método de ranking garante uma maior diversidade sem diminuir

bruscamente o valor do fitness das melhores soluções. A quantidade de indivíduos selecionados foi definida como seis de uma população de vinte indivíduos no total.

Após os indivíduos serem selecionados, é realizado o cruzamento entre esses indivíduos. O cruzamento ocorre de forma aleatória, no qual uma certa quantidade dos genes do pai A ou pai B são selecionados aleatoriamente e inseridos sequencialmente na composição dos cromossomos dos filhos. Caso o tamanho do cromossomo tenha valor par, os filhos recebem metade dos genes aleatórios de cada pai, do contrário cada filho receberá um gene a mais de um pai e um a menos do outro.

Em seguida, feito o cruzamento, é realizado o processo de mutação para aumentar a diversidade da população e reduzir a convergência genética. A taxa de mutação por cromossomo foi ajustada a 10% (dez por cento) e 0,5% (cinco décimas por cento) por gene. O processo de mutação funciona da seguinte forma: Primeiro é gerado valores aleatórios para utilizar como valores probabilísticos e decisórios no processo de mutação, caso esses valores estejam de acordo com as taxas de mutação, é realizada a mutação. Posteriormente, na mutação há a troca entre genes do mesmo indivíduo, o primeiro gene da troca é determinado por fatores probabilísticos, mas o segundo é escolhido de forma aleatória. Dessa forma, o par de genes efetuam a troca de posições.

Subsequentemente, uma nova população é gerada. Essa população repetirá o ciclo da população anterior, que será descartada e a única sobrevivente vai ser a melhor solução, que será comparada com as populações posteriores. Esse ciclo é repetido continuamente até atingir a 30° (trintagésima) população. O resultado desse algoritmo deve ser a melhor solução descoberta até então.

4. Resultados

Para analisar os resultados e comparar os algoritmos de força bruta e genético, iremos utilizar casos testes. A medição será feita através do tempo em segundos gasto para a execução do script, essa medição será feita através do módulo *time*.

Além disso, analisaremos a qualidade das soluções resultantes do algoritmo genético com diferentes parâmetros na taxa de mutação por gene.

4.1 Custo em Segundos

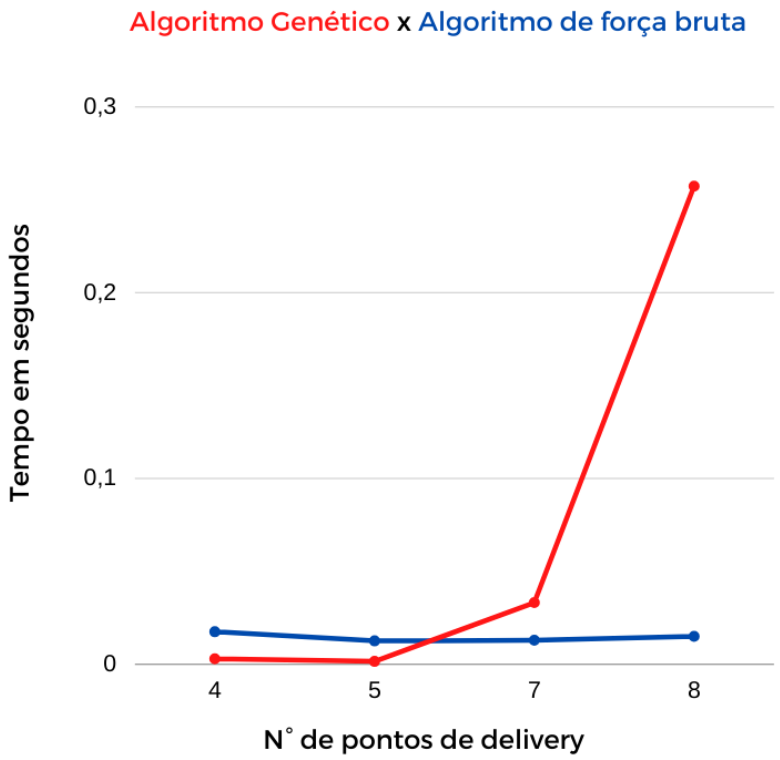

Figura 1. Custo em segundos de acordo com o número de pontos de delivery em cada teste

Ao analisar a imagem acima, é notório perceber que o algoritmo de força bruta é sustentável apenas para um limitante número de pontos de delivery, pois quando extrapola o limite de pontos de deliver, mesmo que seja um pequeno aumento, ocorre a explosão combinatória. Além disso, o algoritmo genético é sustentável quando há muitos pontos de delivery, seu tempo de execução permanece constante. Logo, é muito

custoso utilizar o algoritmo de força bruta quando há um grande número de entregas, o que torna o algoritmo genético mais eficiente nesses casos.

4.2 Fitness por taxa de mutação genética

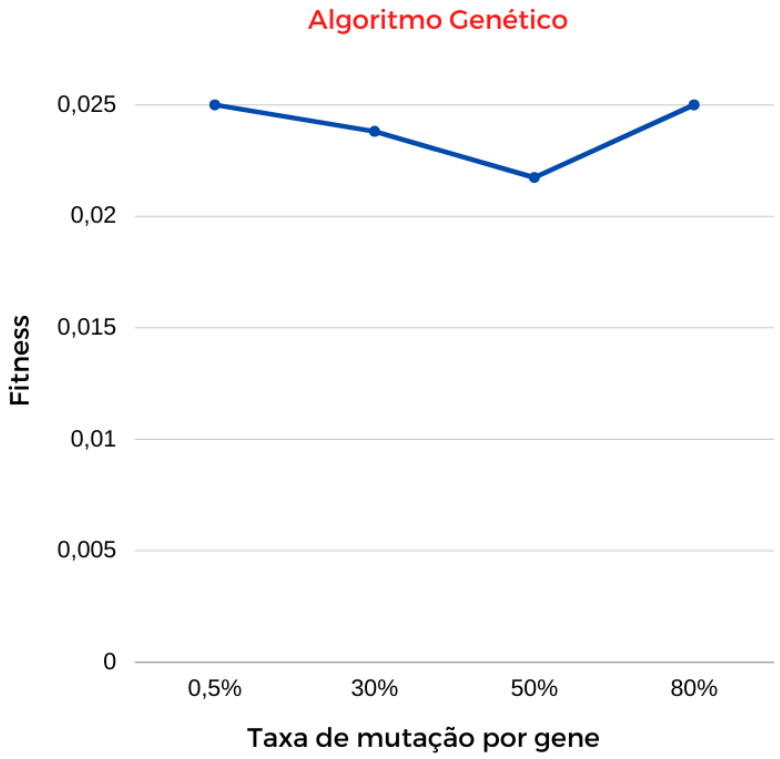

Figura 2. O valor do fitness através da taxa de mutação por gene

Na figura 2, o tamanho da população é o valor dez, seu ponto de parada é na décima população e a sua taxa de mutação cromossômica está no valor de dez por cento . À medida que o valor da taxa de mutação sofre ascensão, o valor do fitness também é alterado. Visto que a taxa de mutação é utilizada para aumentar a diversidade genética numa população com o objetivo de se aproximar da solução ótima, a taxa de mutação é uma variável probabilística, sua ascensão ou descensão pode aumentar ou diminuir a qualidade do fitness, como ocorreu quando a taxa de mutação por gene estava a oitenta por cento quando atingiu uma boa solução de acordo com a imagem.

5. Conclusão

O objetivo desse trabalho foi encontrar uma solução que atendesse as demandas do nosso atual contexto e desenvolver um algoritmo que fosse menos custoso e mais eficiente. E de acordo com os dados analisados através dos resultados anteriormente, o objetivo foi alcançado de forma plena.

As conclusões que podemos definir a partir dos resultados são: Um algoritmo genético funciona de forma eficiente, rápida e pouco custosa, ao contrário do algoritmo de força bruta, que somente é bem utilizável com um pequeno número de pontos de delivery. O algoritmo genético tem a vantagem de não causar uma explosão combinatória e obter uma boa solução em um curto período de tempo.

Dessa forma, posteriormente, nossa finalidade será tornar esse algoritmo genético mais eficiente através de atualizações periódicas de acordo com nossos estudos sobre o tema e a linguagem Python.

Referências Bibliográficas

[1] SILVEIRA, J.f. Porto da. Problema do Caixeiro Viajante. 2000. Disponível em: :<http://www.mat.ufrgs.br/~portosil/caixeiro.html>.

[2] MATAI, R.; SINGH. S. P.; MITTAL, M. L; Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches, *Traveling Salesman Problem, Theory and Applications*, InTech, Croatia, 2010.

[3] HOLLAND, J. H. Genetic Algorithms. Scientific American. 1992.