Universidade Federal Rural de Pernambuco

Departamento de Estatística e Informática

Bacharelado em Sistemas de Informação

Flyfood & algoritmos

José Lucas Cabral Braga

Recife

Fevereiro de 2022

Dedico esse trabalho a todas as que me acompanharam e ajudaram diretamente e indiretamente e ao meu instrutor durante essa jornada.

Resumo

O projeto Flyfood veio com a necessidade de encontrar um novo caminho para as entregas no nosso atual mundo em que temos uma grande demanda e o trabalho humano se valorizou, seu principal objetivo é alcançar uma forma onde que drones consigam fazer entregas de maneira totalmente autônoma, otimizando suas rotas e economizando bateria para completar seu percurso, contudo, se viu um desafio quando a maneira de encontrar a melhor rota se mostrou, foi usado o problema do caixeiro viajante para trabalhar por cima, visto que de inúmeras alternativas, uma deveria ser escolhida e a melhor possível e para isso foi usado o algoritmo de força bruta. Com isso percebemos que ele pode ser usado para poucos pontos de entrega, porém ao decorrer e irem aumentando, se torna inviável por seu tempo de processamento. Por fim foi visto que esse algoritmo se torna lento com muitos pontos de entrega, resultando em maior tempo ou até mesmo em um tempo indeterminado para a máquina conseguir calcular as trajetórias, contudo o código verifica todas as possibilidades e reconhece a melhor entre elas.

Palavras-chave: Algoritmo determinístico, algoritmo não-determinístico, algoritmo de força bruta, problema do caixeiro viajante, problemas de otimização, tempo polinomial, tempo exponencial.

1. Introdução

1.1 Apresentação e Motivação

Em nossa atual realidade, podemos ver que as entregas de delivery estão cada vez mais demoradas e com problemas com sua entrega, seja demorando bastante ou então chegando muitas vezes fria ou o entregador se perdendo por falta de conhecimento do local, além do alto custo para manter essas entregas. Então como proposta para suprir essa demanda e dar mais um passo afrente, uma nova ideia surgiu de criar um segmento de empresa de entregas com drones, na qual sejam programados para seguir uma rota e conseguir entregar com facilidade evitando o trânsito e reduzindo o custo dessa atividade, assim como facilitando para os usuários e os consumidores.

1.2 Formulação do problema

Dentre nossas primeiras configurações, vemos que o Drone tem capacidade limitada de bateria, mas para que ele funcione efetivamente, precisamos condicionar ele a um caminho que seja o melhor entre todos os outros em termos de conseguir alcançar todas as residências e retornar, efetivando o seu melhor desempenho. Sabendo que ele parte de R, que é o estabelecimento comercial, ele deve passar por todas as residências que fizeram seus pedidos, sejam A,B,C e D. Dessa forma temos que escolher o melhor trajeto e para isso usamos a lógica em que F(x) são a quantidade de trajetos e podemos solidificar por coordenadas as localizações das casas.

0	0	0	0	D
0	A	0	0	0
0	0	0	0	C
R	0	B	0	0

Com isso temos todas as localizações em seus determinados locais, porém para alcançá-las temos que passar por outros locais, sendo assim F(x) se tornará vários caminhos

$$F(x) = (x - 1)! = ArgMIN$$

Visto que o restaurante é fixo, sempre teremos outras 4 opções para alcançar e vamos encontrar o caminho mais curto, então logo após teremos apenas 3 opções, novamente vamos ver qual dos três caminhos é o mais curto, tendo dessa vez apenas duas opções e para finalizar teremos apenas mais um caminho que será o de ida e volta.

1.3 Objetivos

Melhorar e revolucionar o método do delivery evitando trânsito e problemas como trocas de pedidos e entregadores perdidos e gastos desnecessários, será criar e implementar uma solução em drones aéreos para conseguirem trabalhar de maneira otimizada para evitar trânsito congestionado e acelerando a entrega pela melhor rota possível.

1.4 Organização do trabalho

Inicialmente veremos a localização principal da ida e da volta, assim de certa forma precisamos também ver os locais e também a capacidade da bateria do drone para assim conseguir calcular a melhor rota possível e nesse trabalho mostraremos, assim depois podemos se situar, primeiro fazendo toda o tráfego referencial, mostrando em que podemos se localizar a partir de um mapa de matrizes, assim como retificar o código de atuação do drone, colocando sobre a linguagem Python, demonstrar como vai ser as possibilidades dos experimentos e as rotas alcançadas e por fim ver se é viável conseguir fazer entregas com o drone. Durante a primeira seção teremos a introdução inteira do trabalho, mostrando suas partes como motivação e objetivos, após isso indo para sua segunda seção teremos os conceitos nos quais trabalhamos e as técnicas usadas, assim como gastos. Na terceira estarão trabalhos que possam complementar ou os demais que tiveram ideias semelhantes ou parecidas para integrar informações. Na quarta seção teremos toda a metodologia, assim como linguagem trabalhada, formas que foram feitas elaboradas os códigos baseando-se nas matrizes apresentadas e a sua atuação dentro do drone. Na quinta seção será visto como foi a natureza que o código foi feito, em quanto tempo ele foi capaz de atuar, por quanto tempo e o seu peso, assim como logo após o seu uso e a sua forma que agiu dentro do drone. Na sexta, todos os resultados foram vistos dentro dos experimentos e as suas modificações aparentes, de certa forma que consiga evoluir com o projeto. E na última e sétima seção teremos todos os resultados finalizados até agora, como tempo de duração com determinado código e seu desempenho.

2. Referencial Teórico

Nessa seção serão apresentadas as motivações de criação do Flyfood, algumas noções de complexidade e sobre Big O e o pior cenário.

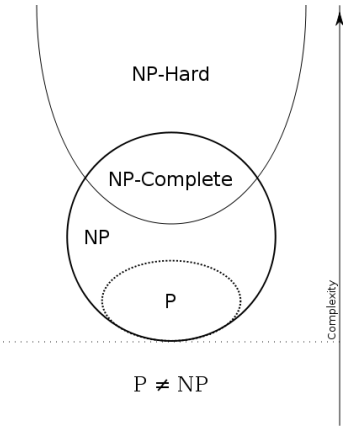
2.1 Flyfood e suas motivações

"*Veículos não tripulados estão cada vez mais presentes no cotidiano das empresas e das pessoas, pois esse tipo de veículo está de forma crescente desempenhando atividades*

que anteriormente eram apenas executadas por seres humanos." [1] assim como visto, drones estão cada vez mais capacitados para fazerem trabalhos humanos, consequentemente durante o tempo e as modificações do trânsito se tornaram visíveis, fazendo com que todo um complexo de trabalho começasse a ter seus problemas, contudo: "*Com o avanço da tecnologia, novas formas de transporte vêm se tornando cada vez mais eficazes para os processos logísticos das empresas, com isso, elas vêm estudando uma forma de implementar os drones, de forma tornar suas entregas mais rápidas e seguras para seus clientes finais.*" [2] Com isso aí entra o projeto Flyfood uma forma autônoma de entrega com a livre movimentação dos drones pelo espaço aéreo, evitando problemas da sociedade atual. "*No entanto, para que tais serviços avançados para cidades inteligentes se tornem realidade, é necessário garantir a segurança nas entregas.*" [3] Após anos, várias modificações na nossa sociedade foram feitas, como o aumento de valores no trabalho humano, assim como a redução dos custos para produção dos drones.

2.2 Complexidade de problemas de classes

"*A complexidade de um problema computacional é o consumo de tempo do melhor algoritmo possível para o problema. É preciso lembrar que para muitos problemas o melhor algoritmo conhecido está longe de ser o melhor algoritmo possível. Podemos dizer, informalmente, que a complexidade de um problema é o tempo (como função do tamanho das instâncias) absolutamente indispensável para resolver o problema.*" [6] O problema de otimização de dados é justamente o tratamento que ele deve receber e o modo em qual seu algoritmo foi ajustado, sendo assim os seus modelos influenciam diretamente no resultado, levando em consideração o tempo e o consumo de dados. Portanto é visto que as classes de problemas podem ser diferentes que classificam os problemas de maneiras ao seu tempo de resposta e pela resolução de seu algoritmo e como um polinômio e assimilando com os dados e o tempo do programa. De modelos de classes temos a: P, NP e NP-Completo.



Algoritmo determinístico é o que sempre produz o mesmo resultado dadas determinadas entradas de dados.

Algoritmo não determinístico é aquele que pode produzir resultados diferentes mesmo com os mesmos dados

2.2.1 Classe P

A classe P é a classe que deve ser o tempo polinomial determinístico que são resolvidos facilmente de maneira determinística, ou seja problemas P são solúveis para computadores reais.

2.2.2 Classe NP

A classe NP é a classe de problemas que são solucionados por um algoritmo não determinístico, podendo ter diferentes saídas para a mesma entrada, também dependendo do tamanho da entrada, pode ter que levar um tempo exponencial para ter a solução de determinado problema.

2.2.3 Classe NP-Completo

A classe dos problemas NP, possuem uma derivação dos problemas NP-Completos sendo sua subclasse de modo que todo problema NP se pode reduzir, com uma redução do tempo. Pode-se dizer que os problemas NP-Completos são problemas mais difíceis de solucionar do que os problemas NP. Nessa classe os problemas são intratáveis, ou seja, demandam uma grande quantidade de tempo para sua solução e dependem do tamanho da entrada e na maioria dos casos de sua solução usam o tempo exponencial.

2.3 BIG O

A notação Big O é de importância universal e origina-se do fato de descrever a eficiência do algoritmo escrito independente da linguagem de programação. Tendo diferentes tipos de complexidades do Big O, como $O(n)$, $O(2n)$, $O(n^2)$, $O(\log n)$, entre outras. Essa notação pode descrever a complexidade de uma seção pelo seu tempo de execução, sendo assim o Big O descreve o tempo em execução em seu pior caso.

2.3.1 Pior caso

Pior caso é quando o algoritmo normalmente excede até o esperando do consumo de dados, por exemplo a memória necessária para funcionar um loop que gere infinitas matrizes e busque uma específica, enquanto outro só irá gerar uma matriz específica, ou até mesmo no problema em que atuamos, onde trabalhe com fatorial, porém existem números fatoriais que excedem os limites e acabe gerando problemas de dados, se tornando algoritmos intratáveis.

2.4 Problema do Caixeiro Viajante

O problema do Flyfood está relacionado com a geração de caminhos, assim como a teoria dos grafos, de certa forma está ligado a PCV de maneira que sua

compreensão e seus caminhos sigam um padrão, contudo existindo diversos caminhos e apenas um que consiga tirar o máximo de vantagem por desempenho, contudo a forma que ele seja encontrado é a maneira mais bruta para ter certeza que o melhor caminho entre todos foi alcançado. Esse problema se torna de maneira um problema de otimização combinatória de forma que vemos a maneira específica para achar o melhor método para se guiar entre as heurísticas, durante o problema do Flyfood se torna um problema tratável por seu baixo número de locais, porém se aumentar logo ele se tornará um problema intratável e se tornando uma classe de problema chamada NP-Difícil.

2.5 Algoritmo de tempo polinomial e exponencial

Na computação a importância do uso do tempo na execução de algoritmos pode ser entendida como categórica de modo que uma função complexidade de tempo de um algoritmo expressa o tempo máximo preciso, isso é dentre as possíveis entradas que podem variar de acordo com o tamanho e dos casos dos problemas.

O algoritmo de tempo polinomial é aquela cujo função de complexidade é $O(p(n))$ para alguma função polinomial $p(n)$, com n sendo o comprimento da entrada. [8]

O Algoritmo de Tempo Exponencial é aquele cuja função complexidade de tempo não pode ser limitada por um polinômio. Por exemplo do caixeiro viajante que estamos usando. [8]

3. Trabalhos relacionados

Durante esta seção, teremos referenciais e trabalhos relacionados ou semelhantes a temática, ou que deram início a procuração do flyfood como uma maneira de implementar suporte de entregas e logística.

3.1 Entrega por drones nos dias atuais

Podemos ver que desde antes, tínhamos o anseio por uma entrega com grande mobilidade e baixo custo, assim como usando drones não tripulados com maior facilidade para alcançar os seus objetivos, sendo mais seguro e também econômico, visando uma forma de efetivar o alcance e as formas que podem ser melhoradas, fazem comparações com programas de entrega por drones já existentes, mostrando que apenas um pequeno percentual mantém esse tipo de trabalho. E visto que os maiores problemas são com a burocracia brasileira para implementar os drones no nosso espaço aéreo se torna algo mais custoso pelas nossas taxas, contudo ainda tem alguns que defendem que poderão ser o futuro de nossas entregas, facilitando e acelerando processos. [2]

3.2 Otimização da entrega de encomendas por drones

Visto do ramo logístico, os drones poderiam ser uma solução para problemas de engarrafamentos e do modal rodoviário, é perceptível que eles seriam uma evolução de entregas pequenas, junto com eles e sua pequena taxa de bateria acabavam contribuindo com o meio ambiente por serem duráveis e serem elétricos, reduzindo a emissão de gases derivados da queima de combustível, contudo foi visto que sua maior desvantagem relacionado aos caminhões é a sua capacidade de não comportar grandes cargas. [4]

3.3 Desenvolvimento De Protótipo De Veículo Aéreo Não Tripulado De Baixo Custo E Integração Com Ferramenta Educacional Para Treinamento De Vôo

Os drones aéreos ganharam grande espaço durante a modernização, como em agricultura, construções civis, meio ambiente, mineração entre outros e com o aumento dessa tecnologia percebeu que deveriam diminuir os gastos para sua produção e ter maior facilidade para configurar ele para exercer sua função, com isso foi projetado interfaces para auxiliar aqueles que estavam moldando um novo drone quadri hélices e começar a treinar vôos em um simulador. [5]

Visto que inúmeros trabalhos estão no ramo de logística para conseguir entregar assim como modelagem e criação, porém é visto que em nossa atual realidade temos que aproveitar os drones e a sensibilização da situação para baixar os custos das taxas aéreas Brasileiras e evitar o trânsito caótico das grandes cidades, assim como auxiliar o meio ambiente com a redução de Co^2, fazendo entregas da melhor forma possível e com mais eficiência sem ter problemas com a manutenção dos drones.

3.4 EXPERIMENTOS COMPUTACIONAIS COM HEURÍSTICAS DE MELHORIAS PARA O PROBLEMA DO CAIXEIRO VIAJANTE

"O Problema do Caixeiro Viajante – PCV (do inglês Traveling Salesman Problem - TSP) é um dos problemas mais estudados em otimização combinatória (Laporte, 1992). Apesar da sua definição singela, o PCV representa, até hoje, um dos desafios da Pesquisa Operacional" [7].Usado no flyfood atualmente para representar o grande número de caminhos que podem ser tomados, contudo apenas um é o melhor e menor caminho que pode solucionar esse problema.

4. Metodologia

Durante essa sessão, teremos os pseudocódigos desenvolvidos como base para implementação da solução do Flyfood sendo por base do problema do caixeiro viajante assim como a formação e implementação, juntamente com ele a solução desenvolvida em Python 3

4.1 Brute force

A primeira solução foi desenvolvida visando o melhor resultado, porém usando o método do algoritmo de força bruta, sendo ele um algoritmo exaustivo já que ele vai tentar coletar todas as informações, ou seja permutar várias rotas diferentes e dentre elas escolher a melhor rota, ou seja a menor rota possível que transite entre todos os pontos e retorne para o ponto inicial.

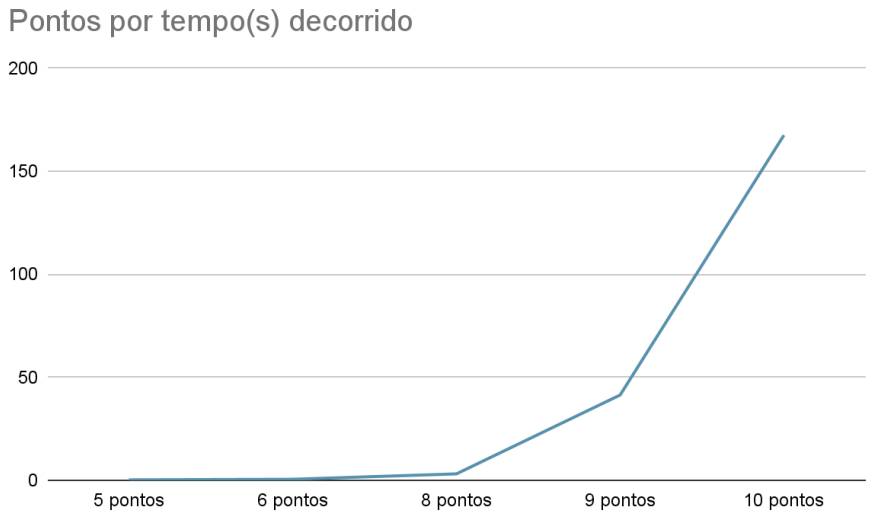
```
     Entrada.split() #Onde deve conter as dimensões de uma matriz
     #Todas as listas que serão ocupadas durante a execução do
programa
     Declarar listas do programa

     #função recursiva para definir o melhor caminho

Função Recursiva melhor_caminho(combinações, coordenadas,
listaVazia)
     # a lista vazia irá armazenar a melhor solução
     menor_rota = 0
     i = 0
     j = 0
     temp = 0
     loop_for c in alcance(0,len(combinações)):
          while (coordenadas[i][0] != combinações[c][d]):
               i += 1
               #Garantir o funcionamento das coordenadas
          while (coordenadas[j][0] != combinações[c][d+1]):
               j += 1
          a = coordenadas[i][1]
          b = coordenadas[j][1]
          x = coordenadas[i][2]
          y = coordenadas[j][2]
          temp = temp + (abs(a-b) + abs(x-y))
          i = j = 0
     se(menor_rota  == 0):
          lista Vazia = combinações[c]
          menor_rota = temp
     senão(temp < menor_rota ):
          menor rota = temp
          lista Vazia = combinações[c]
     temp = 0
```

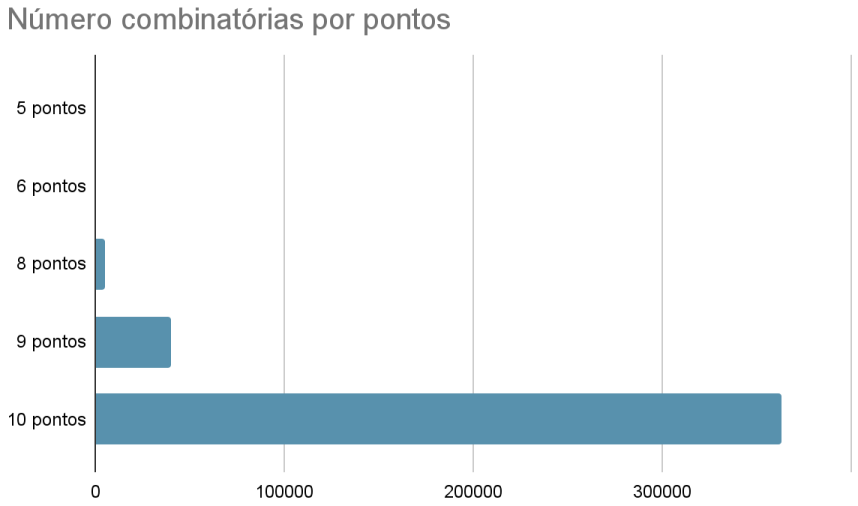
```
imprima(Lista Vazia, combinações e quantidade de combinações)
#Função recursiva para gerar as permutações
Função recursiva permutações(lista, r = None):
     Montar tuplas para
     n = len(pool)
     r = n se r is None senão r
     se r > n:
          retorne
     tabela = list(alcance(n))
     rotação = list(alcance(n, n-r, -1))

     yield tuple(pool[i] for i in tabela [:r])
     while n:
          for i in reversed(alcance(r)):
               rotação [i] -= 1
               se rotação [i] == 0:
                    tabela [i:] = tabela [i+1:] + tabela
[i:i+1]
               rotação [i] = n - i
          else:
               j = rotação [i]
                    tabela [i], tabela [-j] = tabela [-j],
tabela [i]
                    yield tuple(pool[i] for i in tabela
[:r])
                    quebre
               else:
                    retorne
for c in alcance(0, inteiro(dimensoes_matriz[0])):
     matriz.adicione([])
     for d in alcance(0,inteiro(dimensoes_matriz[1])):
          matriz[c].adicione(0)
while True:
     pontos = entradaModeloPlanoCartesiano.split
# Ponto Eixo X e Eixo Y
     if pontos:
          nome = pontos[0].upper()
          locais.adicione(nome)
          x = inteiro(pontos[1])
          y = inteiro(pontos[2])
          matriz[x-1][y-1] = nome
     else:
          quebre
pontoDeInicio = entradaDeString.upper
removerDe.locais(pontoDeInicio)
```

```
arranjo = list(permutacoes(locais, len(locais)))
for item in arranjo:
     melhor_rota.adicione(list(item))
for item in melhor_rota :
     item.adicione(start)
     item.insira(0,start)

for c in alcance(0,len(matriz)):
     for d in alcance(0,len(matriz[c])):
          for e in alcance(0,len(melhor_rota )):
               for f in alcance(0,len(melhor_rota [e])):
                    se len(rotas) == 0:
                         se  matriz[c][d]  ==  melhor_rota
[e][f]:
                    rotas.adicione([matriz[c][d],c+1,d+1])
               else:
                    se (matriz[c][d] == melhor_rota [e][f])
and ([matriz[c][d],c+1,d+1] not in rotas):
                    rotas.adicione([matriz[c][d],c+1,d+1])
#Por fim, você chama as funções recursivas
melhor_caminho(melhor_rota , rotas, comp)
```

5. Experimentos

Durante os experimentos, foi capaz de perceber a mudança a cada ponto adicionado, seja de segundos até mais tempo, além da quantidade de combinações crescendo. De certa forma quando a entrada era uma matriz com cinco pontos, seu tempo era relativamente baixo conseguindo trabalhar da melhor forma possível mas ao decorrer dos pontos serem adicionados, o seu tempo de execução do algoritmo crescia junto com a quantidade de combinações.





6. Resultados

Dentre o termo usado, assim como o algoritmo de força bruta para relacionar o problema do Flyfood de entregas por drones, foi visto que também se torna usável quando vai até 5 pontos, passando disso já se torna inviável começando a apresentar inúmeros problemas, entre eles, fazendo o tempo ser bem mais do que deveria ser, começando a ter problemas em sua otimização, visto que isso tirando a velocidade de programa que deveria entregar isso a pronto momento.

Tamanho da matriz	Número de pontos	Tempo (Segundos)	Combinações geradas
4 x 5	5 pontos	0.21s	24
4 x 5	6 pontos	0.50s	120
5 x 6	8 pontos	3.11s	5040
5 x 6	9 pontos	41.4s	40320
6 x 7	10 pontos	167.5s	362880
6 x 7	12 pontos	Undefined	Undefined

7. Conclusão

Com o decorrer do trabalho, foi perceptível que nossa empresa Flyfood atende uma pequena área de normalmente um bairro com uma pequena quantidade de clientes, onde todos são atendidos efetivamente no menor tempo possível de entrega. Ao achar a lógica do caixeiro viajante temos que ver as inúmeras rotas permutadas, são várias possibilidades de caminhos e rotas a serem tomadas, essa perspectiva leva em consideração que dentre todas sempre vai haver um caminho. Dessa forma o método de brute force sendo um algoritmo exaustivo acaba chegando ao seu limite quando o número de pontos de entrega começa a crescer, quando começa a aumentar exponencialmente começa a se tornar inviável, contudo para pequenos números ele ainda pode ser o ideal conseguindo atender todos no menor espaço de tempo possível, mas é visível por conta de ser um problema categorizado como um NP-Difícil.

Referências Bibliográficas

1. MEDEIROS NETO, Manoel Pedro de. Veículos aéreos não tripulados e sistema de entrega: estudo, desenvolvimento e testes. 2016. 102f. Dissertação (Mestrado em Sistemas e Computação) - Centro de Ciências Exatas e da Terra, Universidade Federal do Rio Grande do Norte, Natal, 2016.
2. DAS GRAÇAS CARVALHO, Edmara et al. ENTREGA POR MEIO DE DRONES NOS DIAS ATUAIS, 2021.
3. DE OLIVEIRA, Fabíola MC; BITTENCOURT, Luiz F.; KAMIENSKI, Carlos A. Prevenção de Colisões em Serviços de Entregas por Drones em Cidades Inteligentes. In: Anais do V Workshop de Computação Urbana. SBC, 2021. p. 182-195.
4. BARBOSA, João Pedro Moutinho Alves. Otimização da entrega de encomendas por drones. 2020. Tese de Doutorado.
5. SHIBATA YAHAGUIBASHI, Henrique. Desenvolvimento De Protótipo De Veículo Aéreo Não Tripulado De Baixo Custo E Integração Com Ferramenta Educacional Para Treinamento De Voo. 2019.
6. FEOFILOFF, Paulo, atualizado em 2021, acessado em março de 2022 www.ime.usp.br/~pf/analise_de_algoritmos/aulas/NPcompleto.html
7. DA CUNHA, Claudio Barbieri; DE OLIVEIRA BONASSER, Ulisses; ABRAHÃO, Fernando Teixeira Mendes. Experimentos computacionais com heurísticas de melhorias para o problema do caixeiro viajante. In: **XVI Congresso da Anpet**. 2002.
8. LOUREIRO, Antonio Alfredo Ferreira. Teoria de Complexidade. UFMG, 2008. acessado em abril de 2022. https://homepages.dcc.ufmg.br/~loureiro/alg/091/paa_Complexidade.pdf
9. Cabral Braga, José Lucas. Flyfood e Otimização. Github, 2022. Disponível em: https://github.com/zeth-l/Flyfood-e-Otimizacao . Acesso em abril de 2022.