Universidade Federal Rural de Pernambuco

Departamento de Estatística e Informática

Bacharelado em Sistemas de Informação

Flyfood: Uma abordagem acerca de algoritmos de roteamento aplicada a logística

Carlos Vinícius Martins da Silva

Recife

Outubro de 2021

Resumo

No seu resumo descreva sucintamente o contexto do trabalho e o que foi feito. Os tópicos a serem abordados são: contexto, objetivo, método, resultados, conclusão. Você pode escrever o resumo seguindo a estrutura a seguir e depois juntar tudo em um único parágrafo.

Contexto:

Objetivo:

Método:

Resultados:

Conclusão:

Palavras-chave: XXXX

1. Introdução

1.1 Apresentação e Motivação

O modelo de vendas utilizando entregas a domicílio, vem se popularizando rapidamente ao redor de todo mundo, até mesmo em uma jovem república como o Brasil podemos observar sinais do alastramento desse tipo de serviço, muito potencializado pelo surgimento da pandemia do novo coronavírus. Sabendo disto, este projeto acadêmico parte de um cenário fictício em que estaríamos no ano de 2025, onde o trânsito estaria caótico, e as empresas de delivery já não conseguiriam fazer entregas em tempo aceitável. Além disso, os custos com entregadores estaria muito alto devido a grande oferta de empregos. Então um ex-aluno do curso de bacharelado de sistemas de informação teria a ideia de criar uma empresa chamada FlyFood que teria como objetivo fazer entregas utilizando drones. Pela descrição do cenário de criação do nosso caso de pesquisa, podemos concluir que: o principal motivo para a criação da empresa seria a falta de velocidade nas entregas feitas convencionalmente. Isto nos leva a fazer o seguinte questionamento: como escolher a melhor rota de entrega visto que o entregador seria um drone?.

Devido à falta de um condutor humano, o drone, dado as coordenadas relacionadas aos pontos de entrega, deveria definir a melhor rota para realizar todas as entregas, e ao fim voltar ao ponto de partida. Um ponto a ser questionado seria: por que é importante que o drone escolha a melhor rota de entrega?. Para responder isso voltaremos ao principal motivo de criação da empresa: a velocidade de realização das entregas. Além disso, podemos citar mais um motivo para a formulação de um novo algoritmo: a economia de combustível fóssil, elétrico ou de qualquer tipo que este drone utilizaria.

1.2 Formulação do problema

O problema do FlyFood consiste em uma matriz de entrada, com *L* linhas e *M* colunas, que seria uma representação dos pontos da cidade. Essa matriz conteria, o ponto de partida determinado pela letra *R*, os *N* pontos de entrega, que seriam representados por coordenadas dentro dessa matriz, tal que seria um ponto com coordenadas (*x*,*y*), e o restante dos pontos seriam os pontos que o drone poderia utilizar para se locomover até o destino das entregas. O drone só poderia se locomover de forma vertical e horizontal pelos pontos da matriz.

Tendo nosso ambiente formado, e seja *i* *a* posição do drone, podemos descrever o evento do drone se mover uma posição como i = i+1. O valor de *i* ao alcançar o ponto de entrega adjacente será o custo da solução entre dois pontos, mas como isso poderia ser descrito de forma sintética?. É nessa parte do problema que descreveremos a equação da distância entre dois pontos. Anteriormente dissemos que um ponto é representado por coordenadas, tal que um ponto *N* seria uma posição (*x*, *y*), imaginemos dois pontos de entrega, sendo eles *A* e *B*, onde *A* seria (x_1,y_1) e B seria (x_2,y_2). Como mencionado anteriormente nosso drone não poderá se locomover pelas diagonais da matriz, devido a este fato utilizaremos em nossa abordagem, uma das geometrias não euclidianas, conhecida como geometria do táxi. Denominaremos a distância entre dois pontos como *D*, portanto, a distância entre dois pontos pode ser determinada por $\boldsymbol{D = |x_2-x_1| + |y_2-y_1|}$. Elucidada a questão de como encontrar a distância entre dois pontos, fica mais fácil visualizarmos nosso próximo questionamento: qual seria a equação do custo da solução de um percurso completo?. Quando dizemos percurso completo nos referimos ao ato de realizar todas as entregas e voltar ao ponto de partida, ou seja, o somatório do custo de entrega de todos os pontos adjacentes. Então seja o custo total do percurso D_t , podemos dizer que $\boldsymbol{D_t = \sum_{k=0}^{n} D}$.

Definidas as equações de distância entre dois pontos e de custo de um percurso completo, cabe a nós acharmos a solução para o problema. Para isso iremos relembrar o motivo pelo qual foi criada a empresa: realizar entregas de forma mais rápida através do uso de drones. Dito isso, a solução para o problema seria achar um circuito, tal qual o custo de solução fosse o menor possível, seja essa solução f(x), podemos dizer que: f(x) = argmin D_t .

1.3 Objetivos

O objetivo do trabalho é conceber uma solução para o roteamento dos drones, fazendo com que percorram a menor distância possível.

O mesmo trabalho, tem como objetivos específicos:

- Analisar o custo dos possíveis algoritmos utilizados no problema de roteamento.
- Construir a habilidade de solucionar problemas reais utilizando algoritmos.
- Produzir material que possa servir de referencial teórico para futuras pesquisas acadêmicas.

1.4 Organização do trabalho

Como o trabalho está organizado?

2. Referencial Teórico

Iremos dar início ao nosso estudo sobre tempo computacional e análise de algoritmos. Abordaremos conceitos aos quais o nosso trabalho está relacionado. Falaremos sobre classes de problema, introduzindo as classes P, NP e NP completo, além de, redutibilidade, algoritmos de verificação, algoritmos determinísticos e não determinísticos. É importante destacar que o custo de tempo que um algoritmo leva para resolver um problema é calculado pelo número de operações básicas que ele executa. Então, para tratarmos de tais assuntos, é de fundamental importância o conceito de problemas resolvíveis em tempo polinomial, também abordaremos sobre o tema.

2.1 Tempo polinomial

Como citamos anteriormente, o custo de tempo que um algoritmo leva para resolver um problema é calculado pelo número de operações básicas que ele executa. Sabendo disso, iremos introduzir o conceito de tempo polinomial para a resolução de um problema. De acordo com Cormen et al(2009) é pertinente afirmar que os problemas de NP-completude são tratáveis por razões filosóficas, e não matemáticas. Segundo o próprio Cormen et al., (2009, p.768) “embora seja razoável considerar como intratável um problema que exige o tempo $\Theta(n^{100})$, um número bem pequeno de problemas práticos exige tempo da ordem de um polinômio de grau tão alto”. Por meio dessas duas citações, o autor deseja nos informar que: embora possa existir um problema com custo na ordem de n^{100} , o que seria um tempo impraticável por resultar em um número grande demais, geralmente este não se aplica na prática, pois, problemas práticos exigem um tempo muito menor. O que nos leva mais uma vez a consultar Cormen et al (2009, p.768) “Ainda que o melhor algoritmo atual para um problema tenha um tempo de execução de $\Theta(n^{100})$ é provável que um algoritmo com um tempo de execução muito melhor logo seja descoberto ”. Todas essas citações nos levam a tomar a seguinte conclusão: quando falamos em um problema que é resolvível em tempo polinomial, estamos considerando um tempo que segue um polinômio. Além disso, estamos considerando um polinômio de grau baixo, tal que resulte em um tempo praticável para a resolução do problema.

2.2 Algoritmos de verificação

Daremos início agora ao estudo dos algoritmos de verificação, eles são muito importantes para as definições das classes de problemas que introduziremos mais a frente. Como citamos anteriormente, um problema resolvível em tempo polinomial é aquele em que este tempo de execução segue um polinômio. Mas existem problemas em que a sua solução é resolvível em tempo polinomial, porém, a sua verificação não, e é para isso que servem os algoritmos de verificação. De acordo com Cormen et al (2009, p.775)" Definimos um algoritmo de verificação como um algoritmo de dois argumentos *A*, onde um argumento é uma cadeia de entrada comum *x* e o outro é uma cadeia binária *y* denominada certificado". Então, um algoritmo *A* verifica se , para qualquer entrada *x* que pertença ao problema, existe um certificado *y* que *A* pode usar para provar que qualquer *x* é encontrado em tempo polinomial.

2.3 Algoritmos determinísticos e não determinísticos

Segundo Cormen et al(2009,p.03)"Informalmente, um algoritmo é qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores como saída". Tendo este conceito introdutório em mente, iremos definir os conceitos que nos são pertinentes no momento.

Podemos definir um algoritmo determinístico como uma função matemática. Onde sua entrada de dados seria o domínio da nossa função, e a saída de dados seria o conjunto imagem dessa mesma função. Além disso, podemos definir o algoritmo em si como a lei de formação da função que estamos apresentando para o nosso estudo. Um algoritmo determinístico é exatamente como uma função, não importando a situação, uma saída resultará sempre o mesmo valor ou conjunto de valores dado um determinado valor ou conjunto de valores como entrada. Para ilustrar esta afirmação, façamos um exemplo utilizando uma função onde ***F(x) = 2x+1***. Suponhamos que *x* seja igual a 1, logo somos obrigados a concluir que *F(x)* = 3, desde que *x* seja igual a 1, *F(x)* resultará no valor 3. É importante também destacar que um algoritmo determinístico é um caso particular de um algoritmo não determinístico. Isso nos leva a concluir que podemos transformar um algoritmo determinístico em um algoritmo não determinístico de forma fácil.

Já um algoritmo não determinístico funciona de maneira contrária a um algoritmo determinístico como o próprio nome sugere. Em um algoritmo não determinístico uma saída pode resultar um valor ou conjunto de valores diferentes dado um determinado valor ou conjunto de valores como entrada. Para ilustrarmos isso, voltemos ao exemplo com a função em que ***F(x) = 2x+1***. Em um algoritmo não determinístico, seja *x* igual a 1, *F(x)* pode resultar quaisquer valores mesmo que *x*

sempre seja igual a 1. Diferente do que destacamos anteriormente, um algoritmo não determinístico não pode ser transformado em um algoritmo determinístico de forma tão fácil.

2.4 Classes de problemas

Agora que temos consolidados os conceitos de: algoritmos resolvíveis em tempo polinomial, algoritmos de verificação, algoritmos determinísticos e não determinísticos, podemos enfim introduzir as nossas classes de problemas. São elas as classes P, NP e NP completo esta que também pode ser abordada por alguns autores como NPC.

A classe P segundo Cormen et al (2009, p.765)"consiste nos problemas que podem ser resolvidos em tempo polinomial". Esta afirmação nos leva a considerar que: um problema que pertença a classe P pode ser resolvido em um tempo que segue um polinômio. Além disso, problemas incluídos na classe P são determinísticos, afinal todo polinômio é determinístico seguindo a nossa definição para o que é ser um algoritmo determinístico.

A classe NP de acordo com Cormen et al (2009) é composta por problemas que são verificáveis em tempo polinomial. Além disso, os problemas NP também são não determinísticos. Podemos dizer que $P \subseteq NP$, isso porque se um problema está em P, significa que podemos resolvê-lo sem nem mesmo precisar de uma verificação em tempo polinomial.

Segundo Cormen et al (2009, p.765)"Um problema está na classe NPC (e nos referimos a ele como um problema Np-completo) se ele está em NP e é tão difícil quanto qualquer problema em NP". Ou seja, problemas NP completo são não determinísticos e não podem ser resolvidos em tempo polinomial. Ainda de acordo com Cormen et al (2009) cientistas da computação acreditam que esses problemas sejam intratáveis devido a alta quantidade de problemas desse tipo já estudados, sem que ninguém tenha encontrado uma solução em tempo polinomial para nenhum deles. Até hoje acredita-se que $P \neq NP$, porém, seja um problema NP completo resolvido em tempo polinomial, então poderíamos afirmar que $P = NP$.

2.5 Classificação do Flyfood nas classes de problema.

O Flyfood, que é o nosso caso de estudo, encontra-se classificado como um problema NP hard. Um problema Np hard é aquele que é tão difícil quanto um problema NP completo.

3. Trabalhos relacionados

Os trabalhos relacionados ao seu (assim como concorrentes e complementares)

4. Metodologia

Nessa seção devem ser apresentados os métodos utilizados no trabalho, ferramentas, dados e materiais.

4.1 Subseção do método

XXX.

5. Experimentos

Explicar qual o passo a passo seguido no trabalho.

Quais experimentos foram realizados?

6. Resultados

Explicar quais resultados foram obtidos.

7. Conclusão

Relembrar qual era o objetivo do trabalho.

Quais resultados foram alcançados?

Quais conclusões podemos tirar a partir dos resultados?

Quais os trabalhos futuros?

Referências Bibliográficas

Beraldi, L. C., & Escrivão Filho, E. (2000). Impacto da tecnologia de informação na gestão de pequenas empresas. Revista Ciência da Informação, Brasília, 29(1), 46-50.

Bishop, C. M. (2006). Pattern recognition and machine learning. springer.

BRASIL (2017). Lei Complementar nº 123, de 14 de dezembro de 2006. Institui o Estatuto Nacional da Microempresa e da Empresa de Pequeno Porte. Disponível em: http://www. planalto. gov. br/ccivil_03/leis/LCP/Lcp123. htm. Acesso em 07 dez. 2017

Conselho Regional de Contabilidade do Paraná (2017) . Balanço Patrimonial Disponível em: https://goo.gl/Lb4jzw. Acesso em 08 dez. 2017