Problema do Caixeiro Viajante implementado com o algoritmo Ant System com auxílio de ferramentas Cloud

Caio T. Frauzino, Rômulo C. Delmondes, Salustiano Z. de Faria

Instituto Federal de Educação, Ciência e Tecnologia de Goiás (IFG)
Goiânia – GO – Brasil

oiac23@gmail.com, romulodelmondes@gmail.com, saluzf@gmail.com

Resumo. *Este artigo apresenta a implementação do problema de otimização do Caixeiro Viajante através do algoritmo bioinspirada em inteligência de enxame - Ant System utilizando a linguagem de programação Python. O cenário apresentado é uma hipótese de um problema do mundo real. Através de uma lista de endereços de entrega, gerar a melhor rota para que o entregador consiga com a menor distância possível realizar todas as entregas. Será feita a extração desta lista, transformação com a geocodificação dos endereços e carregamento dos dados para que o algoritmo Ant System determine qual a solução otimizada.*

Abstract. *This article presents the implementation of the Traveling Salesman optimization problem through the bioinspired swarm intelligence Ant System algorithm using Python programming language. The proposed scenario is a hypothesis of a real-world problem. From a list of delivery addresses, generate the best route, i.e., for the delivery person to make all deliveries with the minimum traveling distance. The data will be extracted from this list, and transformed with address geocoding and loaded into the algorithm Ant System which will determine the optimized solution.*

1. Introdução

Este artigo é apresentado como avaliação do módulo 1 da Pós-Graduação Inteligência Artificial Aplicada do Instituto Federal de Educação, Ciência e Tecnologia de Goiás (IFG). Nele apresentaremos os conceitos apresentados nas disciplinas Linguagem de Programação Aplicada, *Cloud Computing* e Inteligência Artificial.

Uma das aplicações do problema de otimização do caixeiro viajante no mundo real é a determinação de rotas de entrega. Utilizaremos o algoritmo *Ant System* para determinar uma rota otimizada para uma lista de endereços. Genericamente, iremos a partir de uma lista de endereços de entrega e determinar qual a rota otimizada para se percorrer os pontos, ou seja, os endereços da lista. A partir dessa lista de endereços, faremos a geocodificação dos endereços, que é a tradução de um endereço lexicográfico para coordenadas geográficas. Com essas coordenadas, que serão as entradas do algoritmo de otimização, determinaremos a rota com o custo otimizado, ou seja, o mais próximo do mínimo possível. Será feito o uso de ferramentas de nuvem da *Amazon Web Services* para execução de alguns dos passos.

2. Fundamentação Teórica

Esta seção apresenta os fundamentos utilizados como base no desenvolvimento do projeto descrito por este artigo. São apresentados os conceitos teóricos e as ferramentas.

2.1 Problemas Combinatórios

Problemas combinatórios ou otimização combinatória é um campo da otimização matemática que busca encontrar um objeto otimizado dentre um conjunto discreto e finito de objetos, onde a solução pode ser discreta ou reduzida à um conjunto discreto [Schrijver, 2001].

O problema do caixeiro viajante (*Traveling Salesman Problem* – TSP) é um problema computacional combinatório que consiste em determinar o menor caminho que percorra as cidades de uma lista apenas uma vez e retorne à cidade de origem. É um dos problemas mais estudados e pode ser estendido à outras aplicações como a gestão de frotas de veículos [Stuart e Norvig, 2021].

2.2 Algoritmos Heurísticos

Os algoritmos heurísticos são técnicas de busca que procuram soluções aproximadas para problemas complexos, quando não é possível encontrar uma solução exata em tempo hábil. Eles são baseados em regras empíricas, conhecimento prévio do problema e tentativa e erro. Os algoritmos heurísticos são amplamente utilizados em áreas como inteligência artificial, otimização, aprendizado de máquina, jogos e robótica [Hillier e Lieberman, 2001].

2.3 Algoritmos Bioinspirados

Os algoritmos bioinspirados são técnicas de otimização que se baseiam em processos biológicos, como a evolução, o comportamento de colônias de insetos e a comunicação entre neurônios. Esses algoritmos são capazes de encontrar soluções para problemas complexos de forma eficiente e rápida [Holland, 1975].

Um exemplo de algoritmo bioinspirado é o algoritmo de otimização por enxame de partículas, que se baseia no comportamento de enxames de animais, como pássaros e peixes [Eberhart e Kennedy, 1995].

Esse algoritmo utiliza uma população de partículas, que se movem pelo espaço em busca da solução ótima. Eles trabalham com uma população de soluções candidatas e usam operadores de movimento para explorar o espaço de busca. Cada partícula é avaliada de acordo com sua aptidão para resolver o problema, e seu movimento é influenciado pela posição das partículas vizinhas. Os indivíduos mais aptos são usados como referência para os outros indivíduos, e o processo é repetido até que uma solução aceitável seja encontrada [Kennedy e Eberhart, 1995].

O algoritmo *Ant System* faz parte da família do método *Ant Colony Optimization*, do campo da inteligência de enxame na Metaheurística e Inteligência Computacional. Originalmente era chamada de *Ant Cycle* e posteriormente passou a ser designado *Ant System*, é a lógica de base para todos os algoritmos *Ant Colony Optimization* [Brownlee, 2012].

O algoritmo é inspirado na busca por alimentos das formigas, especificamente a comunicação por feromônios que são deixados em um caminho entre o formigueiro e uma

fonte de comida. Quando alimento é encontrado, as formigas deixam feromônio pelo ambiente. Se esse caminho for satisfatório, mais feromônio é acrescentado. Com o tempo, esse feromônio perde intensidade. Isso faz com que a região de busca seja restringida a uma região promissora [Brownlee, 2012].

2.4 Linguagem de programação Python

Python é uma linguagem de programação de alto nível de propósito amplo. Seu enfoque principal é a legibilidade do código com auxílio da indentação e suporta múltiplos paradigmas da programação [Kuhlman, 2013].

Surgiu no fim dos anos 1980, criada por Guido van Rossum, e é derivada da linguagem de programação ABC.

2.5 Serviços em nuvem

Os serviços em nuvem são recursos de tecnologia da informação que são hospedados e podem acessados pela internet e contratados sob demanda. Esses serviços incluem infraestrutura, plataformas e softwares que permitem que o usuário desfrute de recursos computacionais poderosos sem a necessidade de ter de adquiri-los ou mantê-los [Amazon Web Services, 2023].

2.6 Amazon Web Services e seus serviços

A Amazon Web Services (AWS) é uma plataforma de serviços em nuvem que oferece uma ampla gama de serviços de computação, armazenamento, banco de dados, análise, inteligência artificial, Internet das Coisas (IoT), segurança e muito mais. [Amazon Web Services, 2021].

O Amazon S3 (Simple Storage Service) é um serviço de armazenamento de objetos que oferece escalabilidade, disponibilidade de dados e segurança. O S3 é projetado para armazenar e recuperar grandes quantidades de dados de forma eficiente e econômica. [Amazon Web Services, 2021].

O AWS Glue é um serviço de extração, transformação e carregamento de dados (Extract, Transform, Load - ETL) totalmente gerenciado que permite que os usuários movam dados facilmente entre diferentes fontes de dados [Amazon Web Services, 2021].

O Amazon SageMaker é um serviço de aprendizado de máquina totalmente gerenciado que permite que os usuários criem, treinem e implantem modelos de aprendizado de máquina em escala [Amazon Web Services, 2021].

3. Cenário apresentado

A partir de uma lista de endereços, simulando uma lista de endereços gerada para entregas, será feita a geocodificação desses endereços, que é a tradução dos endereços textuais em coordenadas geográficas. Com as coordenadas de cada endereço, que serão as entradas do algoritmo de otimização, determinaremos as distâncias entre esses pontos de entrega e será determinada a rota com o custo otimizado, ou seja, o custo mais próximo do mínimo quanto possível.

3.1 Algoritmo de ETL

O algoritmo de ETL fará o tratamento dos dados presentes na lista de endereços. O formato dos dados nessa lista é o "valores separados por vírgula" (*Comma Separated Values* – CSV) e contém quatro colunas com o seguinte cabeçalho: Logradouro/Nome; Bairro/Distrito; Localidade/UF e CEP. Essas são as informações que compõem o endereço textual.

Para a geocodificação será utilizada a biblioteca geopy*,* com o geocodificador Nominatim da ferramenta gratuita OpenStreetMap. O formato de entrada para essa geocodificação é o endereço completo. Portanto, é necessário fazer a concatenação das colunas da tabela antes de fazer o processo de geocodificação.

Para o tratamento dos dados, será utilizada a biblioteca *pandas,* e seus métodos integrados para fazer as manipulações necessárias para obtermos primeiro uma coluna com as colunas "Logradouro/Nome; Bairro/Distrito; Localidade/UF" concatenadas nesta sequência.

Após a concatenação, utilizamos essa nova coluna como entrada do geocodificador. O geocodificador gera um objeto em que sua saída padrão é o endereço textual, e faz-se necessário acessar a propriedade point que indica a coordenada geográfica do ponto. Então, cria-se uma coluna denominada Coordenadas para receber esses novos valores. O formato padrão dessa propriedade é uma tupla de três elementos, com esse último elemento nulo. Precisamos ajustar esse valor para que esse tenha apenas dois elementos, que é o formato de entrada do algoritmo de resolução do problema do caixeiro viajante. Utilizamos o método apply da biblioteca pandas com uma função lambda que remove valores nulos.

Com isso, para a saída desse algoritmo extraímos a série das coordenadas para uma variável e exportamos essa variável para um arquivo .csv.

3.2 Algoritmo de resolução do Problema do Caixeiro Viajante

Primeiramente, definimos os parâmetros do algoritmo Ant System. São estes: Número de iterações, que é a quantidade de vezes em que se busca melhorar a solução; Número de formigas, que é a quantidade de soluções que são geradas a cada iteração, e geralmente é definido igual ao número de pontos; Fator de decaimento do feromônio, que é o fator com o qual se reduz a cada iteração a contribuição do feromônio de cada ponto para a solução; Coeficiente da heurística que é o valor que determina a contribuição da heurística na formação de uma nova solução; Coeficiente de histórico que é o valor que determina a contribuição das soluções anteriores na formação de uma nova solução.

Com isso, passamos esses parâmetros mais a entrada, que são as coordenadas dos pontos, para o início do algoritmo. Este gera uma solução inicial aleatória e é gerada um valor de custo que é resultado da soma das distâncias entre os pontos. Esta distância é calculada através da fórmula de haversine que calcula distâncias em uma esfera, o que para nosso caso se considera uma boa aproximação, com as coordenadas geográficas de entrada.

A partir dessa primeira solução, é gerada a matriz de feromônios. Com isso, são geradas as novas soluções, que são as formigas da metáfora, através dos parâmetros e elementos encontrados até aqui. A cada iteração a matriz de feromônio é atualizada com

o fator de decaimento e modificada com a melhor solução encontrada. Faz-se isso até que se percorra por todas as iterações definidas nos parâmetros do algoritmo.

Foi criada uma função acessória para acompanhamento da execução do algoritmo a cada iteração. Para saída, são gerados uma lista com a sequência dos pontos e um mapa interativo. Para criação desse mapa foi utilizado a biblioteca folium, que gera o mapa com as imagens do OpenStreetMap.

3.3 Arquitetura Cloud

Para a parte de nuvem, foram projetados o uso de alguns elementos da Amazon Web Services.

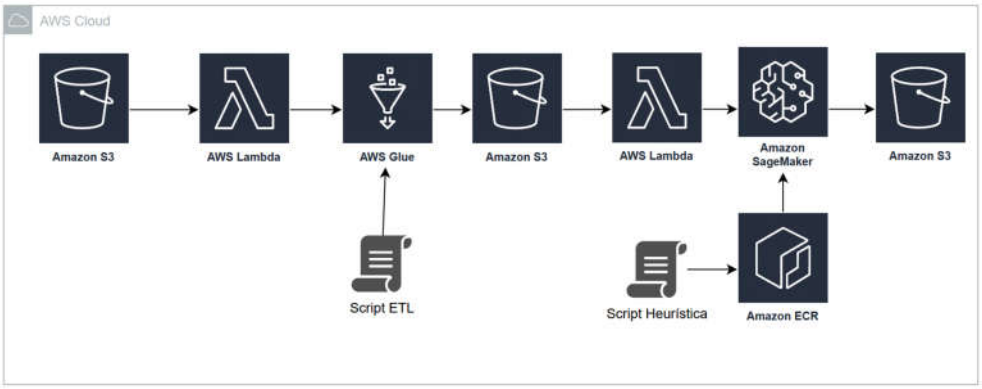

Figura 1. Diagrama de serviços da AWS projetados

Para organização do pipeline, foram determinados os seguintes passos. A lista a ser processada será enviada para um bucket e através de uma função lambda detectando a inclusão de um novo arquivo no bucket acionar um job do AWS Glue ETL. Neste job do Glue ETL será utilizado um script ETL desenvolvido conforme as seções anteriores. Nesse job do Glue será exportado um arquivo csv para uma outra pasta do mesmo bucket. Daí, uma nova função lambda aciona o SageMaker Processing através da função ScriptProcessor com uma imagem Docker armazenada no Amazon ECR que envia um mapa interativo e uma lista com a sequência em que os pontos devem ser visitados.

4. Resultados

Em testes locais, sem o uso das ferramentas de nuvem, os algoritmos desenvolvidos performaram conforme se esperava, retornando caminhos otimizados para as entradas conforme indicadas. Para execução, foi utilizada uma lista com 780 endereços e tomadas amostras do tamanho que se queira, no algoritmo de ETL. Feito o processamento, estes eram utilizados como entrada para o segundo algoritmo, gerando o caminho otimizado e um mapa ilustrativo.

Por conta de sua simplicidade, o algoritmo de busca Ant System tende a ficar enviesado pelas soluções anteriores. Como ele gera novas soluções a partir de soluções anteriores, ele pode ficar preso a mínimos locais. Os outros algoritmos Ant Colony Optimization tratam essa limitação com novos elementos que tentam tirar a solução desses mínimos locais.

Já no desenvolvimento em nuvem, encontrou-se limitação do sistema AWS Academy ou de conhecimento. Não foi possível fazer com que o Glue ETL utilizasse bibliotecas externas dentro dos scripts. Também não foi possível implementar o script de processamento do SageMaker, pois havia a necessidade de criar um Role SageMaker IAM que fosse capaz de enviar a imagem do Docker container para o Amazon ECR [Amazon Web Services, 2023].

A parte cloud implementada ficou limitada ao envio dos dados ao bucket S3 para uma pasta denominada source e através de uma função lambda ativar um job do Glue ETL. Neste job do Glue foi feita apenas a concatenação das colunas e o envio para uma outra pasta denominada processed. Os triggers da função lambda foram diferenciadas pelo prefixo com a pasta de cada trigger. Com isso, a arquitetura implementada ficou com a seguinte forma:

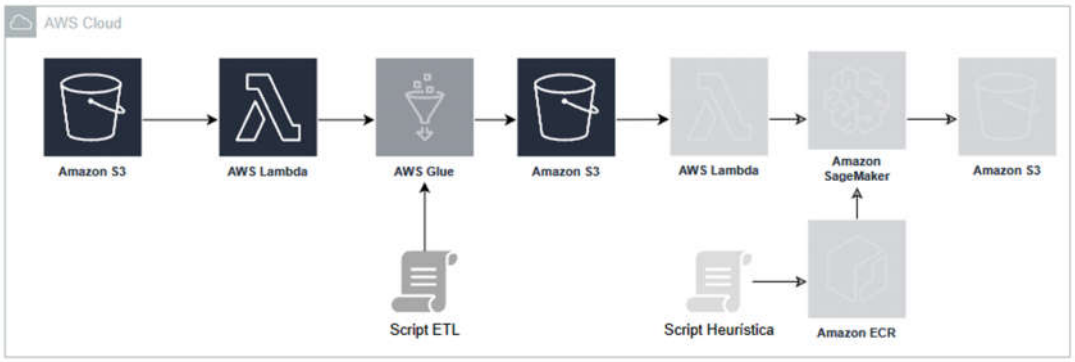

Figura 2. Arquitetura cloud implementada

5. Conclusão

Desta forma, com o que foi apresentado é possível notar que o algoritmo Ant System consegue resolver de forma satisfatória o problema do caixeiro viajante proposto, porém nem sempre este encontrará a melhor rota, visto que este pode ficar preso a mínimos locais devido a sua característica de formar novas respostas a partir apenas de soluções anteriores, ou seja de buscas locais.

Foram apresentados desafios pela arquitetura cloud que não foram superados, e que realmente o que foi apresentado fica aquém das expectativas. Acredita-se que pela necessidade do uso de bibliotecas externas e a falta de familiaridade do PySpark foram pontos que fizeram com que a implementação não funcionasse, além das limitações do AWS Academy.

7. References

Amazon Web Services. (2021). “O que é a AWS?”. <https://aws.amazon.com/pt/what-is-aws/>. Acesso em 18 de outubro de 2023.

Amazon Web Services. (2021). “Amazon S3”. <https://aws.amazon.com/pt/s3/>. Acesso em em 18 de outubro de 2023.

Amazon Web Services. (2021). “AWS Glue”. <https://aws.amazon.com/pt/glue/>. Acesso em 18 de outubro de 2023.

Amazon Web Services. (2021). “Amazon SageMaker”. <https://aws.amazon.com/pt/sagemaker/>. Acesso em 18 de outubro de 2023.

Amazon Web Services. (2023). “O que é a computação em nuvem?”. <https://aws.amazon.com/pt/what-is-cloud-computing/>. Acesso em 18 de outubro de 2023.

Amazon Web Services. (2023). “O que é a computação em nuvem?”. <https://aws.amazon.com/pt/what-is-cloud-computing/>. Acesso em 18 de outubro de 2023.

Amazon Web Services. (2023). “Build Your Own Processing Container (Advanced Scenario)”. <https://docs.aws.amazon.com/sagemaker/latest/dg/build-your-own-processing-container.html/>. Acesso em 20 de outubro de 2023.

Bronwlee, Jason. (2012). “Clever Algorithms: Nature-Inspired Programming Recipes”. Jason Brownlee.

Eberhart, R. C., Kennedy, J. (1995). A new optimizer using particle swarm theory. Proceedings of the sixth international symposium on micro machine and human science, 39-43.

Hillier, F. S., & Lieberman, G. J. (2001). “Introduction to operations research”. McGraw-Hill.

Holland, J. H. (1975). “Adaptation in natural and artificial systems”. University of Michigan Press.

Kennedy, J. e Eberhart, R. (1995). “Particle swarm optimization”. Proceedings of IEEE International Conference on Neural Networks, 4, 1942-1948.

Kuhlman, Dave. (2013). "A Python Book: Beginning Python, Advanced Python, and Python Exercises"

Schrijver, Alexander (2003). Combinatorial Optimization: Polyhedra and Efficiency. Algorithms and Combinatorics. Vol. 24. Springer

Stuart J. Russell e Norvig, Peter. (2021). “Artificial Intelligence: A Modern Approach”. Global Edition-Pearson.