PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS

ESCOLA POLITECNICA

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA SIMULTÂNEA COM DADOS COLETADOS A PARTIR DE APLICATIVOS DE MAPAS.

Jovânio José Galvão Júnior

GOIÂNIA

2022

JOVANIO JOSE GALVAO JUNIOR

PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA SIMULTÂNEA COM DADOS COLETADOS A PARTIR DE APLICATIVOS DE MAPAS.

Trabalho de Conclusão de Curso apresentado à Escola Politécnica, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Alexandre Ribeiro

GOIÂNIA

2022

RESUMO

Neste trabalho foi abordada uma generalização do Problema de Roteamento de Veículos Capacitados (CVRP), chamada Problema de Roteamento de Veículos Com Coleta e Entrega Simultânea (VRPSPD). Nesta variante que generaliza o CVRP é adicionado um conjunto de demandas de coleta, que devem ser atendidas simultaneamente ao conjunto de demandas de entrega. O Objetivo do VRPSPD é determinar rotas que atendam todas as demandas de coleta e entrega para todos os clientes, sendo que todos os veículos devem iniciar no depósito e retornar ao depósito, minimizando a distância total percorrida por todos os veículos. Este trabalho propõe utilizar dados reais coletados a partir de aplicativos de mapas disponíveis e livres. Ao longo do trabalho foi proposto utilizar uma heurística chamada Algoritmo de Otimização por Colônia de Formigas (ACO) para solucionar o VRPSPD, junto ao ACO outras duas abordagens foram utilizadas, um solver chamado Gurobi e uma heurística chamada Busca Tabu (TS), sendo o solver e a heurística TS já existentes na literatura e foram utilizadas para *benchmark*. Os experimentos computacionais consistiram em executar todas as abordagens (soluções propostas) com um conjunto de parâmetros e considerar a melhor solução obtida nessas condições para um conjunto de instâncias criadas a partir dos dados coletados em aplicativos de mapa. O solver Gurobi mostrou-se bem efetivo em instâncias pequenas e até em algumas medias, já em instâncias grandes o solver não conseguiu um bom desempenho com os parâmetros definidos. A TS apresentou-se efetiva em quase todas as instâncias pequenas e médias, já em instâncias maiores não demonstrou um bom desempenho. O ACO mostrou-se melhor na maioria das instâncias, visto que, ficou melhor em tempo para obter a solução e no custo obtido.

ABSTRACT

In this work, a generalization of the Capacitated Vehicle Routing Problem (CVRP) called Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD) was addressed. In this variant that generalizes the CVRP, a set of collection demands is added, which must be met simultaneously with the set of delivery demands. The objective of the VRPSPD is to determine routes that meet all collection and delivery demands for all customers, with all vehicles starting at the depot and returning to the depot, minimizing the total distance traveled by all vehicles. This work proposes the use of real data collected from available and free map applications. Throughout the proposed work was to use a heuristic called Ant Colony Optimization Algorithm (ACO) to solve the VRPSPD, along with the ACO two other approaches were used, a solver called Gurobi and a heuristic called Busca Tabu (TS, being the solver and the TS heuristic already exists in the literature and was used as a benchmark. in map applications. The Gurobi solver proved to be very effective in small instances and even in some medium ones, while in large instances the solver did not achieve a good performance with the parameters and averages, but in larger instances they did not show a good performance. The ACO proved to be better in most instances, that is, it was better in time to obtain the solution and in the cost obtained.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE ALGORITMOS

LISTA DE SIGLAS

OC	Otimização Combinatória
VLSI	projeto Integração em larga escala
VRP	Problema de Roteamento de Veículos
CVRP	Problema de Roteamento de Veículo Capacitado
LR	Logística Reversa
VRPSPD	Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas
TSP	Problema do Caixeiro Viajante
TS	Busca Tabu
TL	lista Tabu

Sumário

1 INTRODUÇÃO

Definir rotas para atender um conjunto de clientes é uma decisão comum a várias empresas, representando uma importante decisão, visto que esta decisão influência nos gastos dessas empresas (OLIVEIRA, 1991).

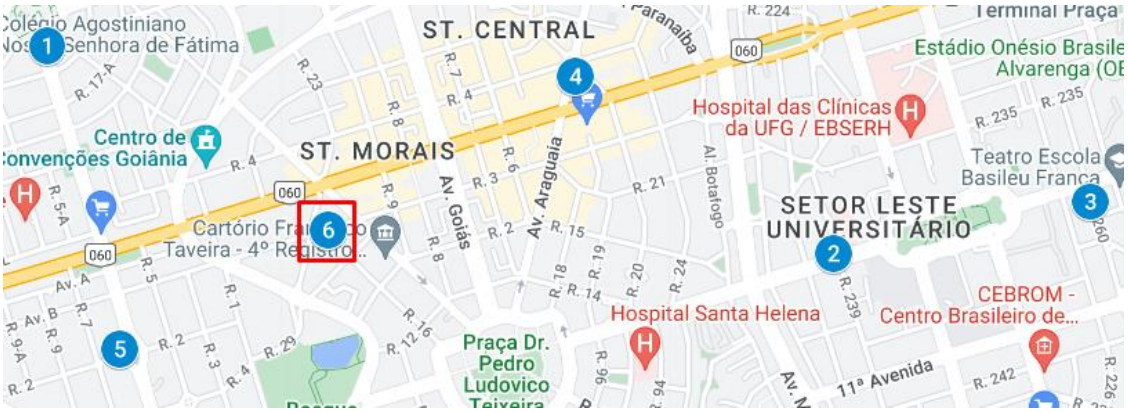
O Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (*VRPSPD*) busca atender um conjunto de clientes que possuem demandas de coleta e entrega definidas. O Objetivo do *VRPSPD* é visitar todo o conjunto de clientes minimizando a distância total percorrida por todos os veículos utilizados, sabendo que todos os veículos partem do mesmo ponto (depósito ou centro de distribuição), respeitando a capacidade máxima de cada veículo, satisfazendo a demanda de coleta simultaneamente a demanda de entrega e todas as demandas de entrega devem ser preparadas no depósito e todas as de coleta devem ser coletadas no trajeto (TOTH , et al., 2002).

A Logística Reversa (LR) se encaixa no VRPSPD já que a LR é um processo de gerenciado da cadeia de suprimentos que move itens dos clientes de volta ao depósito para que estes itens sejam direcionados aos seus devidos fins (descarte ou remanufatura) (HERNÁNDEZ, et al., 2012).

Em muitas aplicações a distância entre dois locais pode ser um fator crucial para explicar o comportamento. Existem inúmeras maneiras de calcular distâncias e esses métodos têm vários graus de realismo. Pode-se, por exemplo, calcular uma distância em linha reta ou “grande círculo” entre dois pontos usando suas coordenadas de latitude e longitude. Se alguém estiver modelando o transporte, no entanto, a distância em linha reta pode ser bem diferente da distância realmente percorrida, por exemplo, de carro ou bicicleta. Calcular a distância real de condução é uma tarefa muito mais complexa do que calcular a distância em linha reta, mas quando medidas precisas de distâncias são importantes para uma análise específica, essa complexidade extra pode ser justificada. Sabendo que a criação de rotas é uma decisão importante, este trabalho aborda a coleta e utilização de dados a partir de aplicativos de mapas (SOUZA, 2015).

Para ilustrar o VRPSPD, considere o problema a seguir de uma empresa farmacêutica que atende 5 farmácias, as farmácias estão localizadas na cidade de Goiânia. A empresa deseja entregar e coletar itens em todas as farmácias, utilizando apenas 2 veículos. Pode ser visto na Figura 1 o diagrama da cidade, em que os clientes estão enumerados de 1 a 5 e o número 6 é o depósito da empresa.

Figura 1 - Instância de exemplo

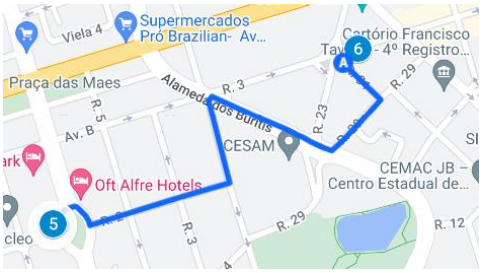


Fonte: elaborada pelo autor.

Para melhor observar a importância dos dados reais a Figura 2 ilustra como é feita a rota do veículo 1. Na Figura 2 (a) pode ser visto a primeira etapa do percurso, na primeira etapa pode ser observado que o veículo 1 sai do depósito (6) e se direciona ao cliente 5. Quando se trabalha com dados reais eles são muito sensíveis, visto que qualquer imprevisto pode mudar a rota totalmente. Na Figura 2 (b) pode ser observado um caminho alternativo para atender o cliente 5 partindo do depósito (6). Na Figura 2 (b) vê-se que ocorreu um acidente no trajeto ilustrado na Figura 2 (a) redirecionando o trajeto da rota,

Figura 2 - Rota feita pelo veículo 1: a e b

a



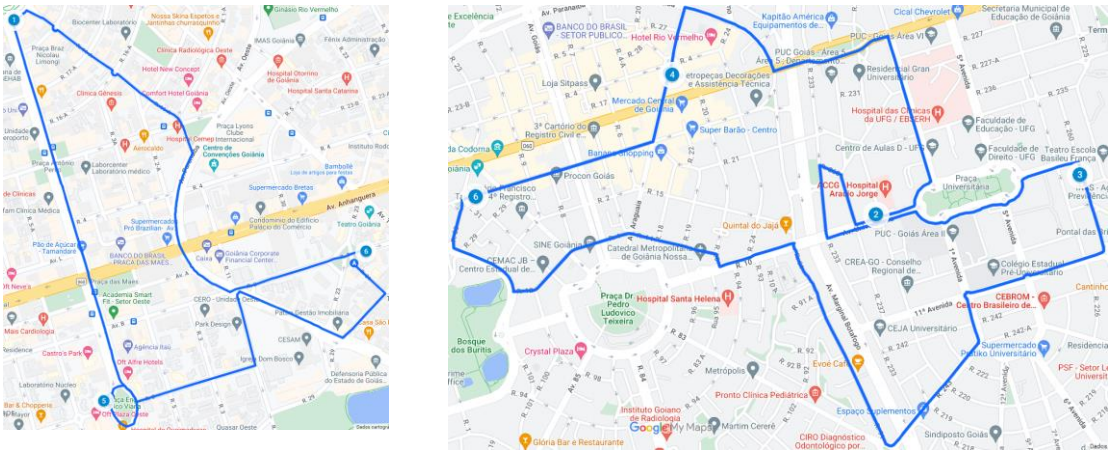
b

Fonte: elaborada pelo autor.

A solução que satisfaz a empresa farmacêutica pode ser vista na Figura 3 – a e b, sendo respectivamente (a) a rota feita pelo veículo 1 e (b) a rota feita pelo veículo 2.

Figura 3 - Solução para instância de exemplo: rota 1 e 2

a b



Fonte: elaborada pelo autor.

A utilização de dados reais possibilita que seja possível modificar o objetivo do VRPSPD. Pode-se além de minimizar a distância total percorrida por todos os veículos, agora é possível minimizar o tempo total gasto para que os veículos atendam todo conjunto de clientes.

O VRPSPD pertence à classe NP-Difícil, dado ser uma generalização do CVRP que já é NP-Difícil, que foi provado ser NP-Difícil por GOLDEN et al., (1981). Devido ao fato de provavelmente não ser possível obter uma solução ótima em tempo polinomial para o VRPSPD, abre-se espaço para a utilização de abordagens heurísticas para tratar o problema.

1.1 Objetivos

O objetivo deste trabalho de conclusão de curso é solucionar o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (VRPSPD), utilizando dados reais coletados a partir de aplicativos de mapas, utilizando três abordagens diferentes, sendo um solver comercial (Gurobi) e duas heurísticas (Busca Tabu e Algoritmo da Otimização por Colônia de Formigas).

Na literatura existem diversos trabalhos que solucionam o VRPSPD, MONTANÉ et al., (2006) propõe resolver o problema com a Busca Tabu, YEOWOON et al., (2012) que propõe uma nova heurística para melhorar a criação de rotas iniciais e MIN, (1989) que utilizou a abordagem cluster-first route-second. Esses autores propõem diferentes abordagens, cada uma dessas abordagens utiliza instâncias fictícias para realizar experimentos computacionais.

Este trabalho propõe utilizar instâncias reais a partir de dados, coletados em aplicativos gratuitos e disponíveis ao público. A fim de comparar a acurácia das soluções propostas, serão feitas comparações entre Busca Tabu, Gurobi e Algoritmo da Otimização por Colônia de Formigas.

1.2 Organização textual

O restante deste trabalho está textualmente assim organizado: o Capítulo 2 apresenta o embasamento teórico sobre as etapas de desenvolvimento, algoritmos heurísticos e a coleta de dados; o Capítulo 3 é composto pela definição do problema e a formulação matemática utilizada; o Capitulo 4 descreve as abordagens utilizadas e a coleta de dados; Capitulo 5 é composto pelos experimentos computacionais sobre a elaboração das instâncias, definição dos parâmetros para cada algoritmo e os parâmetros gerais para o VRPSPD e apresentação dos experimentos computacionais realizados; Por fim o Capítulo 6 conclui com algumas considerações finais.

2 CONCEITOS PRELIMINARES

Neste capítulo são apresentados os fundamentos teóricos para o entendimento deste trabalho. A Seção 2.1 apresenta os conceitos de otimização combinatória.

2.1 Otimização Combinatória (OC)

É um dos campos mais ativos na interface entre pesquisa operacional, ciência da computação e matemática aplicada (SOUZA, 2009). Problemas de otimização combinatória surgem em várias aplicações, incluindo projeto de rede de comunicações, projeto integração em larga escala (VLSI), visão de máquina etc. Além disso, problemas de otimização combinatória existem em diversas áreas, como programação linear, programação linear inteira, teoria dos grafos, inteligência artificial e teoria dos números. Todos esses problemas, quando formulados matematicamente como a minimização ou maximização de uma determinada função definida em algum domínio, têm uma semelhança de discrição. (SOUZA, 2009)

Os problemas de OC podem ser vistos como a busca do melhor elemento em algum conjunto de itens. Muitos desses problemas, são encontrados extensivamente na vida real, sendo categorizados como de problemas NP-Difíceis, segundo D.NELSON, et al., (1985). O Problema de Roteamento de Veículos (VRP) é classificado como um problema NP-difícil (TOTH , et al., 2002), o que significa que o tempo de solução necessário aumenta exorbitantemente com o crescimento do número de nós. O número de soluções possíveis para o VRP é da ordem de n!, onde n é o número de nós (locais que o veículo deve alcançar) (D.NELSON, et al., 1985)

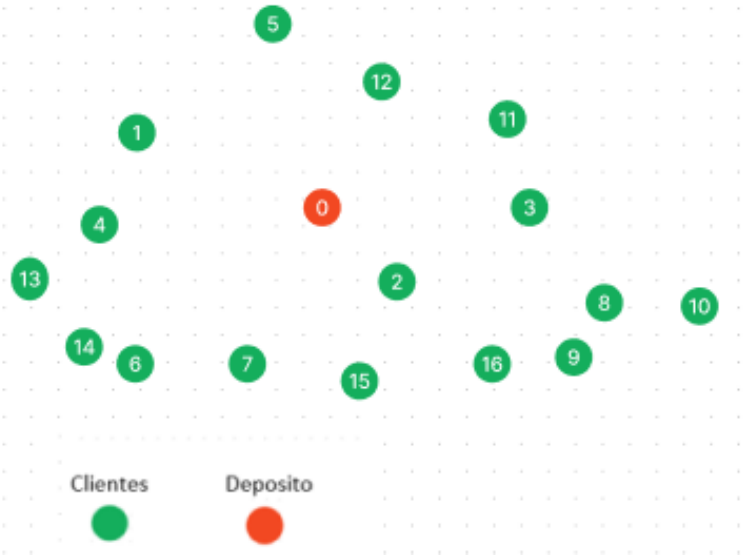
2.2 Problema de Roteamento de Veículos (VRP)

O Problema de Roteamento de Veículos (VRP) é um problema clássico de otimização combinatória, envolvido em muitas aplicações. É um nome genérico dado a toda uma classe de problemas, em que um conjunto de rotas, para uma frota de veículos baseada em um ou vários depósitos, deve ser determinado para várias cidades ou clientes geograficamente dispersos (TOTH , et al., 2002). O objetivo do VRP é atender a um conjunto de clientes com demandas conhecidas, em rotas de veículos de custo mínimo com origem e término em um depósito ou centro de distribuição (DANTZIG, et al., 1959). O VRP é um problema importante nas áreas de transporte, distribuição e logística. Além do VRP clássico, diferentes variantes do VRP continuam a surgir, pois há muitas possibilidades de configurações e características de problemas da vida real como por exemplo: a variação no número de depósitos, diferentes tipos de veículo, variações nos tipos requisitos.

Para ilustrar o problema, suponha o seguinte cenário, em que existe uma empresa que precisa visitar todos os seus 16 clientes em uma cidade. A Figura 4, apresenta o diagrama de uma cidade em que os pontos (clientes) foram enumerados de 1 a 16 e o ponto 0 representa o depósito da empresa. O objetivo desta empresa é visitar todos os seus clientes percorrendo a menor distancia tota. Para realizar esta visita, a empresa irá utilizar 4 veículos, todos partindo do Depósito e os veículos não podem ultrapassar uma distância máxima após sair do depósito. A solução que esta empresa obteve pode ser vista na Figura 5. Visualizando esse problema pode-se perceber que o VRP é importante para

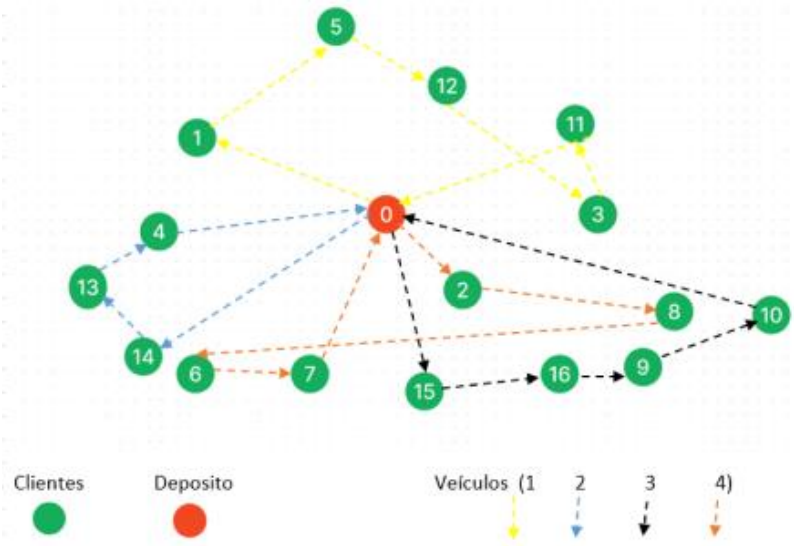
empresas de entrega, mas também para qualquer operação que envolva fazer paradas em vários locais, como caminhões de lixo e empilhadeiras que estocam um armazém.

Figura 4 - Exemplo: Diagrama da cidade



Fonte: elaborada pelo autor.

Figura 5 - Solução exemplo



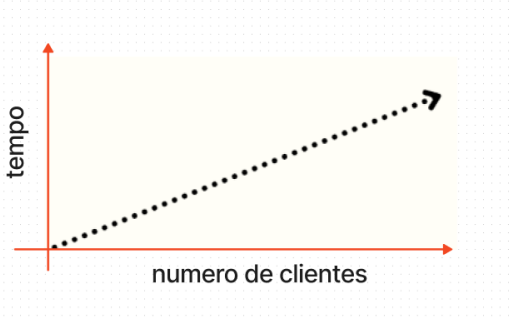
Fonte: elaborada pelo autor.

2.3 Problema de Roteamento de Veículo Capacitado (CVRP)

O CVRP Adiciona veículos com capacidade de carga limitada, em que é preciso coletar ou entregar itens em vários locais. Os itens têm uma quantidade, como peso ou volume, e cada veículo tem uma capacidade máxima que pode transportar. O problema é coletar ou entregar os itens com o menor custo, sem nunca ultrapassar a capacidade dos veículos. Todos os algoritmos exatos usados para resolver esse problema são aplicáveis apenas para pequenas instâncias e enfrentam problemas de complexidade de tempo para grandes redes. A incapacidade das abordagens exatas para resolver problemas de

roteamento de veículos de médio e grande porte, bem como a dificuldade em avaliar a função objetivo em problemas complexos da vida real, são duas razões principais pelas quais heurísticas e meta-heurísticas são necessárias para resolver tais problemas. Pode ser visto na Figura 6 uma relação entre o aumento do número de clientes em relação ao tempo demorado para encontrar uma rota que que atenda todas as restrições do problema. (TOTH , et al., 2002)

Figura 6 - Relação entre tempo x número de clientes



Fonte: adaptado de TOTH, et al., (2002)

As variantes do problema de roteamento estão cada vez mais complexas anexando diferentes tipos de requisitos ao problema (TOTH , et al., 2002). Algumas adicionam janelas de tempo, envolvem o agendamento de visitas a clientes que só estão disponíveis em janelas de tempo específicas. A Logística Reversa (LR) é um tipo de gerenciamento da cadeia de suprimentos que move mercadorias dos clientes de volta aos vendedores ou fabricantes. Depois que um cliente recebe um produto, processos como devoluções ou reciclagem exigem logística reversa. A LR começa no consumidor final, retrocedendo pela cadeia de suprimentos até o distribuidor ou do distribuidor para o fabricante. A logística reversa também pode incluir processos em que o consumidor final é responsável pela destinação final do produto, incluindo reciclagem, reforma ou revenda. A LR pode ser aplicada ao CVRP, de forma que o veículo parte do centro de distribuição vazio e coleta itens no percurso retornando ao depósito com os itens, sem nunca ultrapassar sua capacidade máxima enquanto coleta os itens. O VRPSPD é uma generalização do CVRP onde é adicionado um conjunto de demandas de coleta junto ao conjunto de demandas de entrega.

2.4 Heurísticas

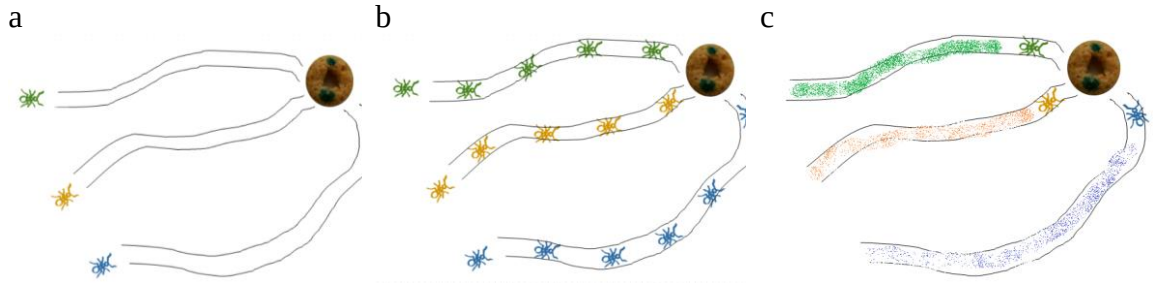
Segundo Santos, et al., (2019) muitos problemas de OC ainda não possuem algoritmos capazes de fornecer soluções ótimas para instâncias práticas em tempo hábil. Para se resolver esses problemas, uma solução razoável pode ser admitida, próxima à ótima, já que a solução exata seria difícil ou inviável de ser obtida. Para buscar essas soluções razoáveis, existem as meta-heurísticas, métodos de solução com estratégias.

Meta-heurísticas são normalmente algoritmos de melhoria, ou seja, começam com uma ou mais soluções viáveis para o problema em questão e sugerem métodos para melhorar tais soluções. Dentre as meta-heurísticas existentes este trabalho aborda duas, o Algoritmo da Otimização por Colônia de Formigas (ACO) e Busca Tabu (TS).

2.4.1 Algoritmo da Otimização por Colônia de Formigas (ACO)

O ACO foi proposto por DORIGO, (1992) e é uma meta-heurística baseada em população que pode ser usada para encontrar soluções aproximadas para problemas de otimização difíceis (GENTILE, et al., 2015). Esse Algoritmo tem como base o comportamento, real de formigas para buscar alimentos. Para entender esse comportamento forma, suponha que existem 3 formigas, uma verde, uma laranja e uma azul, como ilustrado na Figura 7: a-c, em que elas partem de um determinado ponto inicial que pode ser observado na Figura 7 (a), cada uma das formigas vai escolher e percorrer um caminho, até o alimento representado por uma rosquinha Figura 7 a-c. Enquanto as formigas estão percorrendo o caminho visto na Figura 7 (b), elas liberam uma substância chamada feromônio, representado na Figura 7 (c) como a trilha verde, laranja e azul. O feromônio é utilizado pelas formigas para se comunicarem. Essa substância some com o tempo. Logo, quanto maior o caminho percorrido pela formiga, menor será a concentração de feromônio no início do caminho como ilustrado na Figura 7 (c).

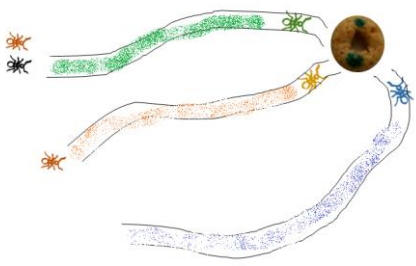
Figura 7- Exemplo ACO: a-c



Fonte: elaborada pelo autor.

Quando outras 3 formigas, uma preta, uma vermelha e uma roxa partem em busca de alimentos (mesmos pontos iniciais que as anteriores) elas estão propensas a escolher o caminho que tem a maior concentração de feromônios, como pode ser observado na Figura 8. Em que o caminho azul pôr o mais longo dos três, em seu início quase não há feromônio pois grande parte já evaporou, não sendo satisfatório para as formigas escolherem aquele caminho.

Figura 8 - Exemplo ACO 2



Fonte: elaborada pelo autor.

Na heurística ACO, um conjunto de agentes de *software* chamados formigas artificiais, busca boas soluções para um determinado problema de otimização. Para aplicar o ACO, o problema de otimização é transformado no problema de encontrar o melhor caminho em um grafo ponderado. As formigas artificiais constroem soluções incrementalmente, movendo-se no grafo. O processo de construção da solução é estocástico e é influenciado por um modelo de feromônios, ou seja, um conjunto de parâmetros associados a componentes do grafo (nós ou arestas), cujos valores são modificados em tempo de execução pelas formigas (nível de feromônio). As formigas constroem as soluções da seguinte forma: cada formiga parte de um local selecionado, então, a cada passo de construção, ela se move ao longo das arestas do grafo; cada formiga guarda uma memória de seu caminho e, nas etapas subsequentes, ela escolhe entre as arestas que não levam aos vértices que já visitou. Uma formiga constrói uma solução depois de ter visitado todos os vértices do grafo. A cada passo de construção, uma formiga escolhe, probabilisticamente, a aresta a seguir entre aquelas que levam a vértices ainda não visitados. A regra probabilística é influenciada por valores de feromônio e informações heurísticas. Quanto maior a concentração de feromônio, o valor heurístico associado a uma aresta, maior a probabilidade de uma formiga escolher essa aresta em particular. Uma vez que todas as formigas tenham completado seu trajeto, o feromônio nas bordas é atualizado.

O Algoritmo da Otimização por Colônia de Formigas vem sendo constantemente usado para resolver variantes do VRP (OLIVEIRA, et al., 2020). Conseguindo bons resultados em muitos das aplicações como YAN, (2018) que aplicou o ACO ao roteamento de veículos autônomos e MA et al., (2017) aplicou o ACO ao VRP com janela de tempo e ambos autores obtiveram bons resultados com o ACO. Este trabalho utiliza o ACO pelo fato que a heurística mostrar bons resultados em outras variantes do VRP e pela fácil implementação do algoritmo.

2.4.2 Busca Tabu (TS)

Glover, (1990) Propôs o método Busca Tabu (TS) em 1986 para resolver problemas de otimização combinatória. Os princípios do método TS são a abordagem de busca de vizinhança e a lista tabu (TL), dado que a busca de vizinhança é o processo para se selecionar possíveis soluções e a lista tabu uma lista de soluções que estão proibidas de serem selecionadas. Embora a busca tabu exista há muitos anos e seus princípios fundamentais sejam bem elaborados, ela ainda é frequentemente implementada de uma forma muito simplista que desconsidera todas as características do método. TS é um dos procedimentos meta-heurísticos mais utilizados para resolver problemas de otimização combinatória. É uma heurística de melhoria baseada em busca local. Ele começa com uma solução inicial para o problema, chamada de solução atual e busca a melhor solução em uma vizinhança adequadamente definida dessa solução. Em seguida, ele designa a melhor solução na vizinhança como a solução atual e inicia o processo de busca novamente. A pesquisa tabu termina quando certas condições de término, envolvendo tempo de execução ou condições de contagem máxima de iteração, ou objetivos de qualidade da solução, ou ambos, foram atendidos. Para evitar que a busca tabu considere as soluções que visitou em iterações recentes, a busca tabu mantém uma lista de soluções de geração de vizinhos que considera proibidos e ignora as soluções que podem ser

alcançadas usando essas soluções. Uma vez que uma solução entra na lista de movimentos tabu, ele permanece lá por um número pre-especificado de iterações. A lista tabu, portanto, muda continuamente durante a execução da busca, tornando a busca tabu um algoritmo de busca de memória adaptável.

2.4.3 Serviços de Mapeamento

É uma maneira de calcular distâncias utilizando serviços de terceiros, como os serviços de mapeamento do Google (GOOGLE), que permite montar a rota entre dois pontos, tendo como retorno uma matriz de distância ou uma matriz de tempo, mas visto que o serviço é pago e é cobrado a cada 1000 requisições $ 10.00 dólares e o valor equivale a R$ 53,87 na conversão de (28/11/2022), se pegarmos um problema com 200 clientes e montar-se a matriz de distância para o mesmo, tem-se que realizar cerca de 40000 requisições à API do Google, cerda de $ 400.00 dólares ou R$ 2154,96 na conversão de (28/11/2022), tendo em vista que é um trabalho busca montar vários tipos de instâncias, o custo real para se utilizar a API do Google torna-se muito alto. Logo foi buscado para outro serviço de mapeamento, um que é gratuito chamado MapQuest (MAPQUEST, et al.) que assim como o Google, ele também tem uma API para se montar a rota entre dois pontos.

3 O PROBLEMA DE ROTEAMENTO DE VEICULOS COM COLETA E ENTREGA SIMULTANEA (VRPSPD)

O VRPSPD é um problema de OC que possui o objetivo que sejam definidas rotas que minimizam a distância total percorrida por todos os veículos e passe por todos os clientes saindo e retornando ao depósito satisfazendo a demandas deles.

3.1 Definição do problema

O Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas (VRPSPD) é uma extensão do CVRP e, nesta variante, os clientes exigem não apenas a entrega de mercadorias, mas também a coleta simultânea de mercadorias. Uma suposição geral é que todas as mercadorias entregues são originárias do depósito e todas as mercadorias retiradas devem ser transportadas de volta ao depósito (TOTH , et al., 2002). O VRPSPD foi introduzido pela primeira vez por MIN, (1989), que estudou a atividade de entrega e coleta de livros entre uma biblioteca central e 22 bibliotecas locais com um determinado número de veículos capacitados. MIN, (1989) propôs uma abordagem cluster-first route-second, na qual as localizações dos clientes são agrupadas primeiro e, em seguida, uma rota para cada cluster é gerada por um algoritmo de problema do caixeiro viajante (TSP). DETHLOFF, (2001) Destacou a importância do VRPSPD nas operações de LR. Sua abordagem de solução é baseada em um método de inserção mais barato. O critério de inserção da abordagem leva em consideração três métricas: distância percorrida, capacidade residual e sobretaxa radial. NAGY, et al., (2005) propuseram quatro heurísticas baseadas em inserção para resolver VRPSPD. Depois de construir rotas parciais para um conjunto de clientes, os clientes restantes são inseridos repetidamente nas rotas parciais pelo método de inserção. MONTANÉ, et al., (2006). Como o VRP é um problema NP-difícil muito complexo (GOLDEN, et al., 1981), a resolução dos VRPs da vida real até a otimalidade geralmente não é possível dentro do tempo de computação limitado disponível em situações práticas. Portanto, a maioria das pesquisas disponíveis tem focado em métodos de solução heurísticas e meta-heurísticas projetados para produzir soluções de alta qualidade em um determinado tempo.

3.2 Formulação Matemática - VRPSPD

Quando MIN, (1989) propôs o VRPSPD, junto a proposta do problema também foi apresentada a formulação matemática de programação linear inteira para esse problema, visto que a formulação de MIN, (1989) tinha várias restrições viáveis para a época e que atualmente não são mais tão satisfatórias, MONTANÉ, et al., (2006) propôs uma formulação matemática alternativa ao VRPSPD. Este trabalho utiliza a formulação alternativa de MONTANÉ, et al., (2006).

Notação

V conjunto de clientes

V_0 conjunto de clientes mais depósito (cliente 0): V_0 = V ∪ {0}

n número total de clientes: n = |V |

C_{ij} distância entre os clientes i e j

P_i demanda de coleta do cliente j, j = 1,...,n

d_j demanda de entrega do cliente j, j = 1,...,n

Q capacidade do veículo

MD distância máxima permitida para qualquer rota k

k número máximo de veículos

variáveis de decisão

$$X_{ij}^k = \begin{cases} 1, se\ arco(i,j)\ pertencer\ \text{à}\ rota\ operada\ pelo\ veículo\ k \\ 0, caso\ contrário. \end{cases}$$

$Y_{ij} = demanda\ coletada\ em\ clientes\ roteados\ até\ o\ nó\ i\ (incluindo\ o\ nó\ i)$

$e\ transportados\ no\ arco\ (i,j)$

$Z_{ij} = demanda\ a\ ser\ entregue\ aos\ clientes\ roteados\ após\ o\ nó\ i$

$e\ transportados\ no\ arco\ (i,j)$

$$Min \sum_{k=1}^{k} \sum_{i=0}^{n} \sum_{j=0}^{n} C_{ij} X_{ij}^k \quad (1)$$

Sujeito a

$$\sum_{i=0}^{n} \sum_{k=1}^{k} X_{ij}^k = 1, \ \ j = 1, \dots, n \quad (2)$$

$$\sum_{k=1}^{k} X_{ij}^k - \sum_{k=1}^{k} X_{ji}^k = 0, \quad \begin{matrix} \mathrm{j} = 0, \dots, \mathrm{n} \\ \mathrm{k} = 0, \dots, \mathrm{k} \end{matrix} \quad (3)$$

$$\sum_{i=0}^{n} X_{0j}^k \leq 1, \ \ k = 1, \dots, k \quad (4)$$

$$\sum_{i=0}^{n} \sum_{j=0}^{n} C_{ij} X_{ij}^k \ \leq MD, \ \ k = 1, \dots, k \quad (5)$$

$$\sum_{i=0}^{n} Y_{ij} - \sum_{i=0}^{n} Y_{ij} = P_j, \ \ \forall_{j \neq 0} \quad (6)$$

$$\sum_{i=0}^{n} Z_{ij} - \sum_{i=0}^{n} Z_{ij} = d_j, \quad \forall_{j \neq 0} \tag{7}$$

$$Y_{ij} + Z_{ij} \leq Q \sum_{k=1}^{k} X_{ij}^{k}, \quad i,j = 0, \ldots, n \tag{8}$$

$$X_{ij} \in \{0,1\}, \qquad i,j = 0, \ldots, n \tag{9}$$

$$Y_{ij} \geq 0, \qquad i,j = 0, \ldots, n \tag{10}$$

$$Z_{ij} \geq 0, \qquad i,j = 0, \ldots, n \tag{11}$$

A função objetivo (1), busca minimizar a distância total percorrida por todos os veículos. A restrição (2) garante que cada cliente seja visitado por exatamente um veículo; restrição (3) garante que o mesmo veículo chegue e saia de cada cliente que atende. A restrição (4) define que no máximo k veículos são usados; a restrição (5) é a de distância máxima (MD). As restrições (6) e (7) são equações de fluxo para demandas de coleta e entrega, respectivamente, garantem que ambas as demandas (coleta e entrega) sejam satisfeitas para cada cliente. A restrição (8) estabelece que as demandas de coleta e entrega somente serão transportadas por meio de arcos incluídos na solução, impõem ainda um limite superior à carga total transportada por um veículo em qualquer seção da rota. Finalmente, as restrições (9) –(11) definem a natureza das variáveis de decisão.

4 ABORDAGENS UTILIZADAS

Neste Capítulo são descritas as abordagens utilizadas para solucionar o VRPSPD. Ao longo do texto são apresentados pseudocódigos para as abordagens utilizadas e é apresentada coleta de dados.

4.1 Gurobi

O Gurobi é um solver comercial que utiliza programação linear inteira para resolver diversos problemas, utilizando a formulação matemática de cada problema. Com Ela foi possível abordar o VRPSPD de forma exata (para instâncias pequenas e médias), permitindo que seja possível realizar comparações com as abordagens heurística propostas.

O Gurobi inicia reduzindo as dimensões da modelagem do problema ao excluir as variáveis e as restrições que sejam redundantes ou triviais. Em seguida, um método de busca enumerativo, baseado no algoritmo Branch-and-Bound é executado, combinando características dos métodos de plano de corte e aplicando heurísticas às soluções visitadas a fim de acelerar a busca. (SANTOS, et al., 2019)

MONTANÉ, et al., (2006) propôs utilizar outro solver para abordar o VRPSPD de forma exata, ele implementou a formulação matemática do problema no solver (CPLEX).

4.2 Busca Tabu (TS)

O trabalho de MONTANÉ, et al., (2006) utiliza como proposta de solução a TS. E as estruturas de memória usadas na busca tabu podem ser divididas em três categorias:

- Curto prazo: A lista de soluções recentemente consideradas. Se uma solução potencial aparecer na lista tabu, ela não poderá ser revisitada até atingir um ponto de expiração.
- Prazo intermediário: Regras de intensificação destinadas a direcionar a busca para áreas promissoras do espaço de busca.
- Longo prazo: Regras de diversificação que direcionam a busca para novas regiões (ou seja, com relação a redefinições quando a busca fica presa em um platô ou em um beco sem saída abaixo do ideal).

A memória de curto prazo sozinha pode ser suficiente para alcançar soluções superiores àquelas encontradas pelos métodos convencionais de busca local, mas estruturas intermediárias e de longo prazo são frequentemente necessárias para resolver problemas mais difíceis. (MALEK, et al., 1989)

4.2.1 Pseudocódigo da Busca Tabu utilizado

O pseudocódigo abordado neste trabalho foi proposto por MONTANÉ, et al., (2006) para solução do VRPSPD

Algoritmos 1 - Busca Tabu

```
1  Sbest = S0 #Gera a solução inicial
2  melhorCandidato = Sbest
3  ListaTabu <- [] #inicializa a lista tabu
4  enquanto (não atingir_os_criterios REPITA)
5      gerar vizinhança para Sbest
6      SCanditata #recebe a primeira solução da vizinhança de Sbest
7      para (cada SCanditata em vizinhança de Sbest)
8          se (SCanditata não esta em ListaTabu e SCanditata é melhor que melhorCandidato)
9              melhorCandidato = SCanditata
10         fim_se
11     fim_para
12     se (melhorCandidato é melhor que Sbest)
13         Sbest = melhorCandidato
14     fim_se
15     atualiza ListaTabu(Adicionar melhorCandidato)
16     se(tamanho(ListaTabu) > MaximoDeElementos)
17         Remover primeiro elemento da ListaTabu
18     fim_se
19 fim_enquanto
20 retornar Sbest
```

Onde na linha 1 é construído a solução inicial; linha 2 melhor solução recebe a solução construída; linha 3 a lista tabu; a linha 4 tem o laço de repetição que irá continuar até atender as condições estabelecidas; nas linhas 5 e 6 respectivamente geramos as soluções candidatas na vizinhança da melhor solução e definimos as soluções candidatas; linha 6 é o laço usado para passar por todas as soluções candidatas; linha 7 verifica se alguma das soluções candidata é melhor que melhor solução candidata atual; linha 8 substitui a melhor solução pela solução candidata; as linhas 11 e 12 respectivamente verifica se a melhor solução candidata é melhor que a melhor solução e se ela for substitui ela pela melhor solução candidata; linha 14 adiciona na lista tabu a melhor solução; linha 15 se a lista tabu está cheia remover o primeiro elemento da lista; linha 17 retornar a solução

4.3 ACO

A abordagem proposta neste trabalho é o ACO, em especifico o MAX-MIN Ant System (MMAS) que é uma variante do ACO, proposto por Stützle, et al., (2000). O MMAS difere do ACO na medida em que alcança uma forte exploração do histórico de busca, permitindo que apenas as melhores soluções adicionem feromônio durante a atualização da trilha de feromônio. Além disso, o uso de um mecanismo bastante simples para limitar as forças das trilhas de feromônio efetivamente evita a convergência prematura da busca.

A Equação 12 de determina taxa de atualização de feromônios.

$$T_{ij} \leftarrow (1 - p) * T_{ij} + \Delta T_{ij}^{best} \tag{12}$$

Os três aspectos principais do MMAS. A seguir são descritos.

- Para explorar as melhores soluções encontradas durante uma iteração ou durante a execução do algoritmo, após cada iteração, apenas uma única formiga adiciona feromônio. Essa formiga pode ser aquela que encontrou a melhor solução na iteração

atual (melhor formiga da iteração) ou aquela que encontrou a melhor solução desde o início da tentativa (melhor formiga global).

- Para evitar a estagnação da busca, o intervalo de possíveis trilhas de feromônio em cada componente da solução é limitado a um intervalo $[T_{\min}, T_{max}]$
- Adicionalmente, inicializa deliberadamente as trilhas de feromônios para T_{max}, conseguindo assim uma maior exploração de soluções no início do algoritmo.

Três Parâmetros também são importantes para que ACO execute de forma eficiente sendo eles α, β e ρ.

O parâmetro α regula a importância do conhecimento do feromônio. O valor deste parâmetro comumente utilizado na literatura é 1, coincidindo com o valor que algumas espécies de formigas possuem. Um valor maior aumenta a importância dos valores do feromônio nos arcos, enquanto um valor menor diminui. Aumentar α faz com que o algoritmo explore o conhecimento acumulado nas primeiras iterações. Observe que nas iterações iniciais todos os arcos possuem o mesmo valor de feromônio, portanto as soluções encontradas primeiro são completamente aleatórias

O parâmetro β intensifica o uso de informações heurísticas em comparação com as informações de feromônio acumuladas até o momento. Este parâmetro é totalmente dependente da instância do problema, esse parâmetro pode mudar de uma instância para outra, quando a informação heurística por si só é suficiente para encontrar o melhor a melhor solução, portanto, definir um valor alto de β que supere o parâmetro α seria uma boa escolha quando a informação heurística não é forte o suficiente para encontrar a melhor solução e recomendado um β inferior a α (GENTILE, et al., 2015), Mas visto que este parâmetro é totalmente dependente do da instância do problema o melhor e realizar várias experimentações para decidir qual o melhor valor de β.

O parâmetro ρ faz o feromônio evaporar como a natureza faz com o feromônio real. Quanto maior ρ, maior o poder de esquecimento do conhecimento passado. Um ρ muito alto pode levar a ficar preso em um ótimo local, enquanto um ρ muito pequeno faz com que o algoritmo demore muito até que o conhecimento acumulado se torne efetivo na escolha dos arcos.

4.3.1 pseudocódigo do MMAS utilizado

O algoritmo 1 exibe o pseudocódigo do MMAS que utilizamos , sendo que na linha 1 e inicializada a matriz de feromônios; linha 2 e gerada a solução inicial para iniciar o algoritmo; linha 3 condição de parada do algoritmo; linhas 4 – 6 inicializam os índices de melhor solução, comprimento da solução, e melhor solução da iteração; na linha 7 tem um loop que passa por todas as formigas para que cada uma possa gerar uma solução; linhas 8 – 10 inicializam as rotas para cada formiga, iniciando vazio, definindo a distância máxima permitida e inicia a lista de visitados. A linha 11 seleciona qual caminho a formiga deve seguir com base no feromônio disponível; linhas 12 – 14 selecionam os próximos pontos que a formiga deve seguir, e verifica se está dentro da duração do percurso; linhas 18 – 24 verificam a acurácia da solução e se deve iniciar outra iteração; e pôr fim a linha 25 retorna à solução final.

Algoritmos 2 - MMAS

```
1       Inicialize a matriz de feromônios: ∀i, j = 1, . . . , n : τi,j = τ0

2       Inicialize a melhor solução global: lgb = +∞, lgb = null
# onde lgb é o comprimento da solução e tgb a matriz contendo os arcos na solução

3       Enquanto(não condição-parada):
4           Inicialize o melhor índice de solução da iteração: ib := null
5           Inicialize o melhor comprimento da iteração: lib := null
6           Inicialize o melhor tour da iteração: tib := null
7           para k:= 1,...,número_de_formigas:
8               Inicialize o tour de ant k: tk := null
9               Inicialize a duração do percurso de ant k: lk := MD
10              Inicialize os clientes visitados set: C := null

11              selecione a primeiro cliente i ∈ V , onde V é o conjunto disponíveis

12              enquanto (∃ cliente /∈ C)
13                      escolha o próximo cliente j ∈ V, j /∈ C a ser visitado

14                      Verificar se (i, j) está dentro da duração do percurso k

15               Fim-enquanto
16              lk := tamanho(tk)
17          Fim-para
18          verifique o índice de melhor solução de iterações: em := melhor(lk)
19          defina o melhor comprimento da iteração: lib := lk : k = ib
20          defina o melhor rota da iteração: tib := tk : k = ib
21          Se lib < lgb ⇒ ( lgb := lib e tgb := tib)
22          faça o feromônio evaporar
23          aumentar o feromônio nos arcos da melhor solução
24      Fim-enquanto
25      retornar a melhor solução global encontrada: lgb, tgb
```

4.4 Coleta dos dados

Para coletar os dados através do MyMaps (GOOGLE) utilizamos uma técnica chamada de *scraping* que é uma forma de mineração de dados que permite que seja feita a extração de grande quantidade de dados de sites ou plataformas (GLEZ-PEÑA, et al., 2014) utilizando esta técnica podemos automatizar a coleta dos dados, como estamos trabalhando com dados reais precisaríamos fazer uma seleção manual. Mas com a técnica podemos apenas definir parâmetros e esperar obter resultados a partir deles.

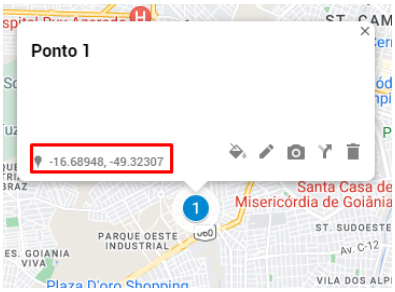
4.4.1 MyMaps

A plataforma permite que seja definida vários alfinetes no mapa pode ser visto na Figura 9, pode ser observado na parte 1, 3 pontos (1, 2 e 3) e pode ser visto na parte 2 as coordenadas do ponto 1, a partir dos alfinetes podemos exportar esses dados em especial as coordenadas dos pontos.

Figura 9 - Dentro do MyMaps

Parte 1

Parte 2



Fonte: elaborada pelo autor.

A Google permite exportar para um arquivo .CSV na Figura 10 pode ser visto como é o arquivo exportado pela Google, com um arquivo CSV podemos utilizar a linguagem de programação Python para extrair do arquivo as coordenadas.

Figura 10 - Arquivo exportado do MyMaps

	A	B
1	WKT	nome
2	POINT (-49.3230765 -16.6894814)	Ponto 1
3	POINT (-49.2834227 -16.7072391)	Ponto 2
4	POINT (-49.26557 -16.7003335)	Ponto 3

Fonte: elaborada pelo autor.

4.4.2 MapQuest

Com as coordenadas em mãos podemos utilizar a API da plataforma para passarmos as coordenadas dos pontos e obter como resposta uma matriz de distância, pode ser visto na Figura 11 os parâmetros que a API precisa, sendo dois ou mais objetos contendo dois atributos sendo lng a longitude e lat a latitude esta API também espera um parâmetro booleano chamado allToALL que é definir se resposta vai ser uma matriz de totós para todos ou não. Como reposta da API é retornado uma matriz de distância e uma matriz de tempo.

Figura 11 - Entrada de dados MAPQUEST

Fonte: elaborada pelo autor.

5 EXPERIMENTOS COMPUTACIONAIS

5.1 Elaboração das instâncias

Como visto na seção 4.4.1 é possível automatizar a coleta dos dados com *scraping* de dados, para criar as instâncias foram definido três critérios diferentes para instâncias pequenas, médias e grandes: respectivamente para as instâncias pequenas foi escolhido o critério de farmácias, elas devem estar dentro da região metropolitana de Goiânia e o centro de distribuição foi definido, arbitrariamente na Praça Cívica de Goiânia; para as instâncias médias foi definido o critério supermercados, elas devem estar dentro do estado de Goiás e, no máximo, 30 km no entorno do estado. O centro de distribuição também definido na Praça Cívica; para as instâncias grandes foi definido o critério restaurante, elas devem estar dentro da região metropolitana de São Paulo e o centro de distribuição foi definido no Mercado Municipal da cidade de São Paulo.

Esses três critérios foram escolhidos não só pelo fato pelo fato de a logística reversa estar introduzida nesses negócios (farmácias tem medicamentos que não podem ser descartados em qualquer lugar e precisam voltar para o fabricante, supermercados tem itens perecíveis ou retornáveis que precisam ser coletados e restaurantes possuem itens que precisam retornar ao fabricante) mas também pelo fato de serem fáceis de coletar e haver muitos.

Foram coletados 279 pontos para as instâncias pequenas (farmácias), 558 pontos para as instâncias médias e 4166 pontos para instâncias grade (restaurantes). Para cada ponto foi coletado coordenadas (latitude e longitude) é para construir as instâncias de teste. Elas foram listadas nas Tabela 1, Tabela 2 e Tabela 3, cada uma destas tabelas foram representadas por Instância sendo o número da instância seguido por um caractere ‘f’, ‘s’ ou ‘r’ sendo respectivamente farmácias, supermercados ou restaurantes, |N| sendo o número de pontos que instância possui e |V| o número de veículos máximo utilizado por aquela instância.

5.1.1 Instâncias Pequenas - Farmácias

Os dados de farmácias foram os menores, após a coleta dos pontos foram construídas, arbitrariamente instâncias com 10, 20, 50, 90 e 130 pontos e uma com todos os 279 pontos.

Tabela 1 - instâncias de farmácia

Instância	|N|	|V|
1-f	10	3
2-f	20	3
3-f	50	3
4-f	90	7
5-f	130	7
6-f	279	7

5.1.2 Instâncias Médias - Supermercados

Os dados coletados de supermercados foram divididos em 3 instâncias 290, 350, 558 pontos, dado que a instância 6-f é a maior de farmácias e possui 279 pontos foi escolhido arbitrariamente criar as instâncias de supermercados partindo de 290 pontos.

Tabela 2 - instâncias de supermercados

Instância	|N|	|V|
1-s	290	8
2-s	350	9
3-s	558	10

5.1.3 Instâncias Grandes Restaurantes

Foram coletados 4166 pontos de restaurantes, as instâncias foram escolhidas arbitrariamente a partir destes pontos, resultado em 3 instâncias, uma de 1000 pontos, uma de 2500 pontos e uma de 4166 pontos.

Tabela 3 - instâncias de restaurantes

Instância	|N|	|V|
1-r	1000	15
2-r	2500	15
3-r	4166	15

5.2 Definição dos parâmetros

5.2.1 Gurobi

O Gurobi é o solver a ser utilizado para se tentar obter a solução exata das instâncias de teste, mas como visto, problemas de OC podem demorar muito ou ser inviáveis em muitos casos, partindo desse pressuposto, o Gurobi possui um parâmetro chamado *TimeLimit* que é o tempo máximo que ele executara até obter a solução, caso atinja esse tempo, ele para. Neste Trabalho foi utilizado um *TimeLimit* de 3600 segundos (1 hora).

5.2.2 ACO

Os parâmetros do algoritmo considerados neste trabalho são os mais populares e críticos em qualquer algoritmo ACO, ou seja, os expoentes α e β, que representam a influência das trilhas de feromônio e informações heurísticas, ρ que é a força de atualização do feromônio e n que número de formigas. Após analisar diversos trabalhos e testar alguns diferentes valores para cada um destes parâmetros, os que foram escolhidos são mostrados na Tabela 4.

Tabela 4 - parâmetros iniciais ACO

Tempo limite	3600 segundos
Número máximo de iterações sem melhora	150
α	1
β	2
ρ	0,2
n	30

5.2.3 TS

A quantidade de soluções excluídas e adicionadas na lista tabu são diferentes (tDeletar e tAdiconar). Isso se baseia na observação de que o número de soluções que podem ser adicionadas é muito maior do que o número de soluções que podem ser excluídas (MONTANÉ, et al., 2006), o que significa que o prazo de validade das soluções excluídas (que são candidatas a serem adicionadas em iterações subsequentes) deve ter um valor maior do que adicionadas na lista. O parâmetro C é a quantidade máxima de elementos que a Lista Tabu pode possuir. A Tabela 5 mostra a relação de parâmetros usados.

Tabela 5 - parâmetros iniciais TS

Tempo limite	3600 segundos
Número máximo de iterações sem melhora	150
C	200
tDeletar	30
tAdicionar	15
tamanho da lista	60

6 RESULTADOS

Para executar todos os experimentos computacionais foi utilizado um servidor adquirido na empresa portuguesa OVH foi utilizado: Processador Dual Intel Xeon Silver 4214R - 24c/48t - 2.2GHz/3.5GHz; Memoria 96GB DDR4 ECC 2400MHz; Armazenamento 2× 960GB SSD NVMe Soft RAID; Localização Alemanha (Frankfurt - FRA); Sistema Operacional Ubuntu Server 18.04 LTS; Linguagem de programação; Python 3.9.7; Gurobi 10.0.0

Para comparar a acurácia dos experimentos computacionais foi utilizado como métrica o erro relativo, que é a razão entre o erro absoluto e o valor experimental. A Equação 13 mostra como é calculado o erro relativo, em que S é o valor da solução exata, S' é o valor experimental. A subtração destes dois valores $S - S'$ gera o erro absoluto, multiplicando estes valores por 100 e dividindo por S obtemos o percentual do *gap* (erro relativo).

$$gap = \frac{|S - S'| * 100}{S} \tag{13}$$

A fim de realizar a comparação entre os três algoritmos, Gurobi, TS e ACO a Equação 13 foi modificada resultando em três equações que definem o erro relativo, para realizar esta comparação, a Equação 14, Equação 15 e Equação 16. Sendo S o valor do solver, S^{ACO} o valor da heurística ACO e S^{TS} o valor da heurística TS.

$$gapGx\mathrm{ACO} = \frac{|S - S^{ACO}| * 100}{S} \tag{14}$$

$$gapGxTS = \frac{|S - S^{TS}| * 100}{S} \tag{15}$$

$$gap\mathrm{ACO}xTS = \frac{|S^{ACO} - S^{TS}| * 100}{S^{ACO}} \tag{16}$$

Para melhor apresentar os resultados eles foram divididos em quatro partes, uma para cada grupo de instâncias. Sendo o Grupo 1 listada as instâncias onde o solver e as heurísticas obtiveram as soluções ótimas, Grupo 2 onde o solver e pelo menos 1 das heurísticas obtiveram a solução ótima, Grupo 3 lista as instâncias que o solver não obteve as soluções ótima, mas obteve alguma solução no tempo limite e as soluções obtidas pelas

heurísticas, finalmente o grupo 4 lista as instâncias onde o solver não conseguiu obter nenhuma solução e as soluções encontradas pelas heurísticas.

Os resultados computacionais estão representados em tabelas para cada um dos grupos, contendo os seguintes dados I sendo a qual instância aquele resultado experimental é referente, |V| sendo o número de veículos usados para resolver aquela instância, |N| é o número de pontos que aquela instância tem, Distancia Gurobi, Distancia ACO e Distancia TS sendo respectivamente a distância obtida pelo solver Gurobi, heurística ACO e heurística TS, Tempo G, Tempo ACO e Tempo TS sendo o tempo gasto respectivamente por cada pelo solver e pelas heurísticas e $gapGx$ACO, $gapGxTS$ e gapACOxTS sendo o respectivamente o erro relativo entre o solver Gurobi e heurística ACO, solver Gurobi e a heurística TS e entre as heurísticas ACO e TS.

6.1 Grupo 1

As instâncias de representadas no Grupo 1, tanto o solver quanto as heurísticas obtiveram a solução ótima abaixo do tempo limite.

Todas as 3 abordagens chegaram à solução ótima. A Tabela 6 mostra os resultados, pode ser observado que o as colunas do *gap* não foram apresentadas, desde que todas as abordagens chegaram na solução ótima este valor é 0. A Busca Tabu conseguiu obter a solução ótima em um tempo melhor na instância 4-f, foi o tempo melhor que 0,9 segundos em relação ao ACO. O Gurobi obteve um tempo médio de 760,38 segundos para obter a solução, a Busca Tabu ficou com um tempo médio de 99,15 segundos para obter a solução e o ACO ficou com um tempo médio de 79,35 segundos para obter uma solução, logo pode ser observado que mesmo a TS ficando melhor em tempo na instância 4-f o ACO foi melhor em todas as outras 5 instâncias do Grupo 1 e ficou com uma média inferior as outras abordagens.

Tabela 6 - Resultados Grupo 1

I	|V|	|N|	Gurobi		TS		ACO	
			Distância	Tempo (s)	Distância	Tempo (s)	Distância	Tempo (s)
1-f	3	10	**13450**	26,3	**13.450**	6,8	**13.450**	**3,1**
2-f	3	20	**32698**	68,6	**32.698**	26,5	**32.698**	**16,5**
3-f	3	50	**98458**	156,8	**98.458**	42,6	**98.458**	**29,2**
4-f	7	90	**268452**	590,55	**268.452**	**68,4**	**268.452**	69,3
5-f	7	130	**626450**	1286,31	**626.450**	196,1	**626.450**	**136,5**
6-f	7	279	**621140**	2433,4	**621.140**	254,5	**621.140**	**221,5**
MÉDIA DE TEMPO				760,32		99,15		**79,35**

O Gráfico 1 exibe a relação entre o crescimento no número de clientes e o tempo gasto para obter a solução ótima que cada algoritmos. Sendo T. Gurobi o tempo gasto pelo solver, T. ACO o tempo gasto pela heurística ACO e T. Tabu o tempo gasto pela heurística TS.

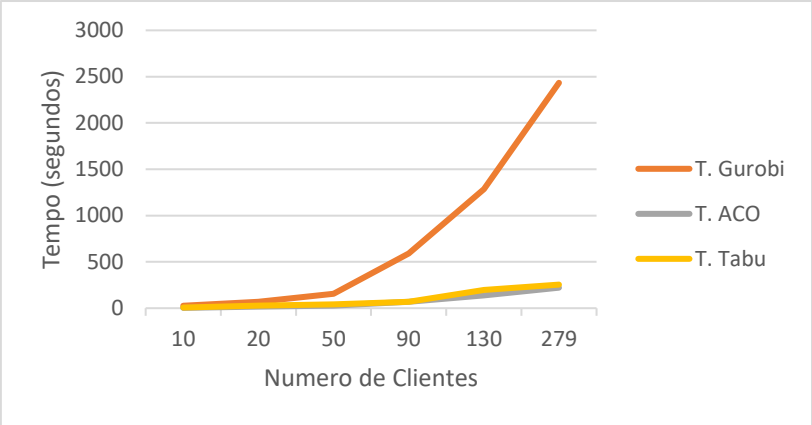


Gráfico 1 - Relação entre quantidade pontos e tempo de execução - Grupo 1

6.2 Grupo 2

O Grupo 2 é formado pela única instância que o solver Gurobi e, pelo menos, 1 heurística obtiveram a solução ótima. Mostrado na Tabela 9, pode ser observado que o solver e a heurística ACO obtiveram a solução ótima, mas a heurística TS não atingiu a solução ótima ficando com um *gap* de 7,80% em relação ao solver e o ACO. A heurística TS parou pois o critério de número máximo de iterações sem melhora foi atingido. Neste caso é provável que o algoritmo TS não tenha conseguido sair de um ótimo local.

Tabela 7 - resultados grupo 2

I	**|V|**	**|N|**	**Gurobi**		**TS**		**ACO**		***gap***	
			Distância	**Tempo**	**Distância**	**Tempo**	**Distância**	**Tempo**	**Gurobi TS**	**ACO TS**
1-s	8	290	**540586**	3150	586325	392	**540586**	**365**	7,80%	7,80%

O ACO conseguiu obter a solução ótima em menor tempo que o Gurobi, ficando mais de 8 vezes mais rápido que o solver.

6.2.1 Grupo 3

Neste grupo estão as instâncias onde o solver Gurobi não obteve a solução ótima, mas conseguiu obter alguma solução no tempo de execução permitido definida na Seção 5.2.1.

Como o Gurobi não atingiu a solução ótima as duas heurísticas foram executadas durante todo seu tempo de execução permitida descritas nas Secções 5.2.2 e 5.2.3 e ambas as heurísticas obtiveram o mesmo resultado. Mostrado na Tabela 8, sendo que não apresenta a coluna de tempo, já que todas as abordagens utilizaram todo o tempo limite disponível (3600 segundos). Dado que as heurísticas atingiram a mesma resposta, a fim de verificar se a solução obtida por elas é a ótima, foi executado o solver sem restrição de tempo, provando que os valores obtidos pelas heurísticas nas instâncias 2-s e 3-s são a solução ótima, o solver gastou respectivamente 8280 segundos (duas horas e trinta

minutos) e 15840 segundos (quatro horas e quarenta minutos) para se obter a solução. Logo é possível observar o desempenho das heurísticas em relação ao solver.

Tabela 8 - resultados grupo 3

I	|V|	|N|	Gurobi	TS	ACO	*gap*	
			Distância	Distância	Distância	Gurobi ACO	Gurobi TS
2-s	9	350	765041	698156	698156	9,58%	9,58%
3-s	10	558	978544	932014	932014	4,99%	4,99%

6.2.2 Grupo 4

Para as instâncias de tamanhos maiores, o solver Gurobi não conseguiu encontrar nenhuma solução dentro do tempo limite, mas ambas as heurísticas obtiveram resultados dentro do tempo limite estabelecido. A Tabela 9 apresenta os resultados obtidas pelo TS e pelo ACO, sendo não apresentada a coluna de tempo, visto que as heurísticas foram executadas até 3600 segundos e o solver não foi representado na tabela, dado que ele não conseguiu obter nenhuma solução.

Em relação as soluções obtidas pelas heurísticas o ACO obteve uma melhora na distância total percorrida em relação a heurística TS. Tendo um *gap* médio 5,33% de erro entre a solução da Busca Tabu.

Tabela 9 - resultados grupo 4

I	|V|	|N|	TS	ACO	*gap*
			Distância	Distância	ACO TS
1-r	15	1000	1735481	1589457	8,414
2-r	15	2500	4935481	4578963	7,223
3-r	15	4166	7836541	7548325	3,677

7 CONCLUSÃO

Neste trabalho foi abordado um problema de OC, chamado de Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (VRPSPD), este problema visa minimizar a distância total percorrida pelos veículos para que sejam satisfeitas as demandas dos clientes, tanto para coleta, quanto para entrega. O trabalho de MONTANÉ, et al., (2006) propõe solucionar o VRPSPD utilizando a solução via solver e o algoritmo de Busca Tabu. Este trabalho propõe utilizar o ACO para solucionar o VRPSPD e comparar seus resultados com as duas abordagens propostas por MONTANÉ, et al., (2006) sendo que a formulação matemática do MONTANÉ, et al., (2006) foi implementada utilizando o solver CPLEX e este trabalho propõe utilizar o Solver Gurobi, dado que ambos *solvers* são comerciais o Gurobi possui uma licença para estudantes acessível.

Como não foram encontradas as instâncias utilizadas por MONTANÉ, et al., (2006). Este trabalho propõe coletar os dados que foram utilizados, todos os foram coletados a partir de aplicativos de mapas disponíveis e livres.

Na comparação, entre os resultados obtidos, vê-se que o ACO mostrou melhores resultados em 11 das 12 instâncias. E apresentou-se melhor que o Gurobi em todos os casos que o solver consegue a solução ótima. Em relação ao TS o ACO em nenhuma das instâncias ficou pior que ele, ficando com valores iguais ou melhores em diversos casos.

Para trabalhos futuros podem ser abordados diferentes tipos de heurísticas para comparar os resultados obtidos nas heurísticas ACO e TS. A possível hibridização de heurísticas e flexibilizar os parâmetros do Gurobi para experimentos acadêmicos.

8 REFERÊNCIAS

D.NELSON, MARVIN, et al. 1985. Implementation techniques for the vehicle routing problem. *elsevier.* 1985, Vol. 12, 5.

DANTZIG, GB e RAMSER, JH. 1959. The truck dispatching problem. *Management Science.* 1959, 6, pp. 80-91.

DELL'AMICO, MAURO, RIGHINI, GIOVANNI e SALANI, MATTEO. 2003. *A branch-and-price algorithm for the vehicle routing problem with simultaneous pick-up and delivery.* s.l. : Dipartimento di Scienze e Metodi per l'Ingegneria - Universit`a di Modena e Reggio Emilia, 2003.

DETHLOFF, J. 2001. Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up. *OR Spektrum.* 2001, 23.

DISTANCEMATRIX. DISTANCEMATRIX. *DISTANCEMATRIX.* [Online] DISTANCEMATRIX. https://distancematrix.ai/nonprofit.

DORIGO, M. 1992. *Optimization, Learning and Natural Algorithms.* Milão : Politecnico di Milano, 1992.

GENTILE, MARIANO e SWOBODA, NIK. 2015. *A Theoretical Consideration of the parameters of the Max Min Ant System* . s.l. : Universidad Politécnica de Madrid - Departamento de Inteligencia Articial, 2015.

GLEZ-PEÑA, D., et al. 2014. Web scraping technologies in an API world. *Briefings in bioinformatics.* 2014, Vol. 5, 15.

GLOVER, FRED. 1990. Tabu Search: A Tutorial. *Interfaces.* 1990.

GOLDEN, B BALL e BODIN, L. 1981. . Current and future research directions in network optimization. *Computers and Operations Research.* 1981, 8.

GOOGLE. API Distance Matrix. *Google maps.* [Online] GOOGLE. https://developers.google.com/maps/documentation/distance-matrix/.

—. My Maps. *google - MyMaps.* [Online] GOOGLE. https://www.google.com/intl/pt-BR/maps/about/mymaps/.

HERNÁNDEZ, CECILIA TOLEDO, MARINS, FERNANDO AUGUSTO SILVA e CASTRO, ROBERTO CESPÓN. 2012. Modelo de Gerenciamento da Logística Reversa. *Gest. Prod.* 2012, Vol. 3, 19.

MA, YANFANG, et al. 2017. An improved ACO for the multi-depot vehicle routing problem with time windows. *Proceedings of the Tenth International Conference on Management Science and Engineering Management.* 2017.

MALEK, M., et al. 1989. Serial and parallel search techniques for the traveling salesman problem. *Annals of OR: Linkages with Artificial Intelligence.* 1989.

MAPQUEST e AOL. mapquest. *mapquest.* [Online] AOL. https://www.mapquest.com/.

MIN, HOKEY. 1989. The multiple vehicle routing problem with simultaneous delivery and pick-up points. 1989, Vol. 23, 5.

MONTANÉ, Fermín Alfredo Tang e GALVAO, Roberto Diéguez. 2006. A tabu search algorithm for the vehicle routing problem with. *Computers & Operations Research.* 2006, 33.

NAGY, G e SALHI, S. 2005. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research.* 2005, 162.

OLIVEIRA, BRUNO ALMÊDA DE, et al. 2020. ALGORITMOS DE INTELIGÊNCIA COLETIVA PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS: REVISÃO. *SBPO.* 2020, Vol. LII.

OLIVEIRA, Marcio Mattos Borges de. 1991. ROTAS DE VEÍCULOS. *Instituto de Ciências Matemáticas e de Computação, University of São Paulo.* Master's Dissertation in Ciências de Computação e Matemática Computacional, 1991.

SANTOS, JORGE MENEZES DOS e RIBEIRO, ALEXANDRE. 2019. *HEURISTICAS PARA O PROBLEMA LEASING k-MEDIAN*. s.l. : RAG - PONTIF´ICIA UNIVERSIDADE CATOLICA DE GOI ´ AS´, 2019.

SOUZA, JOÃO PAULO SENA. 2015. *COMPARAÇÃO DOS MÉTODOS DE CLASSIFICAÇÃO POR ÂNGULO ESPECTRAL E DISTÂNCIA EUCLIDIANA NO MAPEAMENTO DAS FORMAS DE TERRENO.* BRASÍLIA : UnB, 2015.

SOUZA, MARCONE JAMILSON FREITAS. 2009. *Otimização Combinatória, Notas de aula.* s.l. : Departamento de Computação - Universidade Federal de Ouro Preto, 2009.

TOTH , P, e VIGO, D,. 2002. The vehicle routing problem. *SIAM.* 2002.

YAN, FANLEI. 2018. Autonomous vehicle routing problem solution based on artificial potential field with parallel ant colony optimization (ACO) algorithm. *Pattern recognition letters.* 2018, 116.

YEOWOON, JUN e KIM, BYUNG-IN. 2012. New best solutions to VRPSPD benchmark problems by a perturbation based algorithm. *elsevier.* 2012, Vol. 39, 5.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 • Setor Universitário
Caixa Postal 86 • CEP 74605-010
Goiânia • Goiás • Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br • reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante JOVÂNIO JOSÉ GALVÃO JÚNIOR do Curso de CIÊNCIA DA COMPUTAÇÃO, matrícula 2018.1.0028.0265-3, telefone: 62 9 82748416 e-mail jovaniojr3@gmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA SIMULTÂNEA COM DADOS COLETADOS A PARTIR DE APLICATIVO DE MAPA, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto(PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 17 de dezembro de 2022.

Assinatura do autor: Jovanio José Galvão Júnior

Nome completo do autor: JOVÂNIO JOSÉ GALVÃO JÚNIOR

Assinatura do professor–orientador: Alexandre Ribeiro

Nome completo do professor-orientador: Alexandre Ribeiro