UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Arthur Prince de Almeida

Problema de roteamento de veículos dinâmico
Aplicação nas instâncias da loggibud

São Paulo
2022

Arthur Prince de Almeida

Problema de roteamento de veículos dinâmico
Aplicação nas instâncias da loggibud

Monografia apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo, como parte dos requisitos exigidos na disciplina ACH 2017 – Projeto Supervisionado ou de Graduação I, para obtenção do título de Bacharelado em Sistemas de Informação.
Modalidade: TCC curto (1 semestre) – individual.

Orientadora: Karla Roberta Pereira Sampaio Lima

São Paulo
2022

Resumo

Prince, Arthur de almeida. **Problema de roteamento de veículos dinâmico:** Aplicação nas instâncias da loggibud. 2022. 16 f. Monografia (Bacharelado em Sistemas de Informação) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, 11/07/2022.

A logística por trás da alocação de veículos para entrega de pacotes está sujeita a uma explosão combinatória de possibilidades. O problema de roteamento de veículos é uma área de estudo de programação inteira que busca encontrar soluções satisfatórias em tempo viável. O trabalho em questão busca estudar o problema de roteamento de veículos dinâmico, o problema que exige o menor tempo de execução possível. Seu contexto é nas instâncias da loggibud, o qual se refere a uma base de dados conhecida da empresa loggi para testes de benchmark de algoritmos de roteamento. Tendo isso em vista, este trabalho aborda sobre classificação de problema de roteamento de veículos, o problema da loggi e suas peculiaridade. E por fim propõe um algoritmo baseado na meta-heurística Simulated annealing e em grupos naturais para resolução dinâmica e estocástica do problema.

Palavras-chaves: Problema Dinâmico de Roteamento de Veículos, Roteamento Online, Simulated annealing, Kmeans. Loggibud

Lista de figuras

Lista de algoritmos

Lista de abreviaturas e siglas

PRV	Problema de roteamento de veículos
PRVC	Problema de roteamento de veículos capacitados
PRVCE	Problema de roteamento de veículos com coleta e entrega
PRVJT	Problema de roteamento de veículos com janela de tempo
PRVD	Problema de roteamento de veículos dinâmico
PRVE	Problema de roteamento de veículos estocástico
PRVDE	Problema de roteamento de veículos dinâmico e estocástico

Sumário

1 Introdução

O aumento da internet em nosso cotidiano e a facilidade de fazer pagamentos online criaram um ambiente favorável para o comércio eletrônico. As estruturas de varejo globais estão ganhando cada vez mais espaço nesse ambiente. Todavia, esse novo mercado trouxe novos desafios para a logística de entrega de produtos.

O problema de roteamento de veículo – PRV (vehicle routing problem) é um dos problemas mais estudados na área da computação. Em uma das suas instâncias mais simples está o famoso problema do caixeiro viajante (traveling salesman problem), o qual consiste em traçar a rota com menor distância de modo que o veículo passe por diversos pontos especificados no problema. Já em sua instância mais simples fala-se de um problema np-difícil. Portanto, é muito comum na literatura encontrar algoritmos que usam heurísticas ou métodos de programação inteira.

Loggi é uma empresa que trabalha com a logística das entregas baseada em um modelo de economia compartilhada como Uber, Rappi e Ifood. Entregadores independentes usam a plataforma Loggi online para fazer entregas de última milha em seus próprios veículos. O maior desafio da Loggi é criar uma grande rede logística sustentável para entregas de produtos no dia que foi encomendado em todo Brasil.

O objetivo deste trabalho é estudar técnicas de roteamento de veículos usando as instâncias de dados disponibilizadas pela loggi, a loggibud. Para isso é necessário primeiramente caracterizar os tipos de PRV presente dentro do problema. Em seguida, estudar a base de dados e a API disponibilizado pela loggi para resolver o problema. E por fim propor uma solução.

2 Revisão bibliográfica

2.1 Representação matemática do problema de roteamento de veículo estático

É dito que um PRV é estático quando é conhecido todas as informações das entregas previamente. O cálculo de sua solução ótima é praticamente inviável devido ao seu crescimento de complexidade exponencial. Uma das possíveis técnicas para resolvê-lo poderia ser por um sistema de equações lineares inteiras. Portanto, essa sessão será focada na formulação matemática do problema, que além de ser suficiente para a resolução do problema estático também, é necessário para a compreensão do problema.

"A importância e influência do modo de formular um problema de otimização, especialmente em áreas complexas como as de roteamento, recobrimento etc. devem ser bem entendidas. O motivo é evidente: a formulação terá impacto direto no desempenho dos algoritmos de solução." (Goldbarg e Luna, 2000).

O problema da loggibud pode ser formulado da seguinte maneira: um conjunto de veículos idênticos V, com capacidade q, necessitam realizar entregas de pacotes em uma região, representada por um grafo direcionado G. O grafo G consiste em |C|+1 vértice, em que os consumidores a serem visitados são representados pelo conjunto $C = \{c_1, c_2, ..., c_n\}$, e o depósito é representado por c0. Logo o conjunto de vértices do grafo G é representado por $N = C \cup \{c_0\}$. Já as arestas representam a distância entre cada consumidor e estão associadas a um custo d_{ij}. Por fim, cada pacote tem um peso w_i associado a um consumidor c_i

O modelo tem uma variável de decisão x definida como:

$$x_{ijk} = \begin{cases} 1 \text{ se o veículo } k \text{ percorre a aresta}(i, j) \\ 0 \text{ se o veículo } k \text{ não percorre a aresta}(i, j) \end{cases}$$

Portanto o modelo é descrito da seguinte maneira:

$$(1) \quad objetivo: \ min \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} d_{ij} x_{ijk} \ sujeito \ a$$

$$(2) \quad \sum_{k \in V} \sum_{j \in N} x_{ijk} = 1, \ \forall \ i \in C$$

$$(3) \quad \sum_{i \in C} w_i \sum_{j \in N} x_{ijk} \leq q, \ \forall \ k \in V$$

$$(4) \quad \sum_{j \in C} x_{0jk} = 1, \ \forall \ k \in V$$

$$(5) \quad \sum_{k \in V} \sum_{(i,j) \in S, i \neq j} x_{ijk} \leq |S| - 1, \ \forall \ S \subseteq C$$

$$(6) \quad \sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0, \ \forall \ k \in V \ ; \ \forall \ h \in C$$

$$(7) \quad x_{ijk} \in \{0, 1\}, \qquad \forall \ k \in V \ ; \ \forall \ i, j \in N$$

A função objetivo do problema (1) busca minimizar a soma de todas as distâncias percorrida por todos os veículos. A restrição (2) diz que todo consumidor precisa ser visitado exatamente uma vez. A restrição (3) é a restrição de carga em que cada veículo não pode transportar pacotes cuja soma é maior que sua capacidade. A restrição (4) diz que todos os veículos saem do deposito. A restrição (5) diz que para qualquer subconjunto de vértices existe menos arestas do que vértices. Essa restrição é aplicada para PRV abertos e garante que não ocorra ciclos. Por fim a restrição (6) diz que todo veículo que entre no consumidor h também sai de h.

2.2 Principais classes do problema de roteamento de veículos

O PRV é um problema que estuda a logística de entregas veiculares proposto inicialmente por Dantzig e Ramser (1959). De forma geral, seu objetivo é otimização de rotas para coleta e entrega de pessoas ou mercadorias. Devido à quantidade de problemas práticos envolvendo o roteamento de veículos, sua definição vem sendo estudada e estendida ao longo das últimas décadas. A partir disso, surgiram várias subcategorias do PRV, mudando assim suas restrições ou até mesmo sua função objetivo. Nessa sessão será abordado as principais dessas.

Existem diferentes definições de PRVs na literatura, no entanto, todos os autores concordam que é possível separá-lo em três grandes classes: PRVC, PRVJT e PRVCE. A primeira é conhecida como PRV capacitado (PRVC), sua definição surge como a generalização do

problema mais estudado em otimização combinatória, o problema do caixeiro viajante. O PRVC foi descrito inicialmente por Dantzig e Ramser (1959) de forma que, considera-se um depósito central e uma frota de veículos idênticos com capacidade máxima, a fim de atender um conjunto de consumidores que demandam o serviço de entrega ou coleta de mercadorias. Assim, seu objetivo é atender todos os consumidores de modo que minimize a soma total das distâncias de cada veículo. Pode-se ver uma representação na Figura 1.

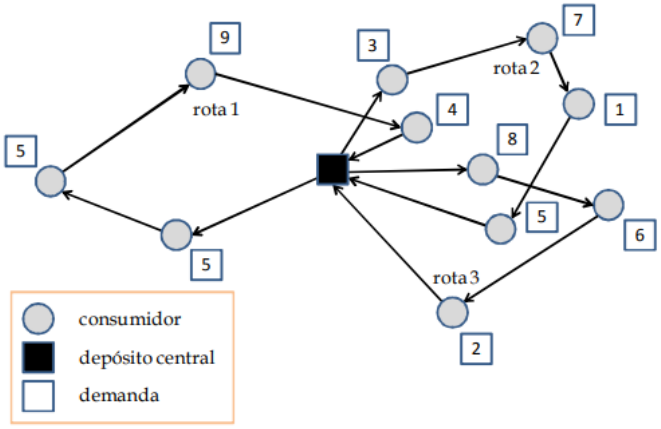

Figura 1: exemplo de PRV

Fonte: Humberto Brandão (2011)

Neste problema todos os veículos iniciam sua rota no depósito com uma carga suficiente para atender as demandas dos consumidores presentes em sua rota. Em seguida passam por todos os consumidores na ordem prevista, e por fim voltam para o depósito. Em algumas instâncias do PRVC, dependendo do número de veículos disponíveis, é possível não existir uma solução. Isso ocorre devido a restrição de carga, e para evitar isso muitas das vezes usa-se um peso adicionar um novo veículo na função objetivo.

Já o PRV com janela de tempo (PRVJT) apresenta todas as características do PRVC, no entanto com um fator de tempo. Essa abordagem considera que há uma janela de tempo para o atendimento do consumidor. Sendo que permite atender a demanda antes da abertura da janela de tempo, entretanto não admite após seu fechamento, ou seja, com atraso. Nesse problema há mais três grandes classes que são caracterizadas pelo período de tempo para a resolução do algoritmo encontrar uma solução, sendo eles o urgente, o repetitivo e o de projeto. O primeiro, como o nome diz, tem caráter de urgência, ou seja, essa classe deve estar preparada para responder rápido em mudanças do ambiente. A segunda normalmente utiliza uma estratégia de definir um subconjunto de demandas e em seguida resolve-os utilizando o algoritmo estático. Por exemplo, os pacotes dos consumidores que chegam de hora em hora são roteados

separadamente. Finalmente o de projeto buscam soluções com mais tempo de processamento disponível, no entanto não pertencem a classe de VRPs

O VRPJT tem uma propriedade interessantes que, diferentemente dos outros VRPs, a classifica como um problema np-completo. Isso implica que, caso se prove que existe um algoritmo em tempo polinomial para o VRPJT, então todos os problemas da classe np-difícil também poderão ser resolvidos em tempo polinomial.

Por fim, o PRV de coleta e entrega (PRVCE) considera que, diferente do PRVC e o PRVJT, ocorre simultaneamente os dois tipos de serviços: entrega e coleta de mercadorias. Neste problema, cada cliente pode fazer, ao mesmo tempo, os dois tipos de pedidos, e cada veículo transporta uma mistura de pedidos de entrega e coleta.

2.3 Problema de roteamento de veículo dinâmico e estocástico

O PRV dinâmico (PRVD) está dentro da categoria dos PRVJT com urgência, como dito anteriormente. Sua principal característica é que trabalha em um ambiente com constantes mudanças. Isso significa que, novos consumidores aparecem e são deletados a todo momento; alguns caminhos podem ser bloqueados; o veículo passa a não poder exercer a coleta de produtos porque quebrou; e o custo de rotas também podem variar. Portanto, o tempo de processamento para o PRVD é essencial, assim as abordagens mais comuns para resolver esse problema na literatura são com uso de heurísticas ou meta-heurísticas.

A grande dificuldade de trabalhar com o PRVD é que mesmo que o algoritmo encontre a melhor solução para uma determinada instância em um tempo polinomial, é possível que após a inserção de um novo consumidor todas as rotas mudem. Por exemplo, uma solução ótima pode levar o veículo ao lado norte da cidade, em seguida surge um novo consumidor que faz com que a solução ótima exija que esse veículo seja levado para o lado sul. Visando controlar essa situação, uma boa estratégia seria criar uma cobertura veicular sobre o território. No entanto, tal estratégia pode aumentar a distância total percorrida. Para auxiliar tal cobertura seria interessante observar o histórico dos clientes.

O PRV estocástico (PRVE) é uma forma de trabalhar com PRVs tendo alguns tipos de dinamismo de maneira antecipada. Nesta classe de problemas, diferentemente do PRVD, existe uma probabilidade bem definida sobre os eventos futuros que são disponibilizados para o algoritmo de roteamento. Esta classe de problemas é inspirada no Problema do Caixeiro Viajante Probabilístico proposto por Jaillet (1985), em que cada consumidor possui uma

probabilidade de requisitar um serviço. O objetivo é encontrar uma rota que minimize a distância total percorrida.

3 O problema da Loggi

No início da loggi, a solução usada para entregar seus produtos foi o PRV estático. No entanto, devido ao seu crescimento, esse modelo se tornou insustentável. Assim, o artigo abordará uma solução aplicada em big data, mas antes vamos ver os detalhes do problema

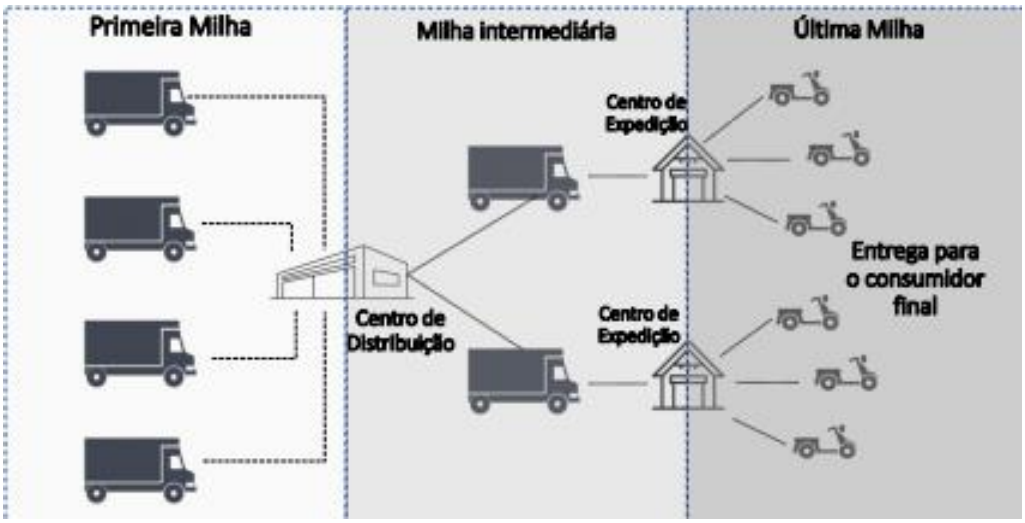

Figura 2: divisão de milhas

Hoje a loggi trabalha com um sistema logístico dividido em três partes. A primeira, conhecida como a primeira milha, é a etapa em que os pacotes encomendados são levados para o centro de distribuição. Neste lugar separa-se os pacotes por diferentes critérios como prioridade, sensibilidade, e a localização. Em seguida, na milha intermediária, os pacotes são transferidos para os centros de expedições. Nessa etapa o transporte é feito por caminhões ou aviões, dependendo da localização do consumidor. Por fim, na última milha, uma máquina chamada de sorter agrupa os pacotes em sacolas de forma dinâmica, de modo que quando o pacote chega a este, direciona-o a sua sacola. Após carregadas, as sacolas são destinadas a um lugar ao qual os entregadores têm acesso. E finalmente os entregadores entregam os produtos seguindo as rotas feitas pela loggi. A divisão de milhas está exemplificada na Figura 2.

O desafio é dividir os pacotes em sacolas de modo que os entregadores entreguem o máximo de pacotes possíveis percorrendo a menor distância, no entanto há dois problemas. O primeiro

é que a solução ótima pode não ser a mais justa tendo em vista que alguns entregadores poderão percorrer uma longa distância para entregar poucos pacotes e outros percorrerão uma distância curta com muitos pacotes. O segundo problema é que, o sorter é uma esteira que divide os pacotes instantaneamente. Ou seja, os pacotes são direcionados para suas sacolas assim que são colocados no sorter. Isso faz com que não haja acúmulo de informações no PRV, portanto uma abordagem que leva o passado em consideração é recomendada.

Como explicado no modelo estático, pode-se resolver o PRV com um modelo de equações lineares inteiras. Com o aumento da quantidade de pacotes, esse processo se torna custoso e operacionalmente inviável, haja vista que se deve ter a informação de todas as entregas antes de serem computadas. Ou seja, enquanto se calcula a melhor rota para uma certa quantidade de pacotes, ocorre a chegada de novos pacotes, fazendo assim com que a loggi não cumpra seu objetivo de entregar seus pacotes em menos de um dia.

A proposta deste artigo é propor uma solução as instâncias da loggibud de tal modo que o algoritmo tenha alto grau de dinamicidade, podendo até considerá-lo um problema online. Ou seja, a rota é traçada enquanto o pacote chega.

Todavia as instâncias da loggibud estão caracterizadas em diversos tipos de PRV sendo elas:

- Open vehicle routing problem (OVRP) - significa que o veículo não precisam se preocupar com a voltar para o depósito já que os entregadores são terceirizados.
- Capacitated vehicle routing problem (CVRP) – cada veículo tem uma capacidade máxima e cada pacote tem um peso.
- Vehicle routing problem multi-depot (VRPMD) – as rotas são formadas no centro de distribuição. E nos centros de expedições apenas segue o modelo já feito ou este é otimizado.
- Dynamic vehicle routing problem (DVRP) – isso implica que o problema deve ser resolvido de forma dinâmica e online. Ou seja, assim que o pacote chegar no centro de expedição deve-se direcioná-lo a um veículo.
- Vehicle routing problem stochastic customers – haja vista que podemos usar um modelo de agrupamento com base nos destinos dos pacotes no passado. Além disso, deve-se tratar esse problema como big-data devido à complexidade do território brasileiro com mais de 5700 municípios.

No caso da Loggi, é preferível que o centro de expedição não tenha conhecimento da milha intermediária. Pois o algoritmo teria que tratar imprevistos como se o caminhão quebrasse ou atrasasse, assim gastando processamento desnecessário. Portanto o artigo trabalhará apenas com o problema da última milha. De forma mais específica o problema considera que o veículo de entrega tenha uma capacidade pequena e constante (180 de volume). E Segue as seguintes etapas exemplificada na figura 3

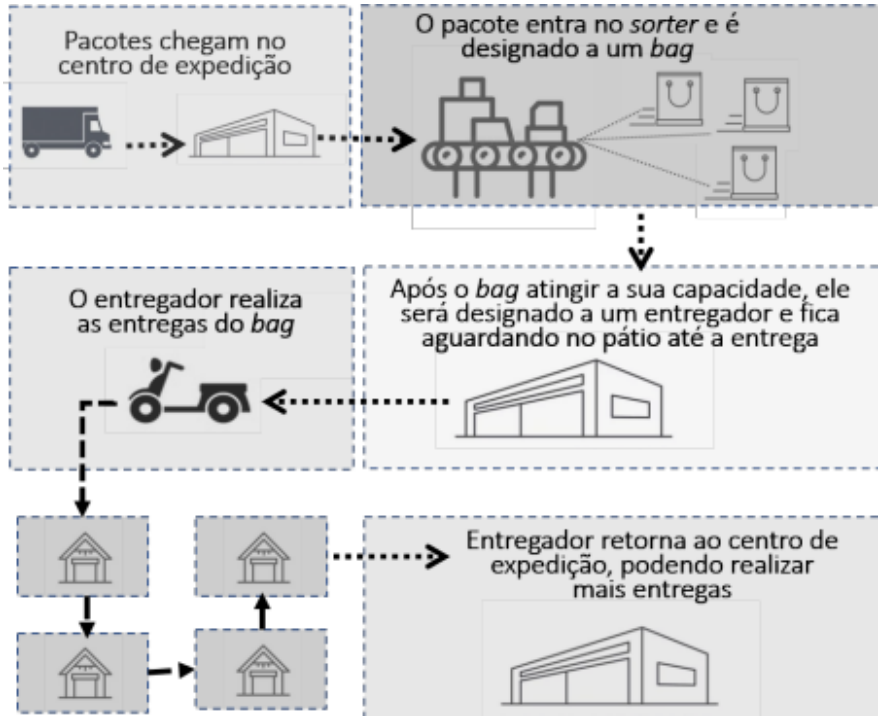

Figura 3: problema de última milha

1. Os pacotes chegam a todo momento de forma online, ou seja, só se tem conhecimento do pacote quando ele está no centro de expedição.

2.Quando os pacotes chegam em um centro de expedição, são separados em sacolas denominados “bag” por um aparelho chamado de “sorter”.

3. Após a sacola atingir a capacidade máxima, elas são dirigidas a um local onde ficarão esperando ser entregues para o entregador.

4. O entregador pega a sacola e faz as entregas dos pacotes e volta para o centro de expedição para pegar uma nova sacola.

Além da informação do sorter é sabido que “O problema de roteamento de veículos denso tem propriedades territoriais fortes” (Surita, 2019). Isso significa que é possível, com base nos acontecimentos do passado, dividir o território em grupos (clusters). Ou seja, pode-se diminuir o espaço de busca de rotas olhando apenas para sub-regiões do território e posteriormente traçar rotas.

4 Explicação do algoritmo

4.1 Introdução

O algoritmo se baseia em 2 etapas: a de pré-processamento e a de execução. A parte do pré-processamento foi implementada pelo curso introdutório da loggi, para o desenvolvimento de solução de PRV em sua API. Sua implementação usa como base apenas o K-means para geração de agrupamentos, ao qual o pacote pertencerá em função de sua localização. Já a etapa de execução é baseada no algoritmo híbrido proposto no trabalho de doutorado do Humberto Brandão (2011). O qual utiliza o simulated annealing para encontrar boas rotas.

4.2 Pré-processamento

O pré-processamento tem o objetivo de diminuir o espaço de busca de rotas, visto que além de normalmente os veículos serem direcionados para locais com clientes próximos entre eles, é necessário criar uma cobertura veicular para PRVD como discutido nas sessões anteriores. No problema em questão é tido mais de 1000 pacotes por instância. Por essa razão é importante que seja possível agilizar o processo com algum método de predição. No caso do artigo, foi percebido que, quando resolvidos com o modelo estático, havia recorrência de rotas em determinadas regiões no período de um dia. Devido a esse fato um bom fator para diminuir o espaço de busca seria criar grupos pelo critério do território.

O algoritmo de agrupamento escolhido foi o “K-means” por três razões. Primeiramente, esse é um algoritmo simples baseado em centroides, ou seja, define-se alguns pontos no território que representarão a região, e para saber se tal pacote pertence àquela região basta verificar a distância do pacote para todos os centroides. Isso deixa o algoritmo mais rápido tendo em vista que existem poucos centroides que representam muitos pacotes. O segundo

motivo é que a loggi havia implementado apenas esse algoritmo para definir qual pacote seria colocado na sacola e obteve um bom resultado. Assim pode-se utilizar código pronto e reavaliar os resultados. E a última razão é que é possível associar cada centroide com uma sacola, ou seja, todo pacote que estiver na rota de determinado centroide será associado a sacola que representa aquele centroide.

Para modelar o “K-means”, foi criado K pontos aleatórios no mapa da região. Onde K é uma escolha arbitrária que depende da região. Assim, para se fazer o pré-processamento, foi utilizado a base de dados de treinamento, e para se obter as distâncias foi usado o software “openStreetMaps”. A entrada em questão foi o valor de K e o nome de um arquivo que representa os pacotes entregues por um centro de expedição no passado. A saída são K pontos com longitude e latitude de modo que cada ponto é um centroide no mapa. A Figura 4 mostra três centroides, representados pela cor preta, um deposito, representado pela vermelha, e vários clientes, representados pela cor azul.

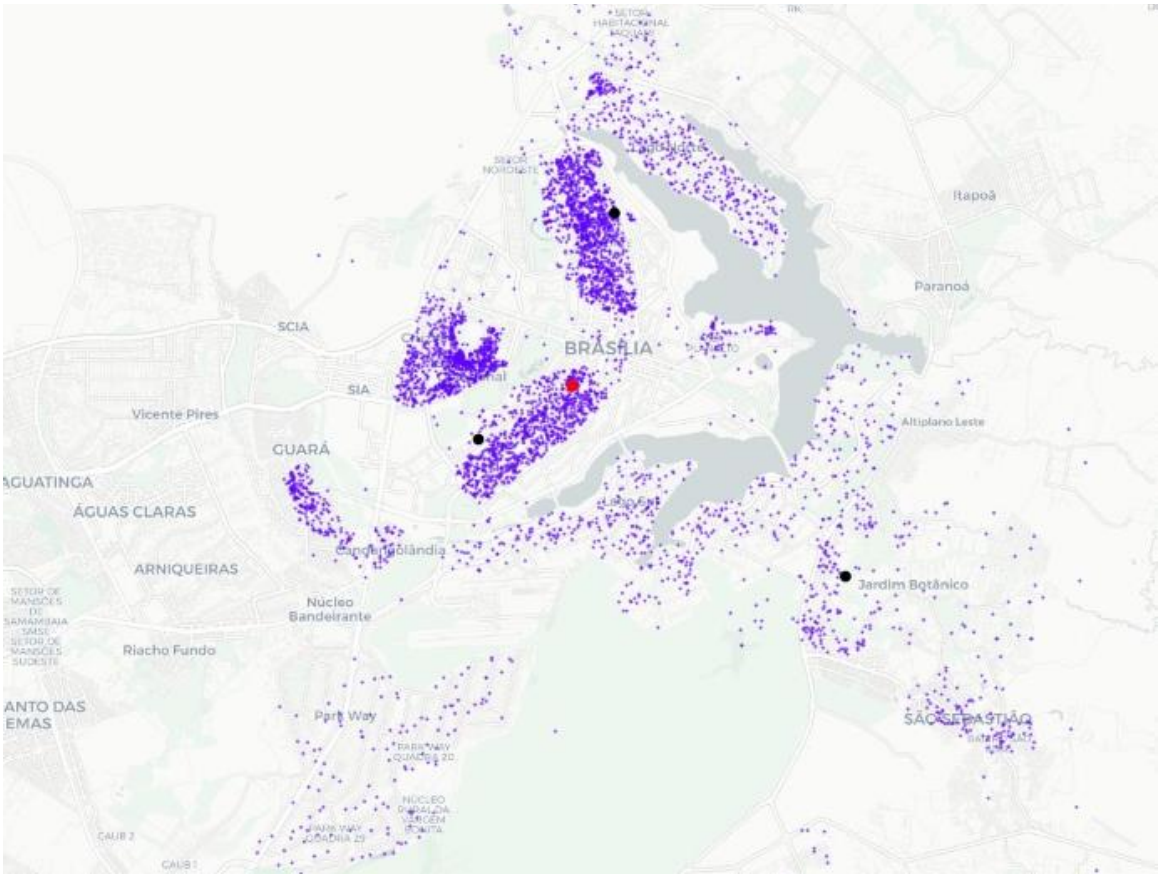

Figura 4: Exemplo de K-means

Como a proposta deste artigo é melhorar o algoritmo da loggi, foi criado um grafo que liga os centroides mais próximos. Esse grafo tem a característica de trabalhar com vizinhança entre regiões, assim, na parte da execução o espaço de busca será o centroide mais próximo da

sacola selecionada e os centroides vizinhos. Desse modo o pacote que é representado por determinado centroide pode ir para outra sacola. Na verdade, o nome desse problema é procura por grupos naturais, que consiste em encontrar fronteiras em aglomerado de dados, no entanto, é criado regiões de intersecção entre os vizinhos. A ideia é tentar predizer qual será a melhor sacola para os pacotes de acordo com as rotas formadas, ao invés de deixar apenas a região determinar seu destino.

O algoritmo escolhido para determinar a vizinhança foi o algoritmo de Prim, o qual forma a árvore geradora mínima. Além disso, como o grafo é direcionado, já que a distância é medida em distâncias de ruas é necessário executar para todos os centroides. Então basta escolher a profundidade no qual será feita a busca. No algoritmo em questão foi usado profundidade = 1, ou seja, apenas os centroides vizinhos farão parte do espaço de busca na execução.

Pré-processamento

```
FazGrafoDeRegiões( pacotesTrain, NumRegioes )
// executa o kmeans passando as instancias de treino
CentrosMedios <- Kmean(pacotesTrain, NumRegioes)

//cria a matriz de distancia
Mdist <- FazMatrizDeDistancia(CentrosMedios)

//para cada ponto em CentroMedio
for origem <- CentroMedio

   //executa dijkstra para fazer a árvore de caminho mínimo onde origem é a raiz
   MenorCaminho <- Prim(origem, CentrosMedios, Mdist)

   //guarda em vizinho os vizinhos do ponto origem
   vizinho[origem] <- CentrosMedios(menorCaminho = 1)

end for

return vizinho
```

Algoritmo 1: pré-processamento com K-means e algoritmo de Prim

4.3 Execução

4.3.1 Simulated annealing

A parte da execução ocorre quando já há os metadados do pré-processamento. Sua função é, efetivamente, definir a sacola para o qual o pacote, que chega de forma dinâmica, será

enviado. O foco dessa parte é no algoritmo do sorter, então considera-se que, após a sacola encher, será executado um PRV estático com os pacotes dessa sacola. E por fim é entregue os pacotes aos entregadores.

Como o cálculo do PRV estático ainda é muito caro computacionalmente, uma abordagem usando heurísticas e meta-heurísticas é recomendada para agilizar o processo. O algoritmo implementado usa como base o simulated annealing proposto primeiramente por Kirkpatrick (1983). Esse algoritmo é uma meta-heurística probabilística que faz referência ao processo termodinâmico de resfriamento de átomos. Foi implantado uma variação desse algoritmo chamado de simulated annealing não-monotônica (SANM) como exemplificado no pseudocódigo abaixo.

Algoritmo de execução – inserir pacotes usando SANM

```
AdicionaPacotesNasSacolas(pacotesKs, sacolas)

sac' <- sacolas

for pac in pacotesKs

  //usa as rotas previamente feitas com os pacotes na sacola e em seguida adiciona pacotesKs usando o
  //PFIH
  s <- solucaoInicialPFIH(sac'.rotas, pacotesKs)
  t <- 5

  rotas <- [ ]

  while tempoDeExecucao < 3 segundos
    //redução do percentual da temperatura
    t <- t * 0.99

    if t < 0.1
      //reaquecimento
      t <- 5
    end if

    //retorna um vizinho de forma que os pacotes dentro das sacolas não saia da sacola
    s' <- solucaoNaVizinhancaDe(s, sac'.pacotes, pacotesKs)

    Δ = f(s') – f(s)

    if Δ < 0 ou numAleatorio() < e-Δ/t
      //troca solução corrente pela solução vizinha
      s <- s'

      rotas <- rotas + s'
    end if
  end while

  melhorRota <- PPCSolver(rotas)

  sac' <- sac' + melhorRota.get(pac)

end for

return sac'
```

Algoritmo 2: Algoritmo de execução – inserir pacotes usando SANM

A diferença entre o simulated annealing padrão e o SANM é que, para evitar os mínimos locais, o SANM reaquece (faz t<-5 quando t<0.1). Assim haverá uma probabilidade maior de aceitar soluções piores e então buscar soluções vizinhas melhores.

4.3.2 Push-Foward insertion heuristic (PFIH)

O algoritmo Push-forward insertion heuristic é uma heurística muito usada em PRV, ela foi introduzida no trabalho de Solomon (1987) e possui uma estratégia de construção de rotas eficiente com alta dinamicidade. Para explicar essa heurística, observe a imagem abaixo que representa a localização de um novo pacote (C 5) que ainda não foi roteado.

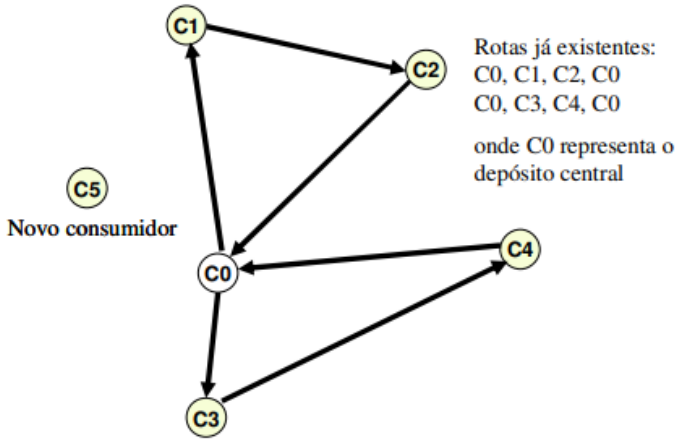

Figura 5: novo consumidor chega no sorter

Fonte: Humberto Brandão (2011)

O que o PFIH faz é olhar todas as possibilidades de onde inserir o pacote na rota(como mostrado na imagem abaixo) verificando se não há nenhuma violação de restrição. No final do algoritmo, é retornada a solução que tem a menor distância.

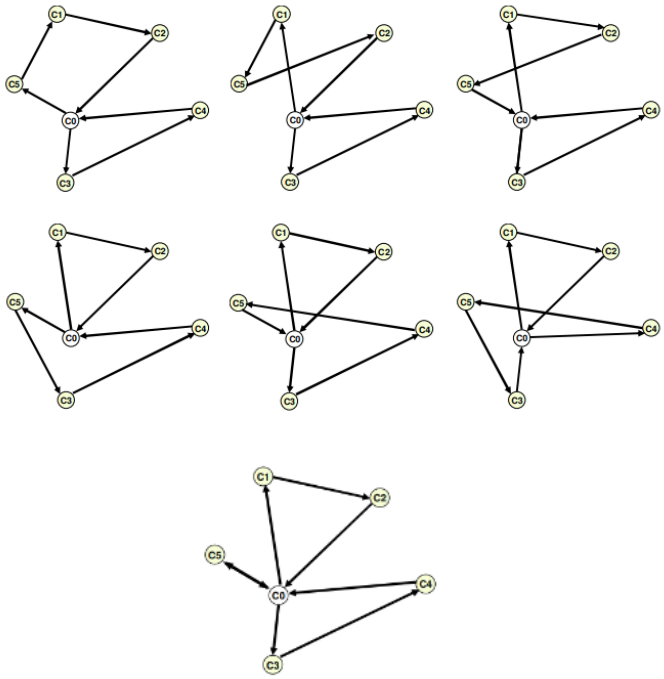

Figura 6: todas as possibilidades de inserção com PFIH

Fonte: Humberto Brandão (2011)

O problema do PFIH é que a ordem de inserção dos pacotes é importante o que a torna insuficiente para encontrar boas rotas. No entanto, essa heurística sempre retorna uma solução viável, por essa razão que ela é uma ótima candidata para gerar uma solução inicial no SANM.

4.3.3 Solução de vizinhança

Para navegar no espaço de busca será necessário levar em consideração que quando o pacote é colocado no sorter, pode-se destiná-lo para uma das sacolas ou para ser armazenado. No primeiro caso, o pacote é vinculado com a sacola, ou seja, o pacote que entra em uma sacola não pode sair dela. Isso faz com que os pacotes dentro da mesma sacola pertençam a mesma rota. No segundo caso, o pacote é guardado para ser roteado futuramente com algum critério estocástico de armazenamento e, futuramente, de seleção. No momento que o pacote é armazenado, adquire-se a informação de sua existência, e, portanto, esse pacote poderá ser roteado em momentos oportunos.

No algoritmo em questão, o critério de armazenamento depende do espaço no armazém e do número de sacolas, que é definido pelo número de centroides. Quando um pacote é colocado na esteira é atribuído a ele uma sacola, em seguida o algoritmo decide se irá armazená-lo ou roteiá-lo. Ou seja, define-se um armazém da sacola que cabe P pacotes, então a probabilidade dele ser armazenado será: (P-k)/P , onde k é o número de pacotes já armazenados. Caso contrário, todos os pacotes armazenados que pertencem à sacola do pacote atual e das sacolas vizinhas são selecionados para o roteamento.

O critério de armazenamento e de seleção é um ponto a ser melhorado, haja visto que não se considera a localização do cliente. Ou seja, não se analisa a probabilidade de um pacote pertencer a uma determinada rota ou se poderia ir para outra eventualmente, pelo contrário, o critério de armazenamento só depende do espaço livre no armazém e nem considera o peso do pacote, e o de seleção é completamente arbitrário já que envia todos os pacotes juntos.

Finalmente, para encontrar a solução da vizinhança, é sabido que existem dois tipos de pacotes, o pacote que será roteado e o pacote dentro da sacola. Este, por sua vez, não sai da sacola, então o possível vizinho pode ser gerado com um método bastante usado nos algoritmos genéticos que se baseia em trocar a ordem que os pacotes serão entregues. Observe que nesse

caso não há necessidade de verificar as restrições pois a carga não está sendo alterada. Já aquele, basta mudar a ordem dos pacotes em seguida usar PFIH.

A estrutura de dados que representa uma rota é uma lista ligada à qual o estoque se localiza na primeira posição, e em seguida a ordem que os pacotes serão entregues. Com isso, é possível fazer 4 operações nos pacotes dentro das sacolas: a troca, a inserção, o embaralhamento e a inversão.

- A aplicação do operador de troca em uma solução s ocorre de forma a permutar dois pacotes gerando assim a solução s'. Como é visto na Figura 7.

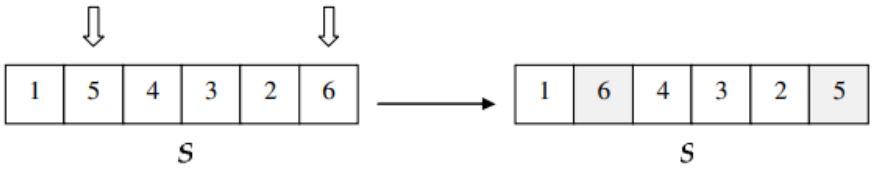

Figura 7: troca

Fonte: Humberto Brandão (2011)

- A aplicação do operador de inserção em uma solução S ocorre de forma a remover um pacote e em seguida adicioná-lo em outra posição gerando assim S'. Como visto na Figura 8.

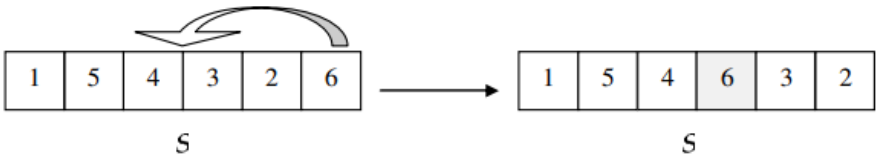

Figura 8: inserção

Fonte: Humberto Brandão (2011)

- A aplicação da operação de embaralhamento em uma solução S ocorre de forma em que se seleciona uma sub-rota q dentro de S, em seguida reordena q de forma aleatória gerando q'. Por fim insere-se q' onde ficava q formando a solução S' (Figura 9).

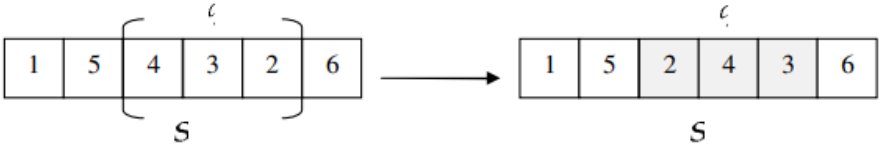

Figura 9: embaralhamento

Fonte: Humberto Brandão (2011)

- A aplicação da operação de inversão em uma solução S ocorre de forma em que se seleciona uma sub-rota q dentro de S, em seguida inverta q formando q'. Por fim insere-se q' onde ficava q gerando a solução S' (Figura 10).

Figura 10: inversão

Fonte: Humberto Brandão (2011)

5 Conclusão

O problema de roteamento de veículo é um problema abrangente que está evoluindo rapidamente. Haja vista que, a definição de Dantzig e Ramser (1959) se ramificaram em diversos tipos de PRVs como PRVC, PRVCE e o PRVJT, e esses se ramificaram ainda mais. O problema da loggi mostrou a alta complexidade dos PRV e a quantidade de possibilidades para se explorar. Portanto pode-se concluir que, apesar do algoritmo proposto ainda não ter sido testado, há ainda uma contribuição no que diz respeito a ideia de encontrar grupos naturais com intersecções entre si a fim de diminuir o espaço de busca. E em seguida usar técnicas de PRV dinâmico e estocástico com armazenamento de pacotes para definir a sacola que o pacote será destinado.

Apesar dos testes não terem sido concluídos, ainda há a intenção de continuar o projeto em uma iniciação cientifica no futuro. Grande parte do algoritmo já foi concluído, a implementação do algoritmo de Prim adaptado ao contexto, o K-means e a comunicação com a API da loggibud já estão funcionando. No entanto ainda há alguns problemas com o SANM, e além disso, os testes podem demorar horas devido a grande quantidade de dados, o que acaba impossibilitando seu término no período previsto para a entrega deste trabalho.

Referências bibliográficas

Brandão, H. algoritmo online para o problema dinâmico de roteamento de veículos. 147(Ciência da Computação) - Instituto de Ciências Exatas da Universidade Federal, Minas Gerais, 2011

Dantzig, G. B. & Ramser, J.H. The Truck Dispatching Problem. Management Science, 1959, Vol. 6, pp. 80-91

Goldbarg, M. C. & Luna, H. P. L. Otimização Combinatória e Programação Linear -- Modelos e algoritmos. Editora Campus, 2000

Jaillet, P. Probabilistic traveling salesman problems. Massachusetts Institute of Technology (MIT), 1985

Juan Camilo Fonseca-Galindo, Gabriela de Castro Surita, José Maia Neto, Cristiano Leite de Castro, and André Paim Lemos. 2020. A Multi-Agent System for Solving the Dynamic Capacitated Vehicle Routing Problem with Stochastic Customers using Trajectory Data Mining. arXiv:2009.12691 [cs.AI]

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. Optimization by simulated annealing. Science, 1983, Vol. 220, pp. 671-680

Lars M. Hvattum, Arne Løkketangen, and Gilbert Laporte. Solving a Dynamic and Stochastic Vehicle Routing Problem with a Sample Scenario Hedging Heuristic Transportation Science 2006 40:4, 421-438

Loggi. 2021. loggiBUD: Loggi Benchmark for Urban Deliveries. GitHub

Neves, Marcos Corrêa et al. Procedimentos automáticos e semi automáticos de regionalização por árvore geradora mínima. 2002.

Psaraftis, Harilaos N. Dynamic vehicle routing problems. Vehicle routing: Methods and studies, v. 16, p. 223-248, 1988.

Solomon, M. M. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research, INFORMS, 1987, Vol. 35(2), pp. 254-265