NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO
DE INFORMÁTICA E MATEMÁTICA

ANA SOFIA DA SILVA MENDES
Licenciada em Estatística Aplicada

PESQUISA TABU APLICADA A UM PROBLEMA DE ROTAS NA GALP

MESTRADO EM ANÁLISE E ENGENHARIA DE BIG DATA

Universidade NOVA de Lisboa
Novembro, 2021

DEPARTAMENTO
DE INFORMÁTICA E MATEMÁTICA

PESQUISA TABU APLICADA A UM PROBLEMA DE ROTAS NA GALP

ANA SOFIA DA SILVA MENDES

Licenciada em Estatística Aplicada

Orientadora: Paula Alexandra da Costa Amaral
Professora Associada, NOVA School of Science and Technology

Coorientador: José Miguel Espinosa Martinez
Head of Enterprise Architecture, Technology & IT Innovation, Galp

Júri:

Presidente: Doutor Pedro Manuel Corrêa Calvente de Barahona
Professor Catedrático, NOVA School of Science and Technology

Arguente: Doutor Nelson Fernando Chibeles Pereira Martins
Professor Auxiliar, NOVA School of Science and Technology

Orientador: Doutora Paula Alexandra da Costa Amaral
Professora Associada, NOVA School of Science and Technology

MESTRADO EM ANÁLISE E ENGENHARIA DE BIG DATA

Universidade NOVA de Lisboa
Novembro, 2021

Pesquisa Tabu aplicada a um problema de rotas na Galp

Agradecimentos

Em primeiro lugar gostaria de agradecer à professora Paula Amaral, orientadora desta dissertação pelo apoio, disponibilidade e incentivo prestados ao longo da realização deste trabalho.

Gostaria de agradecer à empresa Galp por permitir a realização deste projeto. Agradeço a todos elementos da equipa da Galp com que me cruzei, em especial ao José Miguel Espinosa, coorientador desta dissertação por ter demostrado sempre uma enorme abertura e disponibilidade.

Gostaria também de agradecer à professora Regina Bispo, que foi um elemento essencial para que a realização deste trabalho fosse possível.

Quero agradecer à minha família, pelo apoio dado, não só durante a realização desta dissertação, mas também durante o meu percurso académico.

E a todos que apesar de não serem mencionados contribuíram de forma direta ou indireta para a realização desta dissertação.

Resumo

Hoje, como sempre, as empresas procuram estratégias para otimizar os seus recursos, em particular na área da logística existem muitas oportunidades para colocar em prática estas ideias. O foco deste trabalho incide na área da distribuição, mais concretamente na criação de rotas.

Este desafio foi proposto pela Galp e enquadra-se no projeto de otimização da distribuição secundária de combustíveis. Este trabalho consiste no desenvolvimento de um algoritmo eficiente para determinar as rotas diárias de distribuição de combustíveis de acordo com a necessidade dos clientes.

A Galp sentiu a necessidade de alterar o seu sistema de criação de rotas uma vez que a partir da publicação do Decreto-Lei n.º 132/2017 os veículos pesados passam a poder transportar mercadorias até 44 toneladas em vez de 40, neste caso, significa que podem ser utilizadas cisternas de 36 m^3 em vez de 33 m^3. Outra alteração recente é relativa à localização dos clientes, uma vez que foi realizado um levantamento das coordenadas geográficas de cada um, o que significa que se passou a ter acesso à localização exata dos mesmos, isso permite determinar com maior exatidão as distâncias de transporte.

O objetivo deste trabalho é criar um conjunto de rotas que minimize a distância total percorrida e que respeite as restrições do problema, sendo elas a capacidade dos veículos, a janela temporal dos clientes, o horário dos motoristas e as condições de cada tipo de cliente. Para a determinação das rotas será utilizada uma metaheurística, a pesquisa tabu, uma vez que é um método que produz boas soluções e é computacionalmente eficiente. Para averiguar a qualidade das rotas obtidas pelo algoritmo estas serão comparadas com as rotas realizadas pela Galp no período em estudo.

Palavras-chave: Criação de rotas, Heurísticas, Pesquisa tabu

Abstract

Today, as always, companies are looking for strategies to optimise their resources, particularly in the area of logistics there are many opportunities to put these ideas into practice. The focus of this work is in the area of distribution, more specifically, vehicle routing problem.

This challenge was proposed by Galp and is part of the secondary fuel distribution optimization project. This work consists of the development of an efficient algorithm to determine the daily fuel distribution routes according to the customers' needs.

Galp felt the need to change its route creation system from the publication of Decree-Law no. 132/2017 heavy vehicles can now transport goods up to 44 tonnes instead of 40, in this case, it means that 36 m^3 tanks can be used instead of 33 m^3. Another recent change is regarding to the location of the customers, as a survey was carried out of the geographic coordinates of each one, which means that we now have access to their exact location, this allows us to determine more accurately the transport distances.

The objective of this work is to create a set of routes that minimizes the total distance and that respects the constraints of the problem, which are the capacity of the vehicles, the time window of the customers, the drivers' schedule and the conditions of each type of customer. The determination of the routes will be done with a metaheuristic, the tabu search, since it is a method that produces good solutions and is computationally efficient. To ascertain the quality of the routes obtained by the algorithm, these will be compared with the routes taken by Galp during the study period.

Keywords: Vehicle routing problem, Heuristics, Tabu search

Índice

Índice de Figuras

Índice de Tabelas

Siglas

CVRP	*Capacitated Vehicle Routing Problem*
CVRPTWCP	*Capacitated Vehicle Routing Problem With Time Windows and Client Priority*
MDVRP	*Multi-Depot Vehicle Routing Problem*
PRVP	*Periodic Vehicle Routing Problem*
VNS	*Variable Neighborhood Search*
VRP	*Vehicle routing problem*
VRPHTW	*Vehicle Routing Problem with Hard Time Windows*
VRPP	*Vehicle Routing Problem with Profits*
VRPPD	*Vehicle Routing Problem with Pickup and Delivery*
VRPSTW	*Vehicle Routing Problem with Soft Time Windows*
VRPTW	*Vehicle Routing Problem with Time Windows*

1

Introdução

1.1 Motivação e enquadramento

O sector dos transportes é essencial na nossa sociedade e tem um grande impacto na economia. Segundo os dados da comissão europeia [8] o transporte de mercadoria tem aumentado nos últimos anos sendo o transporte terrestre o mais utilizado, por exemplo, em 2019 cerca de 52% dos serviços realizados utilizaram este tipo de transporte. Mesmo sendo um dos meios de transporte mais utilizados, envolve ainda grandes custos operacionais para as empresas. Portanto a otimização da distribuição é relevante não só do ponto de vista económico, mas também do ponto de vista da satisfação dos clientes.

A otimização das rotas de distribuição é um dos problemas que tem vindo a ser estudado ao longo dos anos, uma vez que permite uma diminuição dos custos, seja no combustível, no equipamento ou na manutenção. Mesmo que sejam feitas pequenas melhorias diárias, podem originar uma grande poupança em termos anuais. Segundo Toth e Vigo em [42] o planeamento das rotas de distribuição produz poupanças substanciais nos custos de transporte, sendo geralmente entre 5% a 20%.

Com este trabalho pretende-se fazer a otimização das rotas de distribuição secundária de combustíveis rodoviários da Galp. A Galp é uma empresa portuguesa do sector energético que tem como principal mercado a península ibérica, mas faz a exportação dos seus produtos para mais de 50 países. As suas áreas de atividade são a exploração, produção, refinação e distribuição de produtos petrolíferos e a comercialização de gás natural e de eletricidade. Em 2020 a Galp contava com 6114 colaboradores distribuídos por 11 países e a sua capitalização bolsista foi de 12,6 biliões de euros. Relativamente à sua atividade em 2020, teve uma produção média diária de 130 mil barris, vendeu cerca de 6 mt de produtos petrolíferos, 22.9 TWh de gás natural e 3.3 TWh de eletricidade a clientes diretos [33].

Pretende-se com este trabalho realizar a implementação de um algoritmo que construa as rotas de distribuição entre os centros de carga (local de armazenamento dos combustíveis) e os seus clientes, sejam eles postos de abastecimento (as estações de venda ao público) ou clientes de venda direta (indústrias, hospitais, entidades governamentais

como forças de segurança, entre outros).

A Galp sentiu a necessidade de analisar o seu sistema atual de distribuição tanto por alterações realizadas a nível interno, como pela alteração da lei em relação ao limite de carga dos veículos pesados. Na primeira, foi feito o levantamento das coordenadas geográficas de cada cliente, o que permitirá a realização de um cálculo de distâncias mais preciso. Anteriormente, as distâncias resultavam de um cálculo aproximado entre o cliente e o centro de carga.

O outro fator que influencia a distribuição e que sofreu recentemente alterações, está relacionado com a alteração do Decreto-Lei n.º 132/2017 no dia 12 de outubro de 2017, que regulamenta os pesos e as dimensões máximas permitidas em Portugal para transportes de mercadorias. Em termos práticos esta lei permitiu o aumento dos pesos brutos máximos de 40 para 44 toneladas. Isto permite que as cisternas usadas para o transporte sejam de 36 m^3 em vez de $33m^3$.

Neste momento a Galp ainda se encontra num processo de transição em relação à atualização dos seus veículos, uma vez que se trata de um processo gradual, pelo que na distribuição ainda se podem-se encontrar alguns veículos que suportam as capacidades anteriores.

1.2 A distribuição de combustível na Galp

A distribuição de combustíveis na Galp está dividida em seis áreas de distribuição regional, estando estas alocadas em regime de exclusividade a operadores logísticos (empresas externas à Galp que fazem por contrato a distribuição) e são servidas a partir de centros de carga pré-definidos. O transporte dos combustíveis é realizado por estas empresas contratadas pela Galp que têm os seus próprios modelos para as rotas de distribuição. Essas empresas apresentam a sua proposta de distribuição à Galp, que as aceita ou faz uma contraproposta. Na situação atual a Galp não consegue avaliar se o modelo que lhe está a ser proposto é o melhor em termos de custo. Se a Galp dispuser do seu próprio modelo, poderá analisar as propostas que recebe em comparação com um modelo que se encontra otimizado e dialogar com os operadores a partir de uma base não meramente empírica.

Nas próximas secções será apresentado em detalhe o funcionamento da distribuição, apresentando separadamente os diversos intervenientes desta problemática.

1.2.1 Clientes

Na distribuição da Galp são considerados dois tipos de clientes:

- Postos de abastecimento com gestão de stocks;

- Clientes de venda direta *on-demand*. Estes podem ser revendedores, indústrias, empresas de transporte, hospitais, entidades governamentais como forças de segurança, entre outros.

Os primeiros representam 70% das encomendas, enquanto os restantes apenas 30% do volume dos combustíveis rodoviários distribuídos/entregues pela Galp.

De notar que os clientes de venda direta são considerados prioritários, uma vez que têm de ser abastecidos no dia a seguir à realização da encomenda. Além disso, a cada cliente está associado uma janela temporal, que corresponde ao horário que está disponível para a receção do combustível.

1.2.2 Veículos

Para a distribuição dos combustíveis são utilizados veículos pesados com uma cisterna, que dispõe de compartimentos segregados para permitir o transporte de mais de um tipo de combustível, podendo ir até 6 produtos distintos.

A frota utilizada para a distribuição de combustíveis na Galp é heterogénea, sendo que as principais diferenças são entre dois tipos de veículos:

- Veículos com cisternas com capacidade máxima de 36 m^3 (Figura 1.1(a));
- Veículos com cisternas com capacidade máxima de 15 m^3 ou 20 m^3 (Figura 1.1(b)), que serão designados como veículos rígidos.

(a) Veículo com semirreboque (b) Veículo Rígido

Figura 1.1: Ilustração dos tipos de veículos

Estes últimos são veículos mais pequenos e são utilizados nos clientes em que o local de descarga não permite o acesso a veículos de maiores dimensões (centros de cidades e centros históricos de vilas). Um veículo pode realizar mais do que uma rota por dia, uma vez que pode ser utilizado por 2 turnos de motoristas, cerca de 9 horas cada.

De notar, que por lei existe a limitação do peso bruto máximo. Os veículos a motor-semirreboque, com cinco ou mais eixos não podem exceder o peso máximo bruto de 44 toneladas e os pesos máximos por eixo também não podem ser ultrapassados. Neste

trabalho, de acordo com as recomendações estabelecidas pelo departamento *Refinery Optimization & Logistics* da Galp, será considerado que a carga de um veículo com 36 m^3 não pode exceder as 26 toneladas, já para os veículos rígidos será considerado que não podem exceder as 16 toneladas.

1.2.3 Rotas

É considerada uma rota, quando um veículo sai de um ponto de origem, depósito, serve um conjunto de clientes e volta novamente para o depósito. Um veículo pode fazer mais do que uma rota por dia desde que respeite as seguintes restrições:

- O ponto de origem da rota é o mesmo que o ponto de chegada;
- Cada cliente deverá ser visitado apenas uma vez por dia, por um único veículo, salvo casos excecionais;
- Uma rota não deve exceder as 9 horas diárias;
- A capacidade do veículo não deve ser excedida;
- O peso da carga não pode ser superior ao peso máximo permitido;
- A prioridade para os clientes de venda direta deve ser respeitada;
- Deve-se ter em conta que tipo de veículo o cliente pode receber para a entrega.

De notar que existem clientes em que a quantidade total de produto encomendado pode exceder a capacidade máxima dos veículos, por isso nesse caso esses clientes terão de ser servidos por mais do que um veículo no dia, para que a encomenda seja satisfeita. Para permitir essa opção, sem alterar as regras para os restantes veículos, e não considerar uma dimensão adicional no problema, optou-se por tratar esta situação com uma pequena alteração na estrutura do algoritmo que será referida mais adiante.

1.3 Objetivos

O principal objetivo deste trabalho é a criação de um algoritmo que construa as rotas diárias para a distribuição de combustíveis da Galp. A função objetivo do modelo será minimizar a distância total percorrida pelos veículos.

Para verificar a qualidade do modelo desenvolvido serão utilizados indicadores que permitem comparar os resultados obtidos com as rotas praticadas num período anterior pela Galp. Essa comparação será feita para um período de 6 meses que corresponde ao período dos dados fornecidos pela Galp.

Os principais indicadores a usar para a comparação são:

- Lote médio por rota: $\frac{\text{Capacidade } (L)}{\text{N° de Rotas}}$

- Rácio distância/rota: $\frac{\text{Distância } (Km)}{\text{N° de Rotas}}$
- Rácio distância/capacidade: $\frac{\text{Distância } (Km)}{\text{Capacidade } (L)}$
- Número médio de entregas por rota: $\frac{\text{N° de Clientes}}{\text{N° de Rotas}}$

1.4 Dados

A Galp tem em Portugal seis áreas de distribuição regional e existem diferenças significativas na quantidade de encomendas realizadas em cada uma dessas regiões. Por essa razão foi decidido que os dados a utilizar neste trabalho correspondem a uma área de escala média.

Num âmbito global a Galp tem para a distribuição combustíveis rodoviários 640 postos de abastecimento e 5000 clientes de venda direta e na área em estudo existem 120 postos de abastecimento e 950 clientes de venda direta, ambos multiprodutos, o que significa que um cliente pode encomendar mais do que um tipo de produto (Figura 1.2).

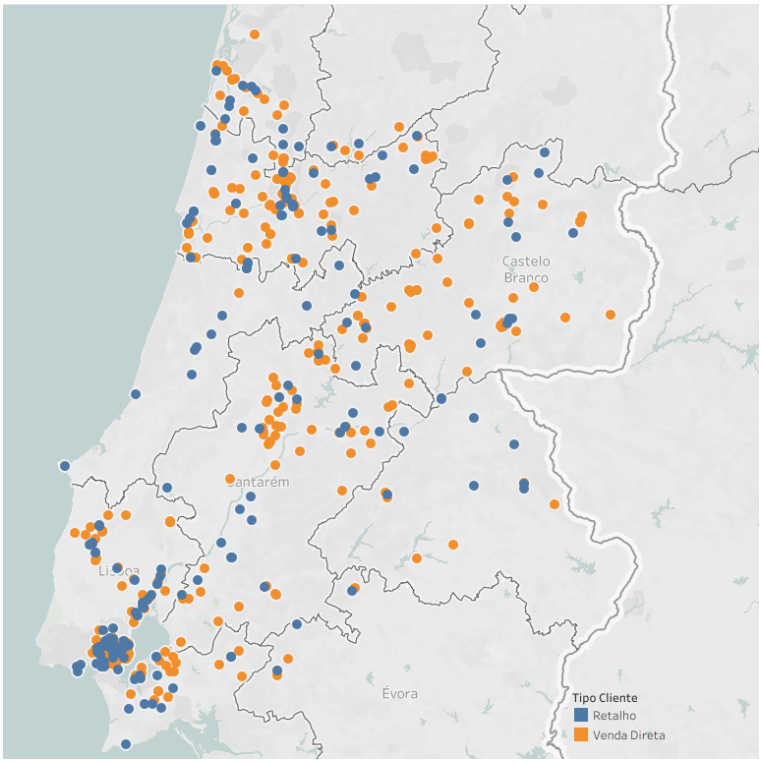


Figura 1.2: Clientes da Galp na área em estudo

A atividade dessa região pré-pandemia correspondia a 354,4 milhões litros/ano, cerca de 11 130 fretes/ano e o frete médio era 32 000 L, este valor ainda não traduz a otimização associada ao aumento da capacidade da carga a ter em conta neste trabalho.

Os dados disponibilizados pela Galp para este trabalho cobrem um período real de seis meses, com atividade não impactada por eventos disruptivos, que correspondem ao período de setembro de 2019 a fevereiro de 2020.

Esses dados contêm informação sobre as rotas definidas durante esse período, assim como o veículo que realizou o transporte, os clientes que foram servidos, o tipo e a quantidade de produto encomendado. Para além disso tem-se também a localização de cada cliente através de coordenadas e será com essa informação que se irá obter as distâncias e o tempo de viagem entre os clientes.

1.4.1 Tipos de produtos

Neste trabalho tem-se apenas em conta a distribuição de combustíveis que é realizada com cisternas. A Galp tem outros tipos de produtos, como gás e lubrificantes, mas estes não serão incluídos neste estudo. Nos dados utilizados são considerados 14 produtos distintos para distribuição, sendo eles vários tipos de gasóleo e gasolina, na tabela 1.1 encontram-se alguns exemplos desses produtos.

Produtos
Gasóleo de Aquecimento
Gasóleo Evologic
Gasóleo Hi Energy
Gasóleo Hi AgroPro
Gasóleo Mineral Corado
Gasóleo Simples
Gasolina Evologic 95
Gasolina Evologic 98
Gasolina Simples 95

Tabela 1.1: Tipos de produtos

1.4.2 Tipo de veículos

Como foi referido na secção 1.2.2 a distribuição que está a ser considerada neste estudo é realizada por dois tipos de veículos, os semirreboques e os veículos rígidos. Para testar a solução do algoritmo foi definido que serão considerados veículos com três dimensões diferentes, uma para os semirreboques e duas para os veículos rígidos, tendo estes segundos capacidades totais diferentes, 21 m^3 e 15 m^3. Na tabela 1.2 têm-se as capacidades dos compartimentos da cisterna de cada um destes veículos.

1.4.3 Parques de armazenamento

A Galp possui diversos parques logísticos para a realização da distribuição a nível nacional. É nesses parques que é realizada a carga dos combustíveis nos veículos para a entrega.

Tipologia	Capacidade Máxima	Comp. 1	Comp. 2	Comp. 3	Comp. 4	Comp. 5	Comp. 6
Semirreboque	36	11	5	4	3	7	6
Veículo Rígido 1	20.8	3	3.9	3	5.9	5	-
Veículo Rígido 2	14.9	5.9	4.3	4.7	-	-	-

Tabela 1.2: Capacidade dos compartimentos dos veículos em m^3

Na figura 1.3 estão representados os parques da Galp existentes em Portugal continental em que cada um deles serve uma área regional, no entanto existem produtos específicos que estão apenas disponíveis em alguns desses parques.

A área em estudo deste trabalho afeta a zona centro, portanto as rotas que são geradas pelo algoritmo desenvolvido irão ter todas o mesmo ponto de partida e de chegada, a CLC (Companhia Logística de Combustíveis) em Aveiras. Nos próximos capítulos este local será designado como depósito.

Figura 1.3: Parques de armazenamento de combustíveis da Galp

1.4.4 Encomendas dos clientes

As encomendas são realizadas pelos clientes e têm a informação da quantidade e do tipo de produtos que pretendem receber. Se estas forem realizadas até às 13:00 horas serão entregues no dia seguinte, no entanto pode haver situações extraordinárias em que essas encomendas serão apenas entregues dois dias após a realização do pedido.

De notar que existem clientes que são considerados prioritários, os clientes de venda direta, portanto o algoritmo considera que estes clientes têm que ser servidos obrigatoriamente no dia seguinte à realização da encomenda.

1.4.5 Distância entre clientes

Os dados que foram fornecidos pela Galp têm uma lista com todos os clientes que foram servidos no espaço temporal em estudo e foram dadas as coordenadas geográficas de cada cliente. Através dessas coordenadas e com o auxílio da ferramenta *Openrouteservice* criaram-se a matriz de distâncias e a matriz com os tempos de viagem. A matriz de distâncias tem a informação da distância em km entre cada um dos clientes, incluído o depósito central e a matriz de tempos de viagem contêm a duração da viagem em minutos. Na secção 4.2 serão apresentados mais detalhes sobre a construção destas matrizes.

1.4.6 Análise dos dados

Analisando as rotas realizadas entre setembro de 2019 a fevereiro de 2020, período em estudo, pode-se concluir que o número de clientes servidos por dia é muito variável. Sem considerar os fins de semana e feriados, o número mínimo de clientes servidos por dia foi 47 e o número máximo 101 (Gráfico 1.4). No dia em que foram servidos menos clientes realizaram-se 27 rotas nas quais foram utilizados 16 veículos, já no dia com maior número de clientes servidos foram usados 20 veículos que realizaram um total de 56 rotas. O número de clientes servidos por rota são no máximo 5, no entanto a maioria das rotas serve apenas um ou dois clientes (Gráfico 1.5).

1.5 Estrutura da dissertação

Esta dissertação tem a seguinte estrutura:

- **Capítulo 1 - Introdução**

  Neste primeiro capítulo foram apresentadas as motivações para a realização deste trabalho e descreveu-se em detalhe o problema e as suas características.

- **Capítulo 2 - Rotas de veículos: Tipos de problemas e algoritmos**

  No capítulo 2 é feita uma revisão bibliográfica sobre os problemas de rotas para veículos e as suas variantes. Serão também apresentados alguns dos algoritmos existentes para a resolução deste tipo de problemas.

- **Capítulo 3 - A pesquisa tabu**

  No capítulo seguinte serão descritos os principais conceitos da pesquisa tabu, uma vez que será o algoritmo utilizado para a resolução deste problema.

- **Capítulo 4 - Modelo proposto**

  Neste capítulo será feita uma descrição detalhada da implementação do algoritmo para a resolução específica do problema em estudo.

- **Capítulo 5 - Um estudo sobre a dupla vizinhança na pesquisa tabu**

  No capítulo 5 será apresentada uma nova abordagem para a pesquisa tabu, onde serão utilizadas duas vizinhanças independentes, em que cada uma delas faz a pesquisa usando direções opostas.

- **Capítulo 6 - Resultados computacionais**

  No capítulo seguinte serão apresentados os resultados obtidos para o algoritmo implementado e é realizada uma comparação entre as rotas obtidas pelo algoritmo e as realizadas pela Galp.

- **Capítulo 7 - Conclusões**

  Por fim, são retiradas algumas conclusões sobre o trabalho realizado e também serão apresentadas algumas ideias que podem ser desenvolvidas num futuro trabalho.

Nota: Existem termos técnicos que apresentam alguma dificuldade em serem traduzidos para a língua portuguesa, portanto optou-se por os escrever na língua inglesa, que é como eles são conhecidos pela comunidade científica. As abreviaturas também serão mantidas na língua inglesa para manter uma uniformidade com a literatura.

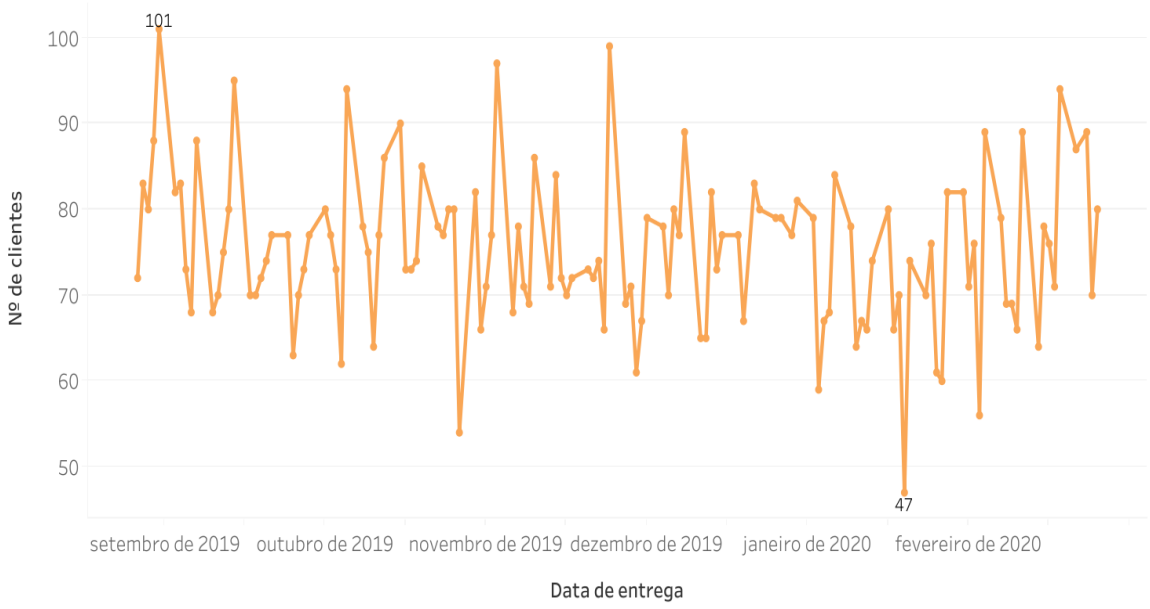


Figura 1.4: Número de clientes servidos por dia entre setembro de 2019 e fevereiro de 2020

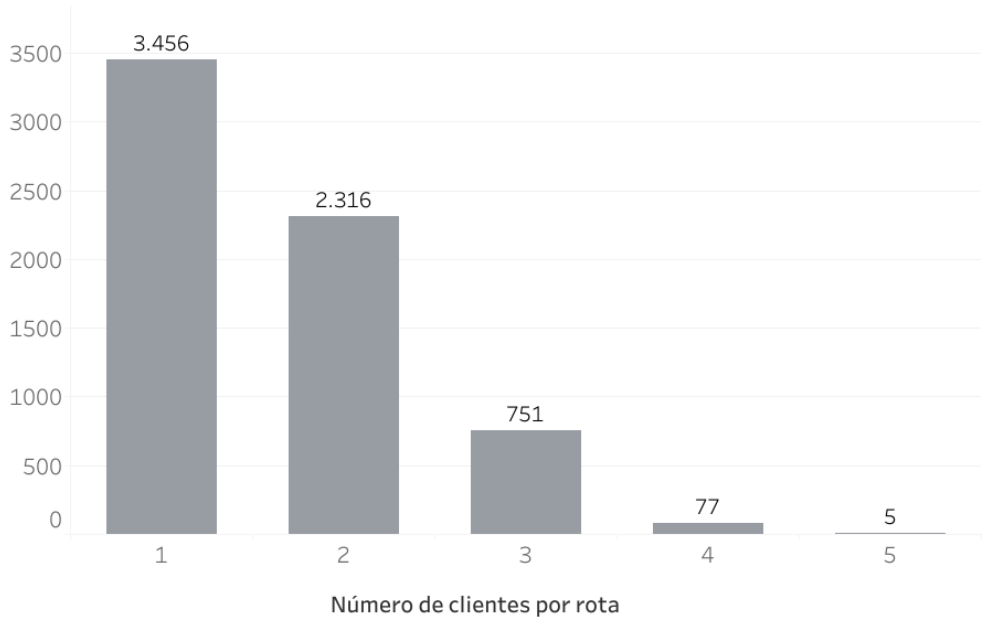


Figura 1.5: Número de clientes servidos por rota

2

Rotas de veículos: Tipos de problemas e algoritmos

2.1 Enquadramento

A otimização de rotas de veículos (*Vehicle routing problem*, VRP) é um problema clássico da investigação operacional, mais exatamente da otimização. Um problema de otimização consiste na minimização ou maximização de uma função objetivo sujeita a um conjunto de restrições:

$$\begin{aligned} &Min \text{ ou } Max\ f(x) \\ &\quad s.a.\ g_i \leq 0, \text{ para } i \in i,...,n \\ &\quad\quad h_j = 0, \text{ para } j \in i,...,m \\ &\quad\quad x \in X \end{aligned}$$

Onde, n e m são inteiros positivos, X um subconjunto em R^n e f, g_i e h_j funções reais em X.

Em geral, para um VRP pretende-se a construção de um conjunto de rotas ótimas que minimizem os custos obedecendo às regras práticas da distribuição. Este é um dos problemas mais estudados na área de otimização combinatória, uma vez que pode ser aplicado a diversos casos reais, como a recolha de lixo, a entrega de correspondência, a distribuição de bens, entre outros [21].

O problema de rotas para veículos foi introduzido em 1959 por Dantzig e Ramser [7] e em 1964 Clarke e Wright [2] propuseram o primeiro algoritmo. A partir daí, devido à aplicabilidade deste tipo de problemas na área da gestão de distribuição foram surgindo novas propostas de algoritmos com resolução exata ou aproximada para diferentes variantes do VRP.

Para as empresas também é importante o estudo do VRP, porque através dele pode-se fazer uma boa gestão da frota de veículos, ou seja, minimiza-se os custos e o tempo de viagem. Com isto também se pode fazer um bom agendamento das rotas e garantir a satisfação dos clientes.

2.2 Problema geral

A versão clássica do VRP considera que as entregas e a procura de cada cliente são determinísticas, que os veículos têm todos a mesma capacidade e que as rotas começam e acabam num depósito central. O objetivo deste tipo de problemas é minimizar os custos operacionais, o que implica reduzir o número de veículos necessários ou a distância total percorrida ou o tempo de viagem.

A estrutura subjacente a um VRP pode ser descrita por um grafo. Um grafo $G = (V, A)$ é descrito por um conjunto de vértices $V = \{0, ..., n\}$ e um conjunto $A = \{(i, j) : i, j \in V, i \neq j\}$ de arcos. Para um VRP os vértices $i = 1, ..., n$ representam os clientes e o vértice 0 corresponde ao depósito. Os arcos (i, j) representam a possibilidade de ir de um local representado por um vértice i para um outro local representado por um vértice j. A cada arco está associado um custo não negativo (c_{ij}) e este pode representar o custo da viagem gasto do vértice i para o vértice j e/ou a duração da viagem de um cliente para o outro, t_{ij} ($i = 1, ..., n, j = 1, ..., n$). Ainda se pode ter associado a cada cliente o tempo de serviço, s_i com $i = 1, ..., n$.

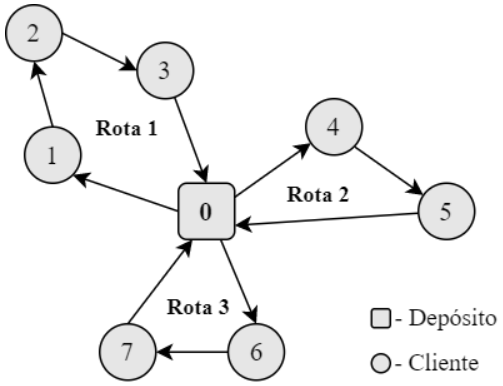


Figura 2.1: Representação do VRP

O VRP clássico consiste em encontrar um conjunto de rotas com custo mínimo que verifiquem as seguintes restrições:

- As rotas têm de começar e acabar no depósito;
- Cada cliente é visitado apenas por um veículo e recebe toda a encomenda contratada.

A formulação matemática para este problema é a seguinte [19]:

$$Minimizar \sum_{i \neq j} c_{ij} x_{ij}$$

Sujeito a:

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V \setminus \{0\} \tag{2.1}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\} \tag{2.2}$$

Restrições para eliminação de subrotas (2.3)

$$x_{ij} \in \{0,1\}, \forall \{i,j\} \in A; i \neq j \tag{2.4}$$

As restrições 2.1 e 2.2 garantem cada vértice entra e sai apenas um arco. A restrição de eliminação de subrotas 2.3 não foi especificada, uma vez que existem diversas formas de fazer cumprir esta condição. Na equação 2.4 é definida a variável binária x, que pode tomar os valores 0 ou 1, significando a utilização, ou não, do trajeto de i para j na rota.

2.3 Variantes do VRP

Como cada problema real tem as suas características específicas houve a necessidade de criar várias variantes do VRP, normalmente motivadas por restrições específicas não contempladas no VRP clássico. Nesta secção serão apresentados alguns exemplos dessas variantes.

2.3.1 *Capacitated Vehicle Routing Problem* (CVRP)

O CVRP é considerado uma das variantes mais simples e também a mais estudada. Para além das restrições apresentadas na secção 2.2, o CVRP tem uma restrição adicional que limita a capacidade total das rotas à capacidade do veículo utilizado.

Resumindo, o objetivo desta variante é minimizar os custos totais, sendo que estes podem ser o número de rotas, a distância total percorrida, o tempo de viagem, entre outros. As rotas começam e acabam no mesmo local, o depósito central. Cada cliente só pode ser visitado uma vez e recebe toda a encomenda contratada. Os veículos têm uma capacidade limitada, C, que não pode ser excedida, sendo que a restrição adicional é que a soma total da procura de cada rota, não pode exceder a capacidade do veículo utilizado para aquela rota.

2.3.2 *Vehicle Routing Problem with Time Windows* (VRPTW)

Esta variante é uma extensão do CVRP, para além da restrição da capacidade, cada cliente tem uma janela temporal nas quais as entregas devem ser realizadas.

A definição deste problema é como a apresentada em 2.2, com adição da restrição de capacidade e de que a cada cliente está associado uma janela temporal $[a_i, b_i]$ que representa o horário em que os clientes estão disponíveis para receberem a encomenda.

Na literatura podem-se encontrar duas formas de trabalhar com a janela temporal [1]:

- ***Vehicle Routing Problem with Hard Time Windows*** **(VRPHTW)** - Onde não são permitidos atrasos no serviço, ou seja, o veículo deve chegar antes do limite superior da janela temporal de cada cliente.

- ***Vehicle Routing Problem with Soft Time Windows*** **(VRPSTW)** - Nesta é permitido que os clientes sejam servidos fora da janela temporal, mas com um certo custo. Se o veículo chegar antes do limite inferior da janela temporal terá de esperar, já se chegar depois do limite superior será aplicada uma penalização ao atraso.

Em qualquer uma destas variantes é necessário também definir uma janela temporal para o depósito, esta é definida por $[a_0, b_0]$, que corresponde à primeira e última hora que os veículos podem sair e voltar de servir os clientes. Portanto nenhuma rota pode começar antes de a_0 e terminar depois de b_0.

2.3.3 *Vehicle Routing Problem with Pickup and Delivery* (VRPPD)

Esta variante surgiu para dar resposta a casos reais, em que ao longo da rota não é só necessário entregar produtos, como também existe a possibilidade de os recolher [42].

No VRPPD cada cliente, i, está associado a duas quantidades d_i, que corresponde à quantidade de produto a ser entregue e p_i a quantidade de produto a ser recolhida, sendo assim a procura do cliente é $q_i = d_i - p_i$. Este valor pode ser negativo, uma vez que podem haver clientes onde só é realizada a recolha. Considera-se ainda que a cada cliente está associado a um local onde é realizada a carga da encomenda O_i e outro onde é realizada a descarga do produto entregue D_i. Se O_i não corresponder ao depósito, então esse ponto deve ser incluído na rota antes de visitar o cliente i. O mesmo acontece com D_i, se este local for diferente do depósito, então tem que ser incluído na rota depois do cliente i ser visitado.

2.3.4 *Multi-Depot Vehicle Routing Problem* (MDVRP)

O MDVRP é uma generalização do VRP, uma vez que nesta variante existem múltiplos depósitos de onde os veículos podem começar e terminar as suas rotas. O objetivo desta variante continua a ser a minimização dos custos totais das rotas.

2.3.5 *Periodic Vehicle Routing Problem* (PVRP)

No PVRP as rotas são definidas num horizonte temporal de T dias. Cada cliente i para além da quantidade de produto encomendado tem associada a frequência, que indica o número de vezes que esse cliente deve ser visitado durante o horizonte temporal. A solução será um conjunto de rotas que satisfaçam as encomendas dos clientes como a frequência das visitas e o objetivo é a minimização dos custos totais no horizonte temporal definido.

2.3.6 *Vehicle Routing Problem with Profits* (VRPP)

Esta variante tem uma proposta diferente das que foram apresentadas anteriormente, porque o objetivo do VRPP é a maximização do lucro e além disso os clientes podem ser visitados mais do que uma vez e também não é obrigatório que sejam visitados todos os clientes.

2.4 Algoritmos para resolução do VRP

Os problemas de VRP pertencem à família de problemas combinatórios NP-difíceis, na prática isso significa que o esforço computacional aumenta exponencialmente com o aumento da dimensão do problema. Este tipo de problemas podem ser resolvidos por: métodos exatos, heurísticas ou metaheurísticas.

Com os métodos exatos procura-se obter a solução ótima do problema, no entanto estes tipos de procedimentos são computacionalmente caros e requerem muito tempo e memória computacional. Uma alternativa aos métodos exatos são as heurísticas ou as metaheurísticas, estas têm o objetivo de encontrar uma boa solução para o problema, podendo esta ser a solução ótima ou não. A vantagem em relação aos métodos exatos é que o custo computacional é muito menor. No entanto, desconhece-se o certificado de otimalidade para esta solução. Se se pretende correr o algoritmo com frequência, havendo pouca disponibilidade em termos de tempo computacional, esta é sem dúvida a melhor opção.

2.4.1 Métodos exatos

Como já foi referido, os métodos exatos procuram a obtenção de uma solução ótima. Como são computacionalmente exigentes nem sempre o método termina com a produção de uma solução ótima, por problemas de memória ou de tempo computacional, ainda assim muitas vezes, pode ser encontrada uma solução admissível e eventualmente uma indicação do desvio máximo em relação ao ótimo. Mais concretamente pode ser obtida uma solução admissível com a indicação de que no máximo poderá existir uma solução ótima cuja diferença em relação à função objetivo é de x%.

Nesta secção serão apresentados os dois principais algoritmos para a resolução destes problemas de forma exata.

2.4.1.1 *Branch and bound*

A ideia subjacente ao *branch and bound* é fazer uma pesquisa com uma estrutura de árvore binária, onde em cada nodo é resolvida uma relaxação do problema original, ao qual foram adicionadas restrições que permitem uma partição do conjunto de soluções admissíveis original. Por exemplo, se num nodo se obteve uma solução em que o valor de uma variável inteira, x_k, é um valor não inteiro x, são criados dois subproblemas,

adicionando às restrições do nodo em questão, respetivamente as restrições $x_k \leq \lfloor x \rfloor$ e $x_k \geq \lfloor x \rfloor + 1$ onde $\lfloor x \rfloor$ representa o maior número inteiro menor ou igual a x. O que o método faz é dividir o problema original em subproblemas menores tornando-se assim mais fáceis de resolver. O termo *branch* (ramificar) indica que é feita a divisão de um conjunto inteiro de soluções em subconjuntos cada vez menores. Enquanto *bound* (limitar) significa que é feita uma avaliação para verificar quão bom é o conjunto de solução, como dois limites um superior e um inferior.

À medida que o algoritmo avança, o problema é dividido em subproblemas, onde irão ser aplicados os limites. O limite superior num problema de minimização é definido pela melhor solução obtida até ao momento, designada por solução incumbente. Já o inferior é obtido através da resolução de uma relaxação do problema. Uma relaxação de um problema de minimização é outro problema de minimização com um conjunto se soluções admissíveis que inclui o original e uma função objetivo tal, que o valor de qualquer ponto admissível é menor ou igual ao valor do mesmo no problema original. Uma das relaxações mais comuns e mais utilizadas em otimização combinatória é a relaxação linear. Esta consiste num problema idêntico ao anterior, mas onde a condição de integralidade é removida ou substituída por limites inferiores e superiores para os valores das variáveis, assumidas como contínuas em vez de binárias ou inteiras. Em cada nodo, ao resolver a relaxação linear, obtemos um limite inferior para o valor ótimo de todos os problemas que venham a ser gerados a partir desse nodo. Assim se este limite inferior for superior ao valor da solução incumbente este nodo pode ser encerrado (*pruned*) e terminada a pesquisa que poderia decorrer a partir do conjunto de soluções admissíveis aí definido, dado que não será possível encontrar uma solução melhor que a incumbente. Um nodo também é encerrado se o subproblema aí representado for impossível, ou se a solução obtida for admissível para o problema original, procurando-se nesse caso atualizar, se possível, a solução incumbente. A estratégia de *pruning* deve ser muito eficaz de forma a garantir que a dimensão da árvore não seja explosiva.

2.4.1.2 *Branch and cut*

O método *branch and cut* é uma combinação do *branch and bound* com planos de corte. Um plano de corte é uma restrição funcional que diminui o conjunto de possíveis soluções sem eliminar nenhuma solução viável. Este algoritmo vai adicionando iterativamente planos de corte à relaxação linear definida no nodo considerado (Figura 2.2).

2.4.2 Heurísticas

Seguem-se alguns exemplos de heurísticas que podem ser aplicados ao problema em estudo. Algumas heurísticas são construtivas, no sentido que uma solução admissível vai sendo elaborada progressivamente. Já os outros métodos, começam com uma solução admissível (que pode ser obtida por uma heurística construtiva) e vão gerando outras

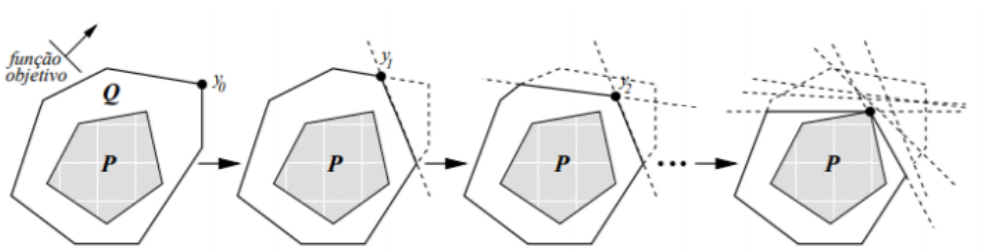


Figura 2.2: Representação gráfica da inserção de planos de corte [23]

soluções a partir desta. Nestes casos as heurísticas diferem sobretudo no modo como o espaço de soluções admissíveis é explorado.

2.4.2.1 Algoritmo *greedy*

O algoritmo *greedy* é um algoritmo simples que vai construindo a solução passo a passo, sendo que em cada passo vai-se escolhendo a melhor opção disponível. No caso dos problemas de rotas escolhe-se sempre o ponto mais próximo, isto se o objetivo for minimizar a distância.

Os passos para a implementação deste algoritmo são os seguintes:

1. Inicia-se uma rota parcial, apenas com o ponto inicial, neste caso será o depósito;
2. Depois é adicionada à rota o cliente em que o custo de transporte do cliente atual para esse cliente for menor. Sendo que todas as restrições devem ser respeitadas;
3. Quando já não for possível adicionar mais clientes, é adicionado à rota o ponto de retorno, o depósito.

Este é dos algoritmos mais populares devido à sua simplicidade. Por essa razão é que também é muito utilizado como solução inicial de alguns algoritmos mais complexos. No entanto uma das principais desvantagens apresentadas a este método é que ao escolher inicialmente sempre as melhores decisões, no final do algoritmo já não existem muitas opções disponíveis e a composição final da solução é muito penalizadora em termos da função objetivo, por essa razão é usado o termo *greedy*.

2.4.2.2 Heurística Clarke e Wright

Uma das heurísticas mais conhecidas para a resolução do VRP foi proposta por Clarke e Wright [2] em 1964, também conhecida como algoritmo *savings* uma vez que um método construtivo baseado no conceito de poupança (*savings*).

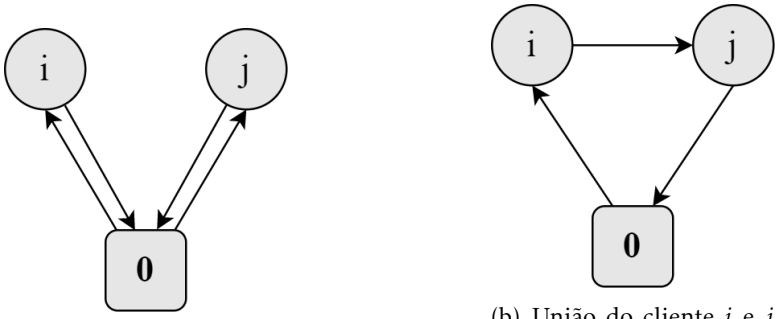
Inicialmente cada cliente é servido por uma rota que apenas o serve a ele, $(0, i, 0)$ e gradualmente estas rotas vão se unindo com a aplicação do critério de poupança. Isto é, as rotas $(0, ..., i, 0)$ e $(0, j, ..., 0)$ vão se unir e formar uma única rota $(0, ..., i, j, ..., 0)$. Com a

união destas duas rotas é gerada uma poupança (*saving*) de $s_{ij} = c_{i0} + c_{0j} - c_{ij}$, onde c_{ij} corresponde ao custo de transporte do cliente i para o cliente j.

Os passos a seguir para a implementação desta heurística são os seguintes passos:

1. Criam-se n rotas, onde n é o número de clientes e cada uma destas rotas só serve um único cliente;
2. Calcula-se o s_{ij} para todos os pares de clientes (i, j);
3. Ordenam-se os s_{ij} por ordem decrescente;
4. Unem-se as rotas dos clientes i e j em que se obtiverem o valor de s_{ij} mais elevado e que respeitem todas as restrições do problema;
5. Repete-se este processo até se esgotarem todas as possibilidades.

Esta é uma heurística simples e requer pouco tempo computacional, por essa razão é que é frequentemente usada como a solução inicial de outros algoritmos mais complexos. Este método geralmente produz resultados razoáveis, no entanto também se podem obter soluções de baixa qualidade. Segundo Cordeau *et al.* [5] a pior característica deste algoritmo é a falta de flexibilidade.



(a) Rotas iniciais

(b) União do cliente i e j em uma rota única

Figura 2.3: Poupança da heurística Clarke e Wrigth

2.4.2.3 Algoritmo *sweep*

A construção deste método é diferente do anterior, no caso desta heurística primeiro são determinados os subconjuntos de clientes, pertencentes à mesma rota e só depois disso é que é determinada a sequência de cada cliente na rota.

Para a implementação deste algoritmo assume-se que cada vértice i é representado por uma coordenada polar (θ_i, ρ_i), onde θ_i é o ângulo e ρ_i o raio entre o depósito e a localização geográfica do cliente i. Escolhe-se um vértice aleatoriamente, i^*, ao qual se atribui o valor θ_i^* e é a partir desse ponto que são calculados os ângulos para os outros vértices.

Esta é uma possível implementação para esta heurística:

1. Escolhe-se um veículo k;

2. Atribui-se ao veículo k um vértice sem rota com menor ângulo. Repete-se este processo até a capacidade do veículo ser atingida. Se ainda houver vértices sem rota, volta-se ao passo 1;

3. Após terem sido criadas todas as rotas estas são otimizadas através da resolução do problema do caixeiro-viajante.

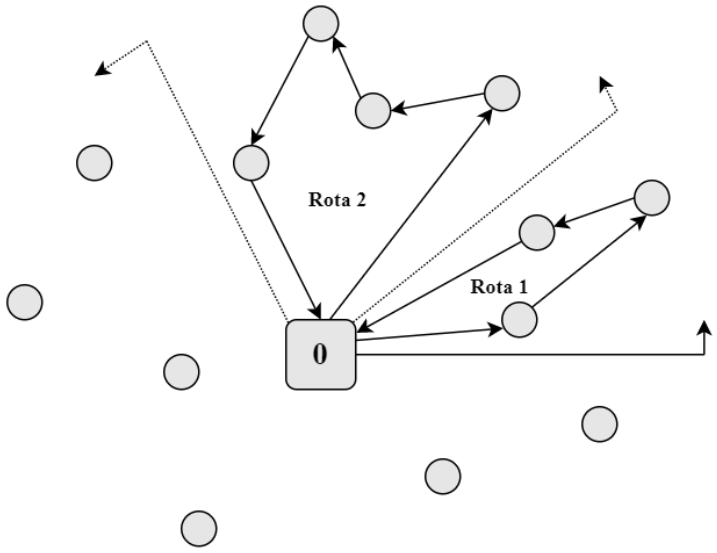


Figura 2.4: Representação do algoritmo *sweep*

2.4.2.4 Algoritmo *petal*

O algoritmo *petal* é uma extensão natural do algoritmo *sweep*, onde são criados conjuntos de rotas, designadas *petals* (pétalas) em que no fim é selecionada a melhor combinação através da resolução de um problema de partição [20].

Para a criação das rotas utilizam-se métodos construtivos, por exemplo o algoritmo *sweep*. Após a criação é realizada a seleção do conjunto de rotas, onde se resolve um problema de partição com a seguinte definição:

$$
\begin{aligned}
Min \sum_{k \in S} c_k x_k \\
s.a. \sum_{k \in S} a_{ik} x_k = 1 \ \forall \ i = i, ..., n \\
x_k \in 0, 1 \qquad \forall \ k \in S
\end{aligned}
$$

Onde c_k é o custo da rota k, a_{ik} é um parâmetro binário (0/1) que indica se o cliente i pertence à rota k e x_k:

$$x_k = \begin{cases} 1, & \text{se a rota } k \text{ é selecionada} \\ 0, & \text{caso contrário} \end{cases}$$

2.4.3 Metaheurísticas

O termo metaheurística foi introduzido por F. Glover em [12], são algoritmos de uso geral que podem ser aplicados a quase todos os problemas de otimização. Tal como as heurísticas estes métodos não garantem a solução ótima, mas os resultados geralmente apresentam boas soluções. Este tipo de algoritmos são bons para utilizar em problemas de grande dimensão, uma vez que costumam apresentar bons resultados num espaço de tempo razoável.

2.4.3.1 Pesquisa local

A pesquisa local é uma das metaheurísticas mais simples. O algoritmo começa numa solução inicial pré-determinada e a cada iteração a solução atual é substituída pela melhor solução da vizinhança, ou seja, a solução da vizinhança que melhora a função objetivo. A pesquisa termina quando na vizinhança as soluções forem todas piores que a solução atual.

Este método oferece boas soluções se o problema não tiver muitos ótimos locais ou se a qualidade dos ótimos locais for semelhante, isto porque o algoritmo é muito sensível à solução inicial. Na figura 2.5 está representado o comportamento do algoritmo para duas soluções iniciais diferentes, a partir desta representação consegue-se visualizar que o resultado algoritmo está muito dependente da solução inicial escolhida. Para evitar este tipo de problemas foram criadas outras metaheurísticas baseadas na pesquisa local, onde foram adicionadas estratégias para o algoritmo escapar de um ótimo local, um desses métodos é a pesquisa tabu.

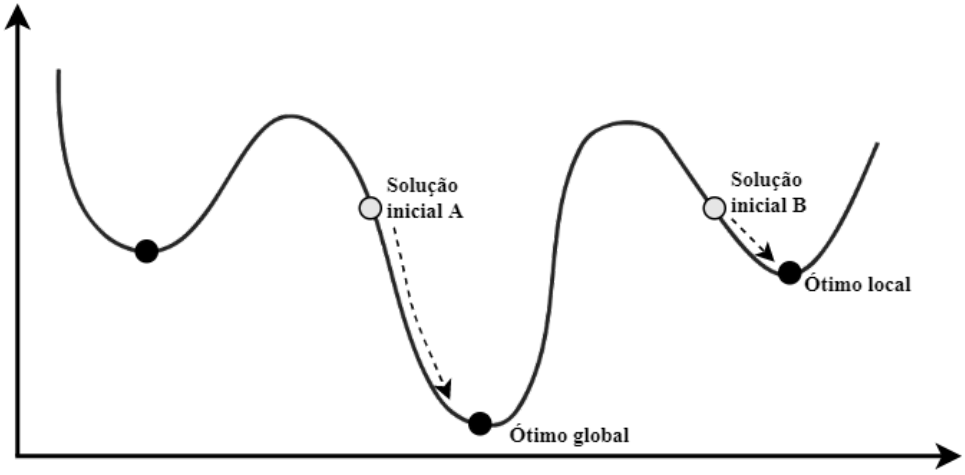


Figura 2.5: Representação da pesquisa local

2.4.3.2 Pesquisa de vizinhança variável

A pesquisa de vizinhança variável (*Variable Neighborhood Search*, VNS) é um método de pesquisa local que explora o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança N_k com $k = 1,\dots n$. O algoritmo explora várias estruturas de vizinhança para obter diversos ótimos locais e assim escapar dos mesmos.

O VNS foi proposto em 1997 por Mladenović e Hansen [24] e eles tiveram como base para a construção do algoritmo as seguintes observações:

- Um ótimo local de uma estrutura de vizinhança não é necessariamente o ótimo local de outra estrutura de vizinhança;
- Um ótimo global é também um ótimo local em qualquer uma das estruturas de vizinhança;
- Para muitos problemas, os ótimos locais estão relativamente perto entre si para uma ou mais estruturas de vizinhança.

Sendo esta última uma observação empírica, constatada por diversos autores.
O VNS básico apresentado pelos autores segue os seguintes passos:

1. Seleciona-se um conjunto de vizinhanças $N_k,\ k = 1,\dots,\ k_{max}$;
2. Determina-se a solução inicial x;
3. Define-se $k = 1$;
4. Repete-se os passos seguintes até $k = k_{max}$:
   a) O ponto x' é gerado aleatoriamente a partir da estrutura de vizinhança k;
   b) Aplica-se o método de pesquisa local, em que se tem x' como solução inicial. O ótimo local obtido através da pesquisa é definido como x'';
   c) Se a solução obtida, x'', é melhor que a solução incumbente (melhor solução obtida até ao momento), $x = x''$ e a pesquisa continua com a primeira estrutura de vizinhança ($k = 1$). Caso contrário, define-se $k = k + 1$, para a pesquisa continuar na próxima estrutura.
5. A pesquisa termina quando o critério de paragem for acionado.

Na figura 2.6 tem-se uma representação gráfica do algoritmo, onde é possível verificar que o algoritmo vai trocando de vizinhança até ser atingido o ótimo local e consequentemente o ótimo global.

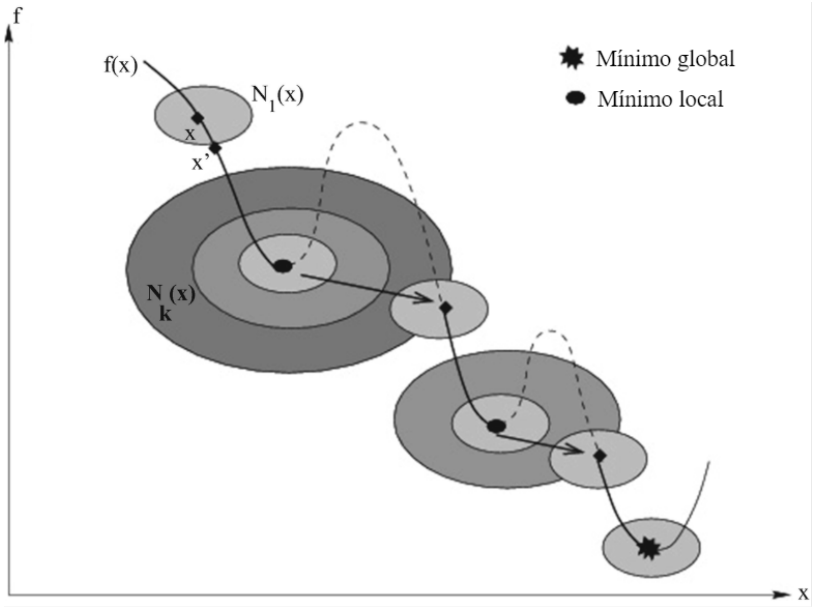


Figura 2.6: Representação do VNS [16]

2.4.3.3 Algoritmo genético

O algoritmo genético é baseado na Teoria da Seleção Natural, de forma resumida, defende que os organismos mais adaptados ao meio ambiente são selecionados, ou seja, sobrevivem, reproduzem-se e passam as suas características aos seus descendentes. Este algoritmo difere das outras metaheurísticas apresentadas até ao momento, uma vez que segue estas três ideias [44]:

1. Utiliza uma população de soluções para orientar a pesquisa;
2. Utiliza o *crossover* que recombina duas ou mais soluções para gerar novas soluções e potencialmente melhores;
3. Utiliza métodos de diversificação para sustentar a pesquisa.

Como já foi referido, os algoritmos genéticos utilizam populações de soluções codificadas como cromossomas, estes são normalmente uma série de *bits* por ser a forma mais fácil de lidar com a informação. Conforme a descrição do autores de [10] o algoritmo genético é constituído por uma população $X^t = \{x_1^t, \ldots, x_N^t\}$, em que f é a função de *fitness*, que corresponde à função objetivo do algoritmo. A cada iteração t, X^t é transformada em X^{t+1}, produzindo assim novas soluções (descendentes), substituindo assim o lugar dos progenitores. A ideia principal é passar as boas características da solução às novas gerações. Nos descendentes são aplicadas mutações aleatórias para assegurar a diversidade da população.

Segue-se uma breve descrição do algoritmo genético, onde k_1 é o número de gerações e $k_2 \leq N/2$ é o número de seleções por geração.

1. Seleção – Seleciona-se aleatoriamente a partir de X^t dois progenitores com probabilidade inversamente relacionada com o valor de f;

2. *Crossing-over* – Criam-se dois descendentes a partir de dois progenitores, ou seja, trocam-se um conjunto de *bits* de um conjunto selecionado aleatoriamente;

Progenitor 1	1	0	1	0	0
Progenitor 2	**0**	**0**	**1**	**1**	**1**
Descendente 1	1	0	**1**	**1**	**1**
Descendente 2	**0**	**0**	1	0	0

3. Mutação – Aplica-se uma mutação aleatória a cada descendente, onde se trocam os *bits* com uma probabilidade pequena;

4. Nova população – Obtém-se X^{t+1} a partir de X^t, removendo as $2k_2$ piores soluções em X^t que são substituídas por os $2k_2$ descendentes recém criados.

2.4.3.4 Otimização de colónias formigas

A ideia por detrás da otimização de colónias formigas é reproduzir o comportamento cooperativo das formigas para a resolução de problemas de otimização. A imitação do comportamento das formigas é interessante para aplicar num algoritmo de otimização, porque elas utilizam um mecanismo simples para o transporte de alimentos e esse mecanismo permite que façam o percurso pelo caminho mais curto.

O que acontece é que quando as formigas fazem as suas viagens para ir encontrar alimentos, elas libertam uma feromona que é deixada no chão. Esta feromona é uma substância olfativa e volátil. O objetivo desta substância é que as outras formigas possam ser guiadas em direção ao alimento. Quanto maior for a quantidade de feromona maior é a probabilidade de as formigas selecionarem esse percurso. Quanto mais curto for o percurso maior será o número de formigas que passam por esse caminho por unidade de tempo. Assim o caminho mais curto vai ficando cada vez mais carregado de feromona.

Na figura 2.7 está uma representação das ações das formigas, pode-se verificar que quando é inserido um obstáculo as formigas escolhem com igual probabilidade o caminho da esquerda ou da direita, mas como a substância vai evaporando ao longo do tempo elas acabam por escolher o caminho mais curto, porque será o que apresenta uma maior quantidade de feromonas.

Taillard, et al. [38] fizeram a seguinte analogia entre o algoritmo e o comportamento das formigas:

- A área de pesquisa das formigas e o conjunto de soluções viáveis para o problema;
- A quantidade de comida associada à fonte e a função objetivo;
- O rasto e a memória adaptativa.

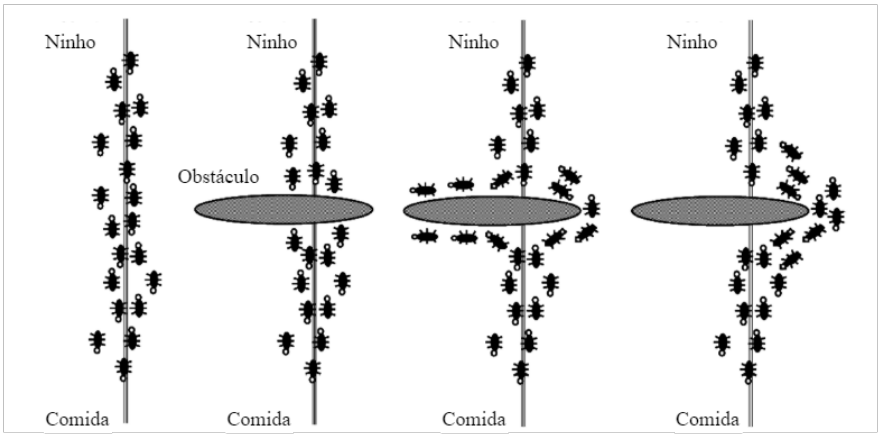


Figura 2.7: Inspiração de uma colónia de formigas que procura um caminho ótimo para o alimento [40]

Com isto pode-se esquematizar o algoritmo da seguinte forma:

1. Inicializa-se o rasto;
2. Enquanto o critério de paragem não for acionado:
    a) Para cada formiga é construído um novo caminho usando o rasto atual e uma avaliação da solução parcial a ser construída;
    b) É atualizado o caminho.

A componente mais importante é a gestão dos rastos. Nas abordagens *standard*, os rastos são utilizados em conjunto com a função objetivo para a construção de novas soluções. Os rastos são atualizados da seguinte forma:

1. Os rastos são todos enfraquecidos, para simular a evaporação das feromonas;
2. São reforçados os rastos que correspondem aos componentes que foram utilizados, isto tendo em consideração a qualidade da solução.

2.4.3.5 Pesquisa tabu

A pesquisa tabu foi proposta em 1986 por Fred Glover [12], mas em [13, 14] é que foram propostos muitos dos princípios que utilizamos atualmente. Esta metaheurística é uma extensão do método clássico de pesquisa local.

A pesquisa tabu explora o espaço de soluções movendo-se em cada iteração de uma solução para a melhor solução num subconjunto na sua vizinhança $N(S)$. Ao contrário da pesquisa local, o procedimento não termina no primeiro ótimo local, quando não é possível mais melhorias, a melhor solução da vizinhança é selecionada como a solução

atual, mesmo que seja pior do que a solução anterior a partir da qual a vizinhança foi definida. Esta abordagem permite que o método escape de um ótimo local e explore uma fração maior do espaço de pesquisa.

Concluindo, a pesquisa tabu começa a partir de uma solução inicial x_1 e move-se a cada iteração t de x_t para o melhor vizinho x_{t+1}, até que o critério de paragem seja satisfeito. Se $f(x)$ denota o custo de x, então $f(x_{t+1})$ não é necessariamente menor do que $f(x_t)$. Para evitar ciclos, soluções que foram recentemente examinadas são proibidas, ou seja, são consideradas tabu, por um determinado número de iterações. O conceito de tabu é essencial e deu o nome a este algoritmo. O tabu pode ser uma solução, um movimento que deu origem a partir da solução central da vizinhança à nova solução selecionada, uma regra de construção de uma solução, parte de uma solução, entre outros [17].

A pesquisa tabu é uma das heurísticas mais eficientes no sentido de encontrar soluções de qualidade em relativamente pouco tempo de execução [5, 26]. Por essa razão foi a abordagem escolhida para este trabalho. No próximo capítulo esta metaheurística será apresentada com mais detalhes.

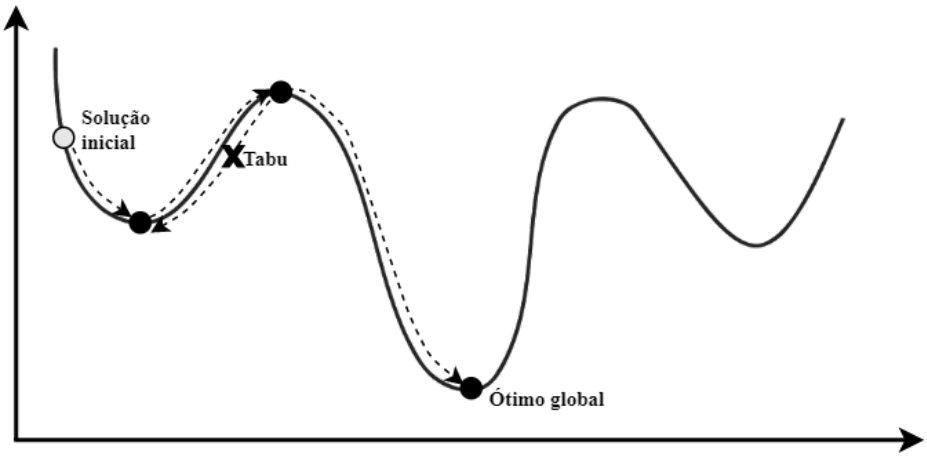


Figura 2.8: Representação da pesquisa tabu

3

A pesquisa tabu

O algoritmo da pesquisa tabu tem diversos elementos que precisam de ser definidos antes da implementação, sendo eles: a função objetivo, a estrutura da vizinhança, a lista tabu, o *tabu tenure*, o critério de aspiração e o critério de paragem. Como existem diversas estratégias que podem ser aplicadas a cada um destes conceitos, ao longo deste capítulo serão apresentadas as diferentes estratégias propostas na literatura.

3.1 O algoritmo básico

A pesquisa tabu é uma metaheurística que explora o espaço de soluções movendo-se a cada iteração para a melhor solução da vizinhança, não aceitando movimentos que originem as soluções já visitadas. No algoritmo 1 são apresentados os principais passos da pesquisa tabu [31].

Algoritmo 1: Pesquisa tabu

$T \leftarrow [\,]$;
$S \leftarrow$ Solução inicial;
$S^* \leftarrow S$;
repita
  Encontrar a melhor solução admissível $S' \in N(S)$, não tabu;
  se $f(S') > f(S^*)$ **então**
    $S^* = S'$;
  S = S';
  Atualiza a lista tabu T;
até *Critério de paragem*;
Output: Solução incumbente S^*

3.2 Solução inicial

Para a inicialização da pesquisa tabu é necessário definir uma solução inicial, a partir da qual é definida a primeira vizinhança de pesquisa. Para determinar essa solução podem

ser utilizadas diferentes estratégias, por exemplo, a solução pode ser gerada de forma aleatória ou determinada através de uma heurística. No segundo caso, uma das heurísticas mais utilizadas pela sua simplicidade são as baseadas em métodos *greedy* (2.4.2.1).

Gerar uma solução inicial aleatória torna-se mais rápido, no entanto o algoritmo pode demorar mais tempo a convergir. Já a utilização de uma heurística geralmente conduz o algoritmo a um ótimo local com um menor número de iterações. No entanto isso não significa que em todos os casos a utilização de melhores soluções iniciais levará o algoritmo a convergir para melhores ótimos locais [40].

3.3 Espaço de pesquisa

O espaço de pesquisa é o espaço que contém todas as soluções admissíveis para o problema. No caso do problema em estudo serão todas as rotas que satisfazem todas as restrições definidas na secção 1.2.3.

3.4 Estrutura da vizinhança

Em cada iteração é considerada uma solução atual, denotada por S e define-se o conjunto de soluções da vizinhança por $N(S)$. Formalmente $N(S)$ é um subconjunto do espaço de pesquisa que contém todas as soluções obtidas por uma transformação de S. Para cada problema existe mais do que uma estrutura de vizinhança possível que pode ser aplicada. Da secção 3.4.1 até à 3.4.3 serão apresentadas algumas opções de vizinhança que podem ser aplicadas em problemas de VRP.

Existem também várias estratégias para selecionar a solução na vizinhança, sendo algumas delas [40, 29]:

- Selecionar a melhor solução da vizinhança, ou seja, são examinadas todas as soluções da vizinhança e é escolhida a solução que apresenta um melhor valor para a função objetivo. Para grandes vizinhanças este tipo de exploração acaba por ser bastante morosa, o que implica um grande esforço computacional;

- Selecionar a primeira solução admissível, esta estratégia consiste em escolher a primeira solução da vizinhança que satisfaça todas as restrições do problema e não seja tabu, assim faz-se apenas uma exploração parcial da vizinhança;

- Outra estratégia que pode ser aplicada consiste em selecionar uma solução da vizinhança de forma aleatória.

A avaliação parcial da vizinhança, pode implicar uma deterioração da qualidade da solução, uma vez que não são analisadas todas as soluções em cada iteração. No entanto, a um nível global esta limitação pode acabar por não ter uma influência negativa na qualidade das soluções produzidas, porque podem ser realizados movimentos que não

seriam escolhidos se fosse realizada a exploração total da vizinhança, conduzindo assim a soluções interessantes [39].

A construção de algoritmos de pesquisa tabu eficientes implica um bom equilíbrio entre o peso computacional de cada iteração (vizinhanças mais complexas com um maior número de soluções a explorar) e o número de iterações a realizar (numa tentativa de explorar melhor todos espaços de soluções). Como já foi referido, as vizinhanças mais complexas enriquecem a pesquisa, uma vez que são consideradas mais soluções a cada iteração, mas ao mesmo tempo existe um risco de esforço computacional que se pode tornar excessivo.

3.4.1 Estrutura de vizinhança simples

Esta vizinhança é criada por movimentos simples, onde a cada iteração é feita a deslocação de um único cliente da sua rota atual. O cliente selecionado pode ser inserido na mesma rota ou noutra. No exemplo da figura 3.1 fez-se o movimento do cliente 5, que será colocado na posição anterior ao cliente 2.

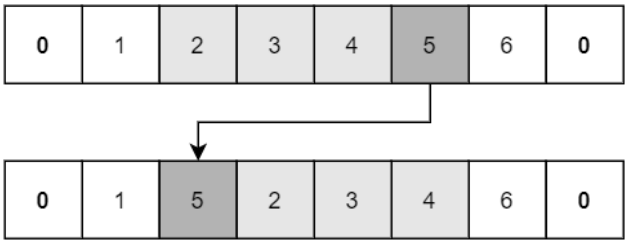


Figura 3.1: Exemplo estrutura de vizinhança simples

3.4.2 *CROSS exchange*

Em [37] foi proposta uma vizinhança designada *CROSS exchange* que consiste na troca de x clientes entre duas rotas. Na primeira rota é selecionado um subconjunto de n_1 clientes e na segunda rota também é definido outro subconjunto com n_2, os subconjuntos de cada uma das rotas não têm necessariamente a mesma dimensão. Depois de serem definidos esses subconjuntos, efetua-se a troca entre eles. Na figura 3.2 tem-se como exemplo a troca de 3 clientes em ambos os subconjuntos das rotas, onde o da primeira rota corresponde aos clientes 2, 3 e 4, enquanto os da segunda são os clientes 7, 8 e 9.

3.4.3 *Ejection chains*

A *ejection chains* [32] são sequências de movimentos coordenados entre os clientes de uma rota com outra e atua tipicamente em mais de duas rotas ao mesmo tempo. Este tipo de vizinhança consiste em identificar primeiro um conjunto de rotas e depois mover os vértices de forma cíclica da rota k_1 para a rota k_2, da rota k_2 para a rota k_3, etc. Como os

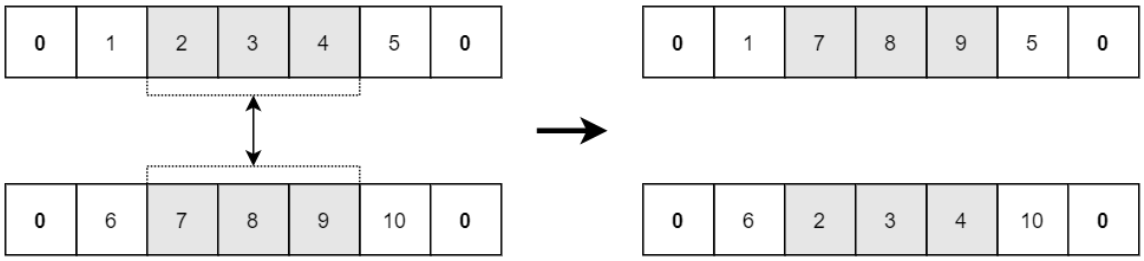


Figura 3.2: Exemplo *CROSS exchange*

$k_i's$ não são necessariamente rotas diferentes, o *ejection chains* permite tanto movimentos intra-rota como inter-rota.

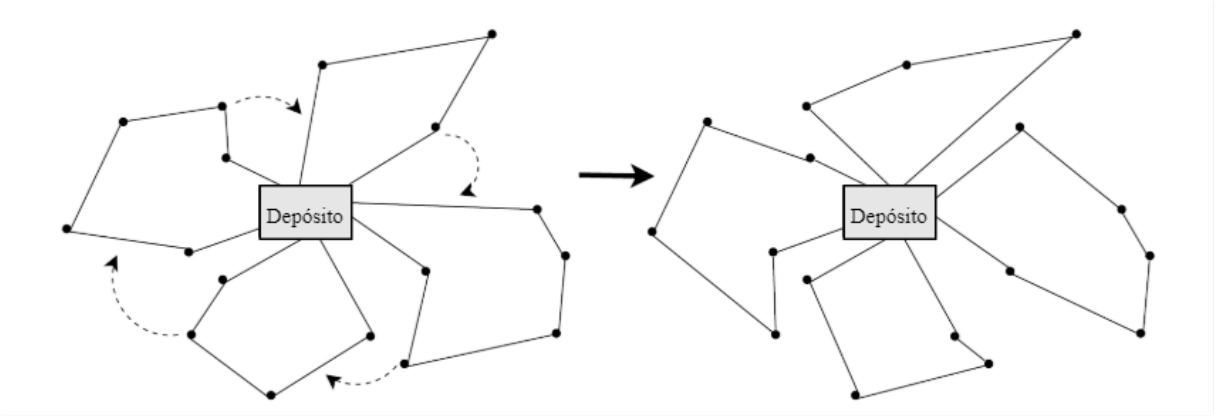


Figura 3.3: Exemplo *ejection chains* [40]

3.5 Lista tabu

A lista tabu é utilizada para evitar que o algoritmo entre num ciclo e também que progrida a partir de um ótimo local, conseguindo sair da esfera de atração desta solução. Realizando assim movimentos que podem vir a melhorar o valor da função objetivo global.

Normalmente nesta lista são guardadas as últimas transformações realizadas na solução e impede-se as transformações inversas, porque seria dispendioso computacionalmente armazenar a solução completa na lista tabu. O número de iterações consecutivas em que movimentos se mantêm tabu é designado por *tabu tenure*. Este valor é um parâmetro crucial no método e tem de ser afinado para cada instância do problema, uma vez que está relacionado com a duração dos ciclos que podem ocorrer no espaço de pesquisa. Na figura 3.4 tem-se o exemplo em que se fez a troca do cliente 2 com o 7, neste caso o movimento tabu será impedir que o cliente 7 volte a trocar com o cliente 2.

Sendo assim a lista tabu funciona como uma fila que pode ser, ou não, de tamanho fixo. Quando a lista tem tamanho fixo um novo movimento é adicionado e o mais antigo é removido. Assim, na exploração do subconjunto da vizinhança $N(S)$ da solução atual,

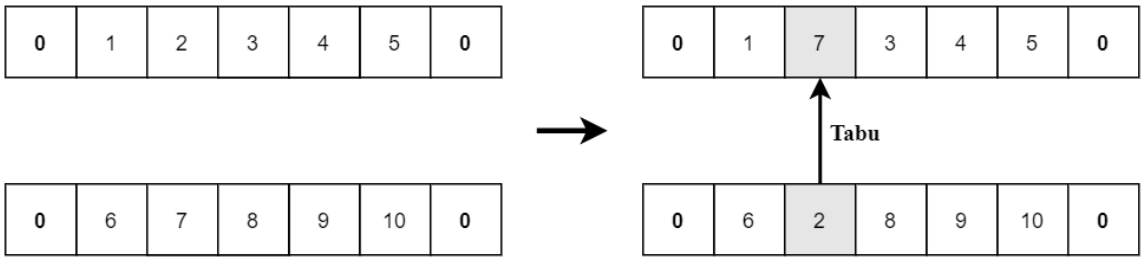


Figura 3.4: Movimento tabu

ficam excluídos da pesquisa os vizinhos que são obtidos pelos movimentos que constam na lista tabu.

3.5.1 *Tabu tenure*

O *tabu tenure* indica o número máximo de iterações que cada atributo deve permanecer na lista tabu. Se este valor for muito pequeno, a pesquisa pode revisitar as mesmas soluções, ou seja, entra num ciclo. Com o aumento do valor do *tabu tenure* aumenta a probabilidade de visitar novas soluções, no entanto se esse valor for demasiado elevado pode impedir uma pesquisa em profundidade numa dada região, ignorando eventuais boas soluções, uma vez que se impediu a realização de muitos movimentos. Portanto o tamanho da lista tabu não deve ser pequeno para evitar ciclos e também não deve ser grande demais para proibir muitos movimentos e ignorar partes significativas do espaço de pesquisa [39].

A definição do *tabu tenure* no algoritmo também pode variar, existem diversas estratégias que podem ser aplicadas, como por exemplo:

- Mantendo uma dimensão fixa e limitada (valor definido inicialmente);
- *Random tenure*, gerando números pseudoaleatórios para definir o número de iterações durante os quais o movimento permanece tabu [36, 9];
- Atribuindo diferentes valores de permanência na lista tabu dependendo do registo histórico [30, 25].

3.6 Critério de aspiração

O critério de aspiração é um mecanismo que retira, sob certas circunstâncias, o estado tabu de um movimento. Em [15] podem-se encontrar algumas alternativas, sendo as seguintes as principais:

- ***Default*** - Realiza o movimento tabu mais antigo se todos os movimentos possíveis são tabu.
- **Objetivo:**

  - **Global** - Aceita um movimento tabu se este melhorar a função objetivo global.
  - **Regional** - Aceita o movimento tabu se o valor da função objetivo for superior ao encontrado na região atual da pesquisa.

Existem outros critérios de aspiração mais complexos, como os propostos em [43] e em [17]. No entanto, um dos critérios mais utilizados, por ser o mais simples, é o critério de aspiração por objetivo global [11].

3.7 Critério de paragem

Na pesquisa tabu existem várias formas de terminar o processo de pesquisa, tais como:

- Após um determinado número fixo de iterações;
- Após um certo número de iterações sem nenhuma melhoria no valor da melhor solução;
- Quando a função objetivo atingir um valor pré-definido. Este critério evita a execução desnecessária do algoritmo quando uma solução é considerada suficientemente boa.

3.8 Estratégias utilizadas na pesquisa tabu

Para aperfeiçoar o processo de pesquisa são utilizadas estratégias de intensificação e diversificação.

3.8.1 Intensificação

A ideia por detrás do conceito de intensificação é replicar o que o ser humano provavelmente faria, que é explorar mais profundamente as secções onde foram encontrados os melhores resultados. Esta estratégia é utilizada para intensificar a procura em regiões que foram consideradas como promissoras, ou seja, o algoritmo volta a explorar a vizinhança das melhores soluções no histórico da pesquisa, de forma mais intensa.

Para a intensificação é necessário identificar as soluções de elite. O conjunto de elite é frequentemente definido em função dos melhores valores da função objetivo. Para a pesquisa mais fina pode-se reduzir o tamanho da lista tabu para um pequeno número de iterações.

A intensificação é utilizada em muitas implementações de pesquisa tabu, mas é sempre uma escolha do utilizador, não sendo sempre necessária para se atingir bons resultados [11].

3.8.2 Diversificação

A diversificação é o oposto, é um procedimento que alarga a pesquisa a regiões do espaço de soluções admissíveis que ainda não foram visitadas, gerando assim novas soluções. O processo de diversificação pode ser aplicado periodicamente ou após não serem registadas melhorias. Estas são três das estratégias que podem aplicadas [11]:

- ***Restart Diversification***: Esta técnica introduz na solução atual ou na solução incumbente pequenas alterações e recomeça a pesquisa nesse ponto. Por exemplo, no caso do VRP, os clientes que não são movidos com frequência podem ser forçados a serem introduzidos em novas rotas.
- ***Continuous diversification***: Neste caso a diversificação é realizada durante o processo normal de pesquisa. Esta pode ser feita introduzindo uma penalização na função objetivo aos movimentos ou soluções que são visitadas com regularidade.
- ***Strategic oscillation***: Esta estratégia permite que sejam consideradas soluções não admissíveis durante a pesquisa, sendo que lhes será aplicada uma penalização na função objetivo. O objetivo desta estratégia é passar por soluções não admissíveis, para depois voltar para uma solução admissível permitindo assim a exploração de outras regiões do espaço de pesquisa.

3.9 Estado da arte

Na literatura existem diversos trabalhos com a resolução de problemas de otimização através da pesquisa tabu, uma vez que se verificou ser um método bastante eficaz para vários problemas, chegando a soluções muito próximas da solução ótima [11]. Este método também se mostrou bastante eficaz para a resolução de problemas de VRP [5]. Seguem-se alguns dos trabalhos mais citados para problemas de VRP com a aplicação da pesquisa tabu.

Em 1993 Osman [29] apresentou uma das primeiras implementações de sucesso da pesquisa tabu para o VRP. Ele propôs a utilização do λ – *iterchange* como estrutura de vizinhança que consiste em movimentos simultâneos de clientes para diferentes rotas e na troca de clientes entre rotas. Ele ainda propôs duas versões para o algoritmo, a primeira consistindo na exploração de toda a vizinhança e escolhe-se o melhor movimento que respeite todas as restrições do problema. Já na segunda versão não se explora a vizinhança completa, porque é selecionado o primeiro movimento admissível.

Depois em 1994 surgiu um novo algoritmo designado *Taburoute* [9], uma das suas principais diferenças é que durante a pesquisa são consideradas soluções não admissíveis, por exemplo não respeitando as condições de capacidade dos veículos. Para isso são considerados dois parâmetros de penalização, um para a sobrecarga e o outro para a duração da rota e esses parâmetros vão sendo ajustados a cada 10 iterações. O objetivo desta estratégia é fazer a pesquisa entre soluções admissíveis e não admissíveis, para a

diversificar e assim diminuir a probabilidade da pesquisa ficar presa em ótimos locais. Em relação à lista tabu também é apresentada uma estratégia diferente: considerou-se que um vértice é movido da rota r para rota s na iteração t, a sua reinserção na rota r será proibida até à iteração $t + \theta$, onde θ é um número aleatório pertencente ao intervalo $[5, 10]$. Este algoritmo também utiliza uma estratégia de diversificação que penaliza os vértices que são movidos frequentemente.

O algoritmo de Taillard [35] tem algumas características do *Taburoute*, nomeadamente o *tabu tenure* ser aleatório e a utilização de técnicas de diversificação. Sendo que o diferencial deste algoritmo é a decomposição do problema em subproblemas, onde cada subproblema é resolvido de forma independente em processadores paralelos.

Rochat e Taillard [34] desenvolveram o conceito de memória adaptativa que consiste num conjunto de boas soluções, geradas por uma heurística, que são constantemente atualizadas através da adição de novos elementos de alta qualidade e com a extração dos elementos de menor qualidade, gerando assim novas soluções.

Em 2003, Toth and Vigo [41] propuseram o *Granular tabu search*, onde a ideia principal consiste em descartar as áreas não promissoras do espaço de pesquisa através da remoção do grafo os arcos mais longos, uma vez que a probabilidade de estas pertencerem à solução ótima ser reduzida. Para definir quais arcos que são excluídas os autores utilizaram um limite máximo que designaram por *granularity threshold*, $\nu = \beta\bar{c}$, onde β é o *sparsification parameter* e $\bar{c}$ é o custo médio dos arcos de uma solução gerada por uma heurística mais simples. O mecanismo de pesquisa adotado é a combinação deste conceito com do *Taburoute*.

O *Unified tabu search algorithm* foi inicialmente desenvolvido para o PVRP e MDVRP [3] e mais tarde foi ligeiramente modificado para o CVRP, VRPTW, PVRP e MDVRP [4, 5]. Este algoritmo tem algumas das características do *Taburoute*, permite a exploração de soluções não admissíveis ao qual é aplicado um parâmetro de penalização e a diversificação também é contínua. A diferença entre estes é o mecanismo do *tabu tenure*, que neste caso é fixo. O mecanismo tabu funciona com um conjunto $B(x)$ associado a uma solução x, ou seja, $B(x) = \{(i,k) : \text{vértice } v_i \text{ é visitado pelo veículo } k \text{ na solução } x\}$. A vizinhança é obtida removendo (i,k) de $B(x)$ e substitui-se por (i,k'), onde $k' \neq k$. O atributo (i,k) é declarado como tabu por um determinado número de iterações.

Duas vantagens deste algoritmo são a sua simplicidade, dado que não tem muitos parâmetros e a outra é a flexibilidade. Foi implementado com sucesso em diversas variantes do VRP e mantiveram-se os parâmetros com o mesmo valor [6].

4

Algoritmo tabu para o VRP - Galp

Este capítulo inicia com a apresentação do algoritmo de pesquisa tabu implementado para o caso da distribuição na Galp (secção 4.1). Na secção 4.2 será feita uma descrição de como foram adaptados os dados disponibilizados pela Galp de forma a poderem ser usados no algoritmo. Por fim, na secção 4.3 serão apresentadas as estratégias adotadas para cada uma das especificidades da pesquisa tabu descritas no capítulo anterior (3).

4.1 Modelo de distribuição na Galp

O modelo adotado permite a definição das rotas diárias de distribuição de combustíveis na Galp, onde será considerada como função objetivo a minimização da distância total percorrida pelos veículos. Considera-se uma rota o trajeto realizado por um veículo que partindo do ponto de origem, serve um conjunto de clientes e volta para o ponto partido. Neste trabalho considerar-se-á que todas as rotas começam e terminam no mesmo local, o qual será designado por depósito.

Como em qualquer problema real existem sempre restrições que devem ser impostas para que as rotas sejam admissíveis. As restrições definidas de acordo com a realidade da Galp são:

- A rota começa e termina no depósito;
- A rota não pode exceder as nove horas diárias;
- A capacidade do veículo não pode ser excedida;
- O peso máximo da carga não pode ser excedido;
- Os clientes prioritários devem ser todos servidos;
- Cada cliente deverá ser visitado apenas uma vez por dia, por um único veículo, salvo casos excecionais;
- O cliente deve ser servido dentro do horário definido para entrega.

- Deve-se ter em conta o tipo de veículo que pode servir um cliente por haver restrições de circulação dentro das cidades.

Os veículos podem realizar mais do que uma rota por dia e podem existir dois turnos de motoristas com 9 horas diárias cada.

Devido às características acimas descritas, este problema é considerado um VRP com a janela temporal de um dia, em que a capacidade do veículo tem que ser respeitada e em que pelo menos os clientes prioritários têm que ser servidos. Sendo assim este problema trata-se de um *Capacitated Vehicle Routing Problem With Time Windows and Client Priority* (CVRPTWCP).

4.2 Recolha de dados

Nesta secção serão apresentados os dados disponíveis e a forma como se obtiveram a partir destes, estimativas para valores necessários à implementação do algoritmo.

- **Encomendas** - Os dados disponibilizados pela Galp correspondem às rotas realizadas num certo período de tempo. A partir dos mesmos foi possível extrair informação relativamente ao número de cliente, à data de realização da entrega, à quantidade (litros), ao peso (kg) e ao tipo de produto distribuído.
- **Matriz de distâncias** - Para a implementação do algoritmo é necessária a distância entre todos os pares de clientes. Para determinar essas distâncias recorreu-se ao *software Openrouteservice*. O *Openrouteservice* [27] fornece vários serviços geográficos a nível global, sendo que os dados são recolhidos a partir do *OpenStreetMap* [28], que é um projeto gratuito e colaborativo. Um dos serviços disponibilizados é a determinação da matriz de distâncias entre pares de pontos geográficos, aqui representando como clientes. Para a determinação dessa matriz foi necessária a lista de coordenadas de cada cliente, a qual foi disponibilizada pela Galp. Como este *software* é personalizável, pode-se selecionar o tipo de veículo, neste caso veículos pesados, o que significa que as distâncias obtidas têm em consideração os caminhos viáveis para este tipo de transporte.
- **Matriz tempo de viagem** - Tendo em consideração o que foi deliberado em reuniões de trabalho com os responsáveis da Galp pelo planeamento de operações relacionadas com a distribuição, definiu-se que o cálculo da matriz tempo de viagem seria realizado com uma velocidade média de 65 km/h. Assim sendo esta matriz obteve-se através do seguinte cálculo:

$$t_{ij} = \frac{d_{ij} * 60}{65}$$

Onde t_{ij} é o tempo de viagem em minutos entre os clientes i e j e d_{ij} a distância em km entre os clientes i e j.

- **Tempo de serviço** - Para o tempo de serviço definiu-se que o tempo médio de carga seria de 40 minutos e que a operação de descarga teria uma duração de 50 minutos.

- **Informações relativas aos clientes** - Para a construção do algoritmo também foi preciso identificar se um cliente é prioritário e que tipo de veículo o cliente pode receber. Para identificar estas duas características utilizaram-se os campos que se encontravam nos dados relativos às encomendas.

4.3 Pesquisa tabu para a distribuição na Galp

4.3.1 Codificação da solução

Para a codificação da solução utilizou-se uma estrutura matricial, onde cada linha representa uma rota. Os elementos da matriz correspondem ao número do cliente. No exemplo apresentado tem-se representada uma matriz com 3 rotas de tamanhos diferentes, porque existem rotas onde são servidos menos clientes do que nas restantes. Este é o caso da rota 2, depois da rota ser definida os restantes elementos são representados com -1.

		Posição *j*				
		1	**2**	**3**	**4**	**5**
Rota *i*	**1**	0	1	2	3	0
	2	0	4	5	0	-1
	3	0	6	7	8	0

4.3.2 Solução inicial

Como foi visto no capítulo 3 a pesquisa tabu necessita de uma solução inicial como ponto de partida para o processo de pesquisa. Para o algoritmo implementado optou-se por utilizar uma heurística do tipo *greedy* uma vez que é simples de implementar.

O algoritmo *greedy* vai construindo as rotas passo a passo, escolhendo sempre o cliente mais próximo do atual. Para que seja criada uma solução admissível as restrições relativas à distribuição devem ser respeitadas, portanto a adaptação da heurística ao problema em estudo foi feita da seguinte forma:

- Como os clientes prioritários tem que ser obrigatoriamente servidos, estes serão os primeiros a serem inseridos nas rotas. Depois de já não haver mais clientes prioritários disponíveis é que se começa a inserir os restantes clientes;

- Quando é selecionado o primeiro cliente, verifica-se se esse cliente precisa de ser servido por um veículo rígido (1.2.2) ou não, dependendo desse resultado escolhe-se que tipo de veículo irá efetuar a rota. Se o cliente não precisar de ser servido por um veículo rígido o transporte será realizado num semirreboque, portanto todos os clientes dessa rota têm que permitir entregas com este tipo de veículo. Caso contrário, se o cliente precisar de ser servido por um veículo rígido, será esse o veículo selecionado e para os próximos clientes da rota não será necessário verificar que tipo de veículo deverá ser utilizado dado que qualquer cliente pode ser servido por um veículo rígido;
- Quando é adicionado um novo cliente é necessário verificar as restrições do problema:
  - Se o veículo tem espaço suficiente para inserir a encomenda do cliente;
  - Se o peso máximo do veículo não é excedido com a inserção do produto encomendado;
  - Se o tempo da rota não ultrapassa o tempo máximo definido;
  - Se o veículo chega ao cliente dentro do horário em que este está disponível para receber a encomenda;

Se nenhum dos clientes verificar as condições anteriores, então a rota termina, ou seja, o veículo volta para o depósito. O algoritmo continua a inserir novas rotas, até todos os clientes serem servidos ou até já não existirem mais veículos disponíveis para a realização das mesmas. Recordando que os clientes não prioritários podem ser abastecidos nos dias subsequentes.

Se houver clientes que não foram incluídos nas rotas é criado um vetor com essa informação, uma vez que quando se realiza a pesquisa tabu vão ser feitas trocas às rotas e estes clientes podem vir a ser inseridos em alguma delas.

O algoritmo 2 apresenta o resumo da implementação da heurística *greedy* adaptada ao problema em estudo.

4.3.3 Estrutura de vizinhança

A estrutura de vizinhança adotada neste trabalho foi a *CROSS exchange* (3.4.2) que permite fazer a troca de subconjuntos de clientes entre duas rotas e estes subconjuntos não precisam de ter o mesmo tamanho. Quando se faz a implementação desta estrutura define-se qual é o tamanho máximo para os subconjuntos. Na figura 4.1 tem-se um exemplo de um movimento entre duas rotas onde o tamanho dos subconjuntos são diferentes.

4.3.4 Lista tabu

Na lista tabu, como é computacionalmente dispendioso armazenar as soluções completas, serão guardadas as transformações que criam as novas soluções. Neste caso optou-se por

Algoritmo 2: Determinação da solução inicial

```
enquanto houver clientes ou veículos disponíveis faça
    Inicia uma nova rota;
    repita
        se ainda não foram inseridos todos os clientes prioritários então
            seleciona o cliente prioritário mais próximo do cliente atual;
        senão
            seleciona o cliente mais próximo do cliente atual;
        se é o primeiro cliente da rota então
            Determina o veículo a usar na rota;
        se capacidade dos produtos for menor ou igual à capacidade máxima do veículo
         então
            se peso total dos produtos for menor ou igual ao peso máximo do veículo
             então
                se tempo total da rota for menor ou igual ao tempo máximo permitido
                 por rota então
                    se horário de chegada ao cliente estiver de dentro da janela temporal
                     então
                        O cliente é inserido na rota;
    até não ser possível inserir mais clientes na rota;
```

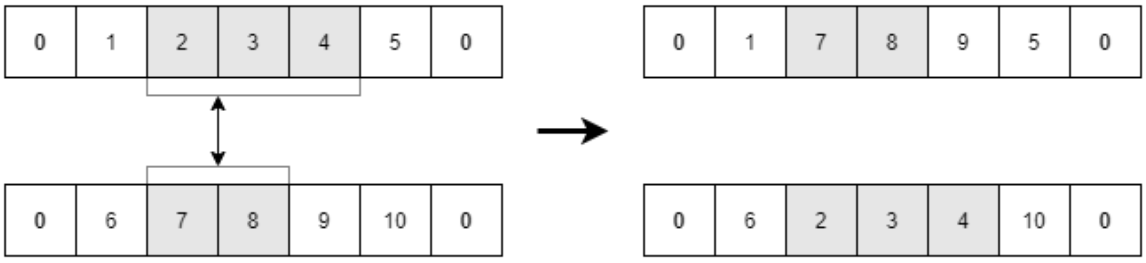


Figura 4.1: Exemplo de um movimento com o *CROSS exchange*

guardar o primeiro cliente de cada subconjunto movido e uma solução é considerada tabu se houver um movimento entre dois subconjuntos que contenham estes dois clientes. Por exemplo na figura 4.2 efetuou-se um movimento de dois subconjuntos com 3 clientes cada, enquanto estes clientes permanecerem na lista tabu não vão ser permitidos movimentos entre subconjuntos que contenham os clientes 2 e 7.

O tamanho da lista tabu, o *tabu tenure*, será um parâmetro fixo que é definido no início do algoritmo e para determinar o seu valor serão realizados testes para fazer a afinação do parâmetro.

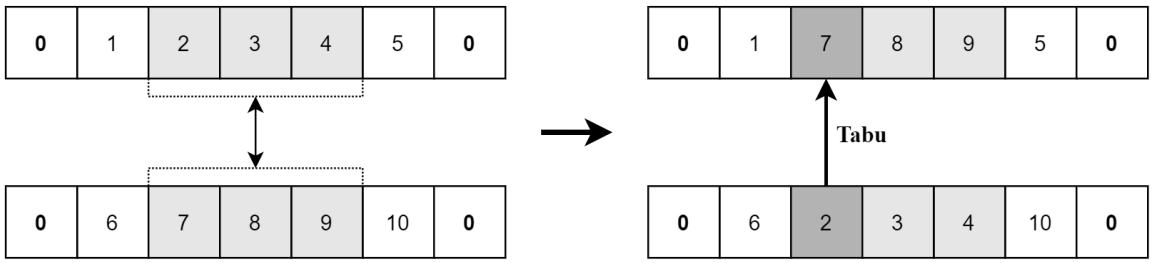


Figura 4.2: Exemplo de um movimento tabu

4.3.5 Critério de aspiração

Para o critério de aspiração decidiu-se utilizar o critério mais utilizado nos algoritmos de pesquisa tabu, o critério de aspiração por objetivo global. Esta estratégia retira o estado tabu a um movimento se este melhorar o valor da solução incumbente. Ou seja, o movimento é aceite se a distância total das rotas for menor que o melhor valor da função objetivo obtido até ao momento.

4.3.6 *Multistart*

A estratégia *multistart* é utilizada quando não existem soluções na vizinhança a ser pesquisada que não sejam tabu e consiste em reiniciar a pesquisa a partir de uma solução admissível ainda não utilizada. As soluções utilizadas no *multistart* são guardadas ao fim de um determinado número de iterações e estas são soluções admissíveis que não foram usadas na pesquisa em iterações anteriores. Esta estratégia permite que sejam exploradas novas soluções em áreas do espaço de solução ainda não visitadas, promovendo assim uma maior diversidade na pesquisa.

4.3.7 Critério de paragem

Este algoritmo tem dois critérios de paragem, o que significa se um deles for acionado o algoritmo termina. No primeiro a pesquisa termina quando forem executadas um determinado número de iterações. Já o segundo verifica qual foi a última iteração em que foi obtida a melhor solução e se ao fim de determinado número de iterações não tiverem sido obtidas melhorias o algoritmo termina. O número de iterações utilizadas para os critérios de paragem são definidas no início do algoritmo.

4.3.8 Restrições

Na pesquisa tabu realizam-se os movimentos que permitem definir as soluções de uma dada vizinhança, mas depois é preciso verificar se as soluções obtidas são admissíveis para o problema. De seguida serão apresentadas as restrições do modelo e uma explicação de como o algoritmo foi implementado para verificar cada uma delas:

1. Verifica se todos os clientes prioritários estão na solução – é criada uma lista com os clientes prioritários e verifica-se se estes estão todos incluídos na solução.

2. Valida se o peso total das rotas não excede o peso máximo permitido – primeiro define-se o tipo de veículo que realiza cada rota, consoante os tipos de clientes a servir. Depois de ser atribuído o veículo à rota, calcula-se o peso total de cada rota e verifica-se se o peso máximo não é excedido.

3. Valida se a quantidade de produto não excede a capacidade do veículo – tal como na restrição do peso, define-se o veículo que realiza cada uma das rotas. A seguir calcula-se a quantidade total de cada tipo de produto para cada rota e verifica-se se a quantidade total de produto não excede a capacidade do veículo. Se isso não ocorrer será ainda necessário verificar se cada tipo de produto cabe no compartimento dos veículos.

4. Verifica se as rotas não excedem o tempo máximo permitido – calcula-se a duração de cada rota e verifica-se se excede o tempo máximo definido. Para calcular a duração da rota, considera-se o tempo de viagem e o tempo de serviço para realizar a carga e a descarga do produto.

5. Verifica se os clientes são servidos dentro da janela temporal – Calcula-se o horário de chegada do veículo a cada cliente e valida-se se essa hora está dentro do intervalo temporal para a descarga definido para o cliente.

4.3.9 Algoritmo

Considerando os conceitos da pesquisa tabu e as restrições associadas ao problema em estudo obtém-se o algoritmo 3. Nesta secção também será explicado como se lidou com os clientes que podem ser servidos mais do que uma vez por dia e será apresentada uma estratégia para reduzir o número de clientes na pesquisa tabu.

Nas restrições apresentadas na secção 4.1 é referido que um cliente só pode ser visitado uma vez por dia, salvo casos excecionais. Esta exceção é aplicada quando a quantidade total de produto encomendada pelo cliente é superior à capacidade do veículo. Para não ser adicionada uma nova dimensão ao problema, ou seja, considerar que os clientes podem ser servidos por mais do que um veículo por dia, decidiu-se que o algoritmo iria interpretar como se fossem clientes diferentes. Por exemplo, se consideramos que um cliente faz a encomenda de uma carga do veículo completa do produto A e outra do produto B. A implementação realizada considera que a primeira carga é para o número de cliente definido e cria um novo para a segunda carga, sendo assim as matrizes de distâncias e de tempo de viagem têm que ser atualizadas para este novo cliente. Sendo que no final o número de cliente criado será convertido novamente para o número de cliente original.

Algoritmo 3: Pesquisa tabu para o VRP - Galp

```
Determina a solução inicial com um algoritmo do tipo greedy;
enquanto o critério de paragem não for acionado faça
    repita
        Define a nova solução da vizinhança a ser explorada;
        se todos os clientes prioritários estão incluídos nas rotas então
            se capacidade dos produtos nas rotas não excede a capacidade máxima dos
              veículos então
                se peso total dos produtos nas rotas não excede o peso máximo permitido
                  dos veículos então
                    se a duração das rotas não excede o tempo máximo permitido por
                      rota então
                        se horário de chegada dos clientes está dentro da janela temporal
                          então
                            se distância total das rotas for menor que a melhor distância
                              obtida na vizinhança então
                                se o movimento não é tabu então
                                    A solução é definida como a melhor solução da
                                      vizinhança;
                                senão
                                    se distância total das rotas for menor que a melhor
                                      distância obtida durante a pesquisa (critério de
                                      aspiração) então
                                        A solução é definida como a melhor solução da
                                          vizinhança;
    até serem realizados todos os movimentos da vizinhança;
    Atualiza a lista tabu;
    se a distância total da solução obtida na vizinhança for inferior à melhor distância
      obtida durante a pesquisa então
        A solução é definida como a melhor solução obtida (solução incumbente);
Devolver a solução incumbente;
```

O exemplo apresentado é um caso simples onde se considerou apenas dois tipos de produto, mas na realidade os clientes fazem normalmente encomendas com vários tipos de produtos. Nesses casos optou-se por ordenar as quantidades dos produtos encomendas dos clientes por ordem decrescente. Quando a capacidade do veículo é ultrapassada começam a ser criados novos clientes um para cada tipo de produto. A tabela 4.1 tem um exemplo ilustrativo para estes casos. A soma dos litros do produto A com o produto B excede os 36000 L, a capacidade máxima do veículo, portanto vai-se atribuir um número de cliente diferente para cada um dos produtos, como está representado na tabela 4.2.

Cliente	Tipo de produto	Quantidade de produto em L
1	A	25000
1	B	17000
1	C	15000
1	D	7000

Tabela 4.1: Distribuição inicial

Cliente	Tipo de produto	Quantidade de produto em L
1	A	25000
2	B	17000
3	C	15000
4	D	7000

Tabela 4.2: Distribuição final

Durante os testes do algoritmo verificou-se que existem rotas que servem apenas um cliente, uma vez que a quantidade de produto encomendada por esses clientes corresponde à carga completa do veículo. Como não é possível inserir mais clientes nas rotas quando se está perante estes casos, optou-se por fazer a pesquisa tabu sem estas rotas uma vez que não é possível otimizá-las, tornando assim o algoritmo mais eficiente. Os clientes que verificam estas características são identificados antes de se iniciar a pesquisa tabu. Depois da execução do algoritmo, as rotas com esses clientes são adicionadas à solução final obtida pela pesquisa tabu.

4.3.10 Estratégias de pesquisa da vizinhança

As regras de definição da vizinhança são cruciais no desempenho da pesquisa tabu. Regras complexas podem produzir um elevado número de soluções vizinhas, mas penalizam o tempo computacional de cada iteração. Regras simples tendem a produzir menos soluções vizinhas, mas são menos pesadas computacionalmente. Por essa razão é comum aplicar diferentes estratégias na forma como as soluções numa vizinhança são exploradas. Essas diferentes estratégias são depois comparadas com o intuito de identificar a que produziu melhores resultados. No caso deste trabalho foram estudadas foram as seguintes estratégias:

1. **Melhor solução da vizinhança** - A primeira estratégia aplicada é a mais comum e explora toda a vizinhança. Ela consiste em escolher na vizinhança a solução que apresenta melhor valor para a função objetivo, neste caso será a solução com menor distância total na vizinhança.

2. **Primeira melhoria da solução atual** - Nesta estratégia não se explora toda a vizinhança, porque é selecionada a primeira solução da vizinhança que tiver uma distância total menor do que a solução atual. Quando na vizinhança não for encontrada uma distância menor do que a atual a pesquisa reinicia com uma solução guardada anteriormente (*multistart*, 4.3.6).

3. **Primeira melhoria da solução atual + intensificação** - Esta estratégia pode ser considerada um híbrido entre as duas anteriores, isto porque a base desta estratégia é igual à estratégia 2, ou seja, escolhe-se a primeira solução que melhorar a solução

atual. Mas como assim a pesquisa é muito parcial, decidiu-se aplicar uma estratégia de intensificação que será aplicada ao longo da pesquisa. De 10 em 10 iterações o algoritmo faz uma pesquisa completa à vizinhança em que é selecionada a melhor solução obtida (estratégia 1). No final da pesquisa é ainda realizada uma pesquisa mais intensa à solução obtida, usando a estratégia 1 durante um determinado número de iterações.

5

Um estudo sobre a dupla vizinhança na pesquisa tabu

Paralelamente ao desenvolvimento do algoritmo para a distribuição na Galp, foi testada uma nova abordagem para a pesquisa tabu. A ideia é a criação de duas vizinhanças, onde a primeira é criada da forma habitual, ou seja, a pesquisa movimenta-se para o ponto da vizinhança em que o valor da função objetivo é melhor e na segunda vai alternando entre a seleção do melhor e pior valor da função objetivo. Ao longo deste capítulo esta nova estratégia será apresentada em detalhe.

5.1 Motivação

O que levou a considerar o estudo desta nova estratégia foi o facto de muitas das vezes a pesquisa tabu demorar a afastar-se dos ótimos locais. Com esta proposta pretende-se que a pesquisa consiga escapar da zona de atração de um ótimo local, mais rapidamente permitindo assim que o espaço de pesquisa seja mais explorado.

Para ilustrar esta ideia de forma mais clara, na figura 5.1 está representada uma função sendo o objetivo determinar o mínimo global. Considerando duas vizinhanças, a primeira utiliza a estratégia clássica, neste caso será selecionar a solução com o valor mínimo da vizinhança e a segunda vai alternado entre selecionar o valor máximo e mínimo.

No exemplo, para a primeira vizinhança a pesquisa tabu começa na solução inicial, x_0 e considerando como vizinhança o intervalo assinalado, o melhor ponto na vizinhança é x_1. Na segunda iteração o melhor ponto na vizinhança de x_1, $Viz(x_1)$ é o ponto x_2. Este ponto é um ótimo local. Com esta estratégia é a clássica serão necessárias muitas iterações para escapar a este ótimo local porque ao escolher o melhor ponto da vizinhança, que não é tabu, será eleito o ponto mais próximo de x_2. Por isso é que se optou por usar duas vizinhanças. A par da estratégia habitual de escolha da melhor solução da vizinhança (num problema de minimização), implementa-se uma segunda estratégia, que persegue uma sequência de piores soluções até encontrar um máximo local. Uma vez atingido um máximo local a estratégia inverte-se e começam a escolher-se as melhores soluções na vizinhança. Quando é obtido um ótimo local, neste caso um mínimo local, a estratégia

volta a inverter-se mudando o sentido da pesquisa novamente para o máximo da função objetivo. Assim, com duas vizinhanças, a segunda vizinhança partindo da solução inicial x_0, seleciona como próximo centro da vizinhança o ponto x_3. A pesquisa continua e na segunda iteração move-se para a solução x_4, como foi atingido um máximo local a estratégia da segunda vizinhança inverte-se e passa a selecionar a melhor solução na vizinhança. Na próxima iteração como está a minimizar a $Viz(x_4)$ é o ponto x_5 e assim sucessivamente até atingir o ótimo global, x^*. Neste exemplo pode-se verificar que com a utilização desta estratégia a pesquisa pode convergir mais rapidamente para um ponto ótimo, além disso o espaço de vizinhança será mais explorado.

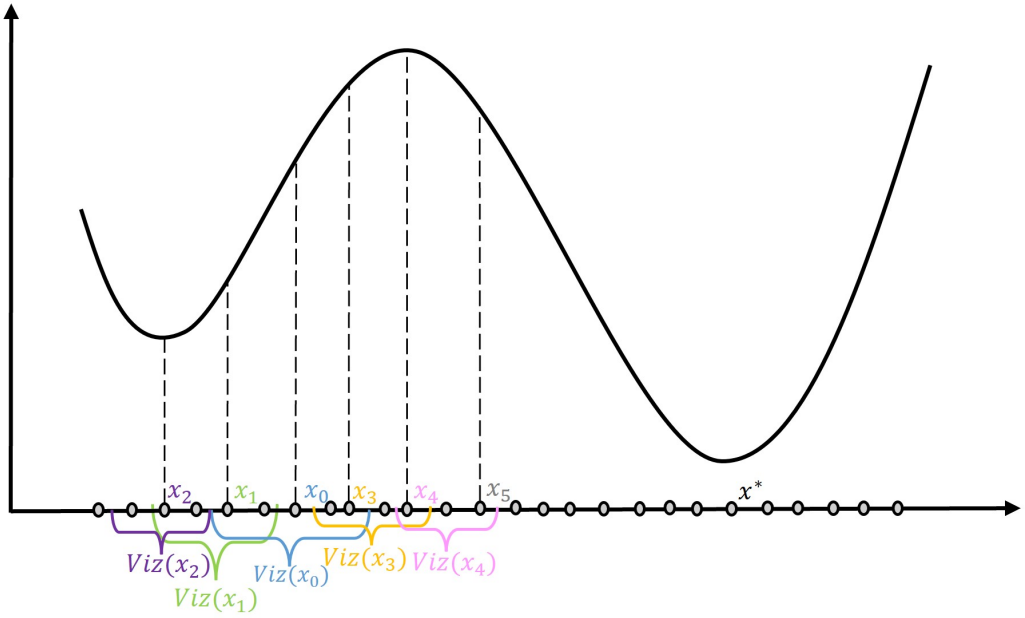


Figura 5.1: Exemplo ilustrativo para a dupla vizinhança

5.2 Descrição

No capítulo 3 foram apresentados alguns dos conceitos da pesquisa tabu e como se pôde verificar existem diversas alternativas que podem ser aplicadas a cada um desses conceitos. Neste capítulo será apresentada uma proposta que consiste numa estrutura de dupla vizinhança. Os outros conceitos mantêm-se, sendo que têm que ser adaptados ao problema em estudo.

Como já foi referido nesta alternativa são consideradas duas vizinhanças, em que as pesquisas são realizadas de forma independente e com estratégias diferentes. Na vizinhança 1 a pesquisa realizada é a mais comum, onde é selecionada a solução da vizinhança que apresenta um melhor valor para a função objetivo. Já a vizinhança 2 varia entre a seleção da melhor solução da vizinhança e da pior, ou seja, alterna-se entre uma direção

de minimização e de maximização com vista a escapar o mais rapidamente possível de uma zona do espaço de pesquisa correspondente a ótimo local.

No início da pesquisa tabu começa-se por escolher na vizinhança 2 o pior valor da função objetivo. Quando é atingido um extremo local, ou seja, quando na vizinhança só houver melhores valores do que da solução atual é que é feita a troca e a pesquisa começa a escolher os melhores valores da vizinhança. A pesquisa volta a mudar quando ao fim de t iterações não houve melhorias no valor obtido pela vizinhança. No algoritmo 4 tem-se a descrição de como são realizadas as trocas na vizinhança 2, para um problema de minimização.

Algoritmo 4: Mudança de estratégia vizinhança 2

```
A pesquisa inicia com a vizinhança a maximizar o valor da função objetivo;
se a pesquisa estiver a maximizar então
    se for atingido um máximo local então
        A estratégia da vizinhança troca e a pesquisa começa a minimizar;
se a pesquisa estiver a minimizar então
    se não houve melhorias na solução ao fim de um determinado número de iterações
     então
        A estratégia da vizinhança troca e a pesquisa começa a maximizar;
```

Com a adição de uma nova vizinhança, implica também o surgimento de novos parâmetros, sendo eles o *tabu tenure* e o número de iterações que a pesquisa deve permanecer sem melhorias na vizinhança 2. Como as vizinhanças são independentes então o *tabu tenure* também será, ou seja, as vizinhanças podem ser definidas com valores de *tabu tenure* diferentes. A forma como são determinados os valores dos parâmetros podem variar de acordo com o problema em estudo, no caso do trabalho desenvolvido os parâmetros da pesquisa tabu são definidos no início do algoritmo. Correu-se o programa para diferentes valores do *tabu tenure*, para fazer uma afinação dos parâmetros e determinar o melhor valor para o problema.

5.2.1 Exemplo

Para demonstrar o funcionamento desta nova proposta para a pesquisa tabu, será apresentado um exemplo que descreve os passos do algoritmo. O problema do exemplo será um CVRP, onde o objetivo é minimizar as distâncias percorridas e tem como restrição que a capacidade dos veículos não pode ser excedida.

Este exemplo é constituído por 6 clientes e um depósito central, descrito como 0, ponto onde as rotas têm que começar e acabar. A capacidade máxima dos veículos é 8 unidades de volume (u.v.), tem-se a matriz de distâncias na tabela 5.1 em unidades de distância (u.d.) e a matriz com a o volume em u.v. do produto a recolher em cada cliente na tabela 5.2.

d_{ij}	**0**	**1**	**2**	**3**	**4**	**5**	**6**
0	0	64	58	54	41	58	41
1	64	0	70	95	103	81	40
2	58	70	0	36	92	113	81
3	54	95	36	0	72	112	91
4	41	103	92	72	0	61	71
5	58	81	113	112	61	0	41
6	41	40	81	91	71	41	0

Tabela 5.1: Matriz de distâncias

Cliente	**Capacidade**
1	3
2	2
3	4
4	3
5	2
6	2

Tabela 5.2: Matriz de capacidades

A solução inicial utilizada para inicializar a pesquisa tabu está representada na figura 5.2 e esta foi determinada pelo algoritmo *greedy* anteriormente descrito, obtendo-se uma solução com valor 460 u.d..

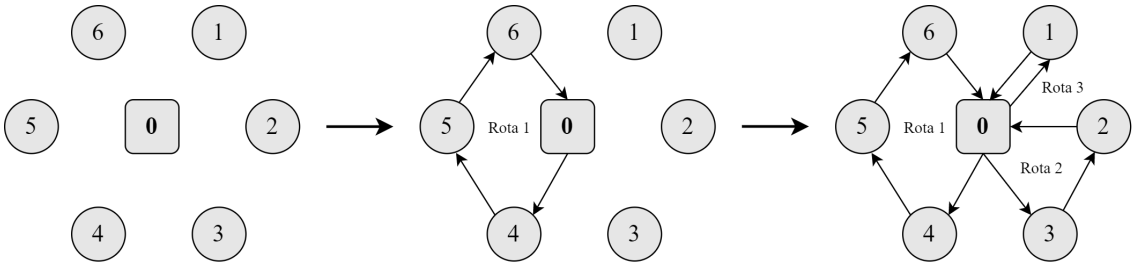


Figura 5.2: Determinação da solução inicial

A pesquisa tabu começa por selecionar a próxima solução na vizinhança da solução inicial. Na tabela 5.3 estão os movimentos admissíveis entre as rotas e os respetivos valores da função objetivo. Por movimentos admissíveis consideram-se os movimentos que quando realizados originam uma solução em que as condições de capacidade sejam respeitadas. Como se trata de um problema de minimização a vizinhança 1 irá selecionar a solução com menor distância total e a vizinhança 2 começa por fazer o oposto, seleciona a solução com maior distância. Neste caso na vizinhança 1 será escolhida a solução que se obteve através do movimento (6,0) entre a rota 1 e 3, que corresponde à inserção do cliente 6 na rota 3 (Figura 5.3) e na vizinhança 2 o (5, 2), onde os clientes 5 e 2 trocam de rota (Figura 5.4) .

Na primeira iteração a vizinhança analisada é a mesma para as duas vizinhanças, uma vez que estas iniciam com a mesma solução, mas a partir da segunda iteração a vizinhança a explorar já será diferente para cada uma das estratégias. Nas tabelas 5.4 e 5.5 estão os novos movimentos de forma obtidos para cada uma das estratégias na segunda iteração.

Na tabela 5.4 estão os movimentos correspondentes à vizinhança 1 o que significa que será escolhida a solução com menor distância. Já na vizinhança 2 (Tabela 5.5) é feito o oposto, uma vez que a vizinhança ainda continua a escolher a pior solução. De notar que o movimento (2,5) não é admissível para a vizinhança 2 uma vez que é um movimento tabu, porque a solução atual foi obtida pela troca dos clientes 5 e 2. As listas tabu das

Rotas	Movimento	Distância total
(1 , 2)	(4, 3)	567
(1 , 2)	(4, 2)	548
(1 , 2)	**(5, 2)**	**607**
(1 , 2)	(5, 0)	542
(1 , 2)	(6, 2)	587
(1 , 2)	(6, 0)	500
(1 , 3)	(4, 1)	457
(1 , 3)	(4, 0)	496
(1 , 3)	(5, 1)	489
(1 , 3)	(5, 0)	504
(1 , 3)	(6, 1)	477
(1 , 3)	**(6, 0)**	**453**
(2 , 3)	(3, 1)	484
(2 , 3)	(3, 0)	513
(2 , 3)	(2, 1)	513
(2 , 3)	(2, 0)	484

Tabela 5.3: Movimentos da vizinhança na 1ª iteração

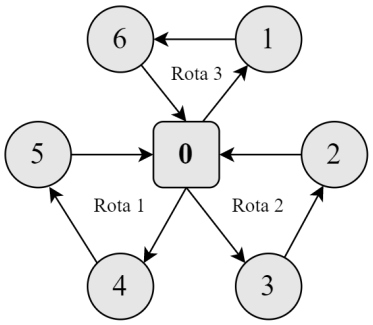


Figura 5.3: Solução obtida na 1ª iteração pela vizinhança 1

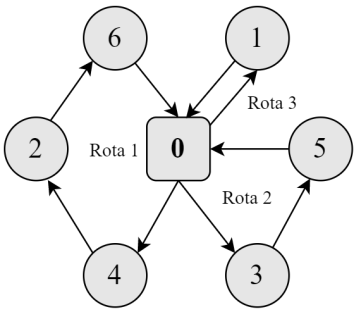


Figura 5.4: Solução obtida na 1ª iteração pela vizinhança 2

vizinhanças são independentes, portanto na vizinhança 1 não existe qualquer restrição à troca destes dois clientes.

A vizinhança 2 muda de estratégia quando atingir um ótimo local, neste caso isso verifica-se na 4ª iteração, em que o valor atual da solução desta vizinhança é 627 u.d., e como se pode verificar na tabela 5.6 na vizinhança desta solução nenhum valor é mais elevado do que o da solução atual, isto significa que na próxima iteração a estratégia da vizinhança 2 vai mudar e vai passar a escolher a solução com menor distância total.

A pesquisa na vizinhança 2 vai continuar a minimizar a distância até esta vizinhança estar 4 iterações sem melhorias. Pelo o gráfico da figura 5.5 pode-se observar que a pesquisa na vizinhança 2 está a minimizar o valor da distância da 4ª à 9ª iteração, que é quando atinge um mínimo local. A partir dessa iteração não há melhorias na distância das soluções obtidas, portanto na 14ª iteração a estratégia da vizinhança 2 volta a mudar

Rotas	Movimento	Distância total
(1 , 2)	(4, 3)	560
(1 , 2)	(4, 2)	541
(1 , 2)	(5, 3)	541
(1 , 2)	(5, 2)	560
(1 , 2)	(5, 0)	488
(1 , 3)	(4, 1)	504
(1 , 3)	(4, 6)	496
(1 , 3)	(4, 0)	480
(1 , 3)	(5, 1)	496
(1 , 3)	(5, 6)	504
(1 , 3)	**(5, 0)**	**433**
(2 , 3)	(3, 1)	538
(2 , 3)	(3, 6)	553
(2 , 3)	(2, 1)	553
(2 , 3)	(2, 6)	538
(2 , 3)	(2, 0)	511

Tabela 5.4: Movimentos da vizinhança 1 na 2ª iteração

Rotas	Movimento	Distância total
(1 , 2)	(4, 3)	500
(1 , 2)	(4, 5)	588
~~(1 , 2)~~	~~(2, 5)~~	~~460~~
(1 , 2)	**(2, 0)**	**618**
(1 , 2)	(6, 5)	618
(1 , 2)	(6, 0)	567
(1 , 3)	(4, 1)	562
(1 , 3)	(4, 0)	612
(1 , 3)	(2, 1)	565
(1 , 3)	(2, 0)	569
(1 , 3)	(6, 1)	573
(1 , 3)	(6, 0)	560
(2 , 3)	(3, 1)	566
(2 , 3)	(3, 0)	584
(2 , 3)	(5, 1)	584
(2 , 3)	(5, 0)	566

Tabela 5.5: Movimentos da vizinhança 2 na 2ª iteração

Rotas	Movimento	Distância total
(1 , 2)	(6, 3)	569
(1 , 2)	(6, 5)	608
(1 , 2)	(6, 2)	572
(1 , 3)	(6, 1)	618
(1 , 3)	(6, 4)	564
(1 , 3)	(6, 0)	616
(2 , 3)	(3, 1)	565
(2 , 3)	(3, 4)	568
(2 , 3)	(5, 0)	516
(2 , 3)	**(2, 0)**	**623**

Tabela 5.6: Soluções da vizinhança 2 na 4ª iteração

e começa a procurar as soluções com piores distâncias. Esta vizinhança continua a fazer estas trocas de estratégias até o critério de paragem ser acionado.

A solução final obtida para este exemplo está representada na figura 5.6 e como se pode ver na figura 5.5 esta solução foi obtida pela vizinhança 1 na segunda interação e na nona na segunda vizinhança. Neste exemplo não existe vantagens em realizar a pesquisa com duas vizinhanças uma vez que a solução ótima é obtida logo na segunda iteração, mas no próximo capítulo serão apresentados os testes realizados com esta nova abordagem, em problemas mais complexos.

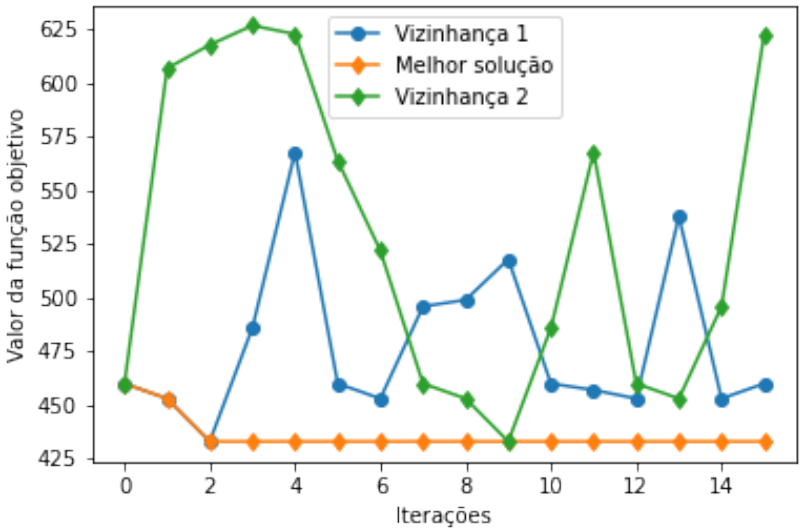


Figura 5.5: Trajetória da pesquisa tabu

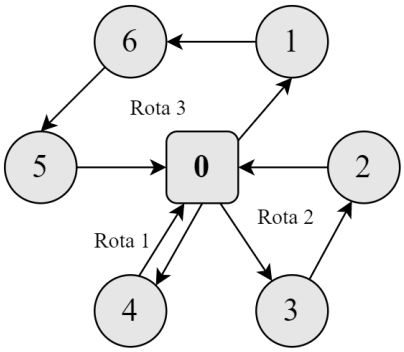


Figura 5.6: Solução final obtida pela pesquisa tabu

6

Resultados computacionais

Neste capítulo serão apresentados os resultados obtidos para a abordagem proposta nos capítulos anteriores e será feita uma comparação entre as rotas efetuadas pela Galp no período em estudo e as obtidas pela pesquisa tabu. A implementação do algoritmo foi feita em *Python* na versão 3.8 e os resultados foram executados numa *Workstation* 2* CPU: Intel® Xeon® Processor E5-2650 v3 (10 Cores 25M Cache, 2.30 GHz), 160GB RAM @ 2,133GHz, as representações gráficas foram obtidas com a utilização do *Tableau*.

6.1 Resultados experimentais

Na apresentação dos resultados o objetivo é comparar as rotas obtidas com as realizadas pela Galp, no entanto existem algumas rotas realizadas pela Galp em que os parâmetros têm um valor superior ao valor limite definido, por exemplo existem rotas em que o peso da carga é superior ao valor limite imposto pela Galp. Por essa razão e para a comparação ser mais fidedigna os valores dos parâmetros que não estavam de acordo com os dados foram alterados, para que fosse possível o algoritmo criar as rotas com as mesmas restrições que as realizadas no passado. Além disso, também foram excluídos alguns clientes, porque as encomendas tinham sido realizadas por veículos com compartimentos de capacidades diferentes e não era possível serem transportadas nos veículos atualmente definidos pela Galp.

6.1.1 Afinação dos parâmetros

Como já foi referido a pesquisa tabu tem parâmetros em que o valor foi definido *a priori*, como o *tabu tenure*. Tal como já foi referido este parâmetro pode influenciar o resultado, ou seja a melhor solução encontrada, pelo que é necessário repetir a corrida do algoritmo para os mesmos dados ensaiando diversos valores para o *tabu tenure*. Este processo é habitualmente definido por "afinação dos parâmetros" e existem duas estratégias para o fazer: *offline* ou *online* [40]. Na inicialização de parâmetros *offline* os valores são definidos antes da execução da pesquisa tabu e na *online* os parâmetros são controlados e atualizados de forma dinâmica ou adaptativa durante a execução.

Para o problema em estudo optou-se por fazer a afinação *offline*, ou seja, foram realizadas diversas execuções onde se alterou apenas o valor do *tabu tenure* e foi escolhido o valor em que se obtiveram os melhores resultados. Como os dados utilizados para testes correspondem a 6 meses seria inviável fazer este procedimento para todos os dias, então decidiu-se fazer a afinação do *tabu tenure* para um mês do conjunto de dados e o valor para os restantes meses foi ajustado de acordo com o valor obtido no mês de testes.

Depois de serem testados diversos valores para o *tabu tenure* verificou-se que o melhor valor variava em função dos dados do dia considerado. Este facto não é surpreendente dado que o número de clientes e o tipo de rotas a realizar são muito variáveis. De forma a tornar o processo de escolha do *tabu tenure* mais automático fez-se uma regressão linear simples entre o número de clientes e o valor do *tabu tenure*, uma vez que parece haver uma relação entre estas duas variáveis. O ideal seria fazer a afinação *online*, visto que o *tabu tenure* não depende só destas duas variáveis, mas a utilização dessa opção iria sobrecarregar mais o algoritmo a nível computacional.

6.1.2 Comparação de estratégias para a pesquisa tabu

No capítulo 4 descreveu-se como o algoritmo foi implementado e na secção 4.3.10 foram apresentadas três estratégias para a seleção da solução na vizinhança. Estas três estratégias foram testadas e os resultados encontram-se na tabela 6.1. A estratégia em que é selecionada a melhor solução da vizinhança é designada por estratégia 1, na 2 escolhe-se a primeira solução que melhora a solução atual e a estratégia 3 seleciona a solução que melhora a solução atual com o processo de intensificação. Estas estratégias foram testadas para os dados de um mês, para avaliar em qual se obtém os melhores resultados e depois de definida qual é a melhor estratégia para o problema é que se executa o algoritmo para todos os meses em teste.

Observando os resultados da tabela 6.1 verifica-se que melhor estratégia a usar depende do dia, mas em termos gerais pode-se concluir que ambas as estratégias melhoram a distância das rotas praticadas pela Galp, no entanto existe uma diferença significativa entre a estratégia 1 e as restantes. As estratégias 2 e 3 não exploram todas as soluções da vizinhança em todas as iterações, o que significa que o tempo de execução é menor. Um dos objetivos do teste destas estratégias era verificar se existia um bom equilíbrio entre o tempo de execução e os resultados finais. Mas observando-os verifica-se que estas estratégias funcionam bem para uns casos, mas o mesmo não acontece com outros, por exemplo quando o número de clientes é mais elevado. Portanto, com base nestes resultados optou-se por definir como estratégia final para executar todos os dados de teste a estratégia 1.

6.1.3 Representação gráfica das rotas

Nesta secção serão demonstrados graficamente os diversos passos da execução do algoritmo, utilizou-se como exemplo um dia do conjunto de dados em que os clientes a ser

Dia	Nº de Clientes	Estratégia 1	Estratégia 2	Estratégia 3	Galp
1	13	1629.47	1630.43	1630.43	1630.43
2	60	6140.61	6215.7	6101.79	6578.63
3	44	5173.64	5273.22	5147.81	5123.18
4	55	6096.64	6132.63	6246.23	6350.66
5	36	4195.15	4203.15	4200.69	4206.3
6	62	7386.12	7801.84	7506.86	7439.34
7	25	2885.83	2937.57	2937.19	2835.32
8	53	5870.27	5924.06	5903.51	5858.16
9	46	5447.97	5282.6	5600.07	5526.09
10	51	5664.71	5660.92	5654.19	6123.33
11	40	4238.96	4484.32	4419.72	4359.26
12	59	6968.46	7047.6	6831.19	7398.01
13	28	3107.21	3104.87	3048.59	3336.07
14	43	5435.71	5433.39	5498.37	5563.53
15	54	5983.03	5978.83	6040.01	6324.32
16	49	6248.14	6056.16	6250.95	6144.94
17	59	7042.63	7213.02	6999.93	7303.56
18	33	3585	3632.41	3663.86	3963.59
19	57	5998.04	5977.72	6000.91	6229.37
20	62	6866.32	7038.7	7011.36	7237.64
21	54	6536.67	6676.32	6626.82	6641.1
22	50	6443.63	6532.95	6606.71	6508.38
23	20	2245.4	2245.5	2245.50	2409.86
Total	1053	121189.61	122483.91	122172.69	125091

Tabela 6.1: Distância total em km das rotas diárias para cada uma das estratégias da pesquisa tabu

servidos estão representados na figura 6.1. Antes de ser iniciada a pesquisa tabu é necessário determinar uma solução inicial, neste caso esta é determinada pelo algoritmo *greedy* e a representação dessa solução para o exemplo apresentado encontra-se na figura 6.2. A pesquisa tabu é executada e o valor da solução selecionada em cada iteração, para a vizinhança considerada, ao longo da pesquisa está representado na figura 6.3. Por fim, quando o algoritmo termina são apresentados os resultados finais e as rotas determinadas pela pesquisa encontram-se representadas na figura 6.4. As representações gráficas das rotas são criadas no *Tableau* e o objetivo é elas serem criadas após a execução do algoritmo para haver um suporte visual das rotas obtidas.

Uma dificuldade verificada na implementação deste algoritmo está relacionada com o tempo de execução, uma vez que ele aumenta significativamente com o aumento do número de clientes (Figura 6.5). A principal razão para a execução do algoritmo demorar mais do que esperado deve-se ao facto de o modelo apresentar várias restrições, sendo que a maior parte delas apresentam alguma complexidade, por exemplo para verificar se a encomenda excede a capacidade do veículo não basta validar se a soma da quantidade

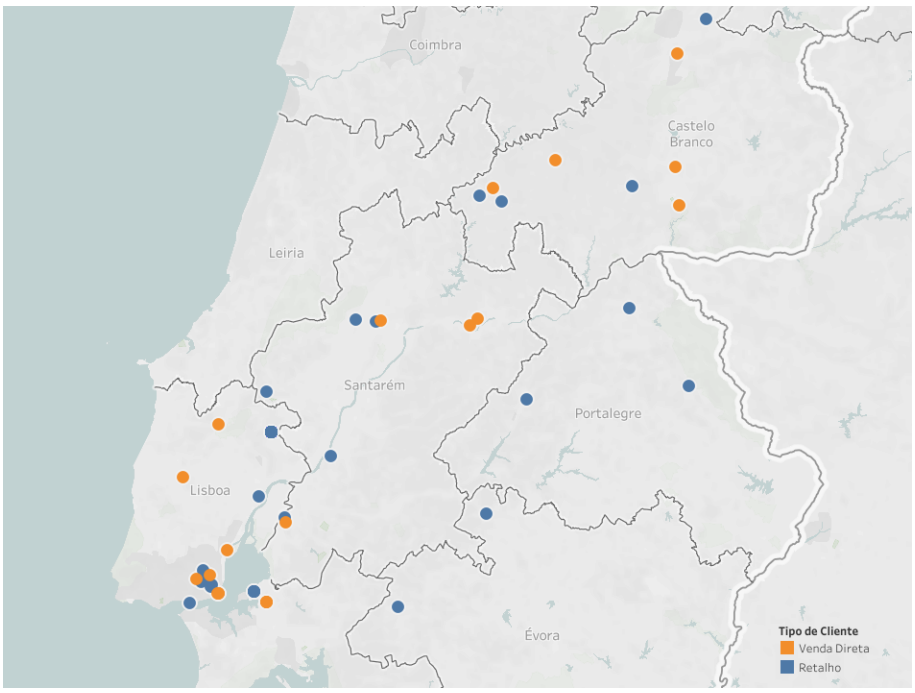


Figura 6.1: Clientes da Galp a servir num dia

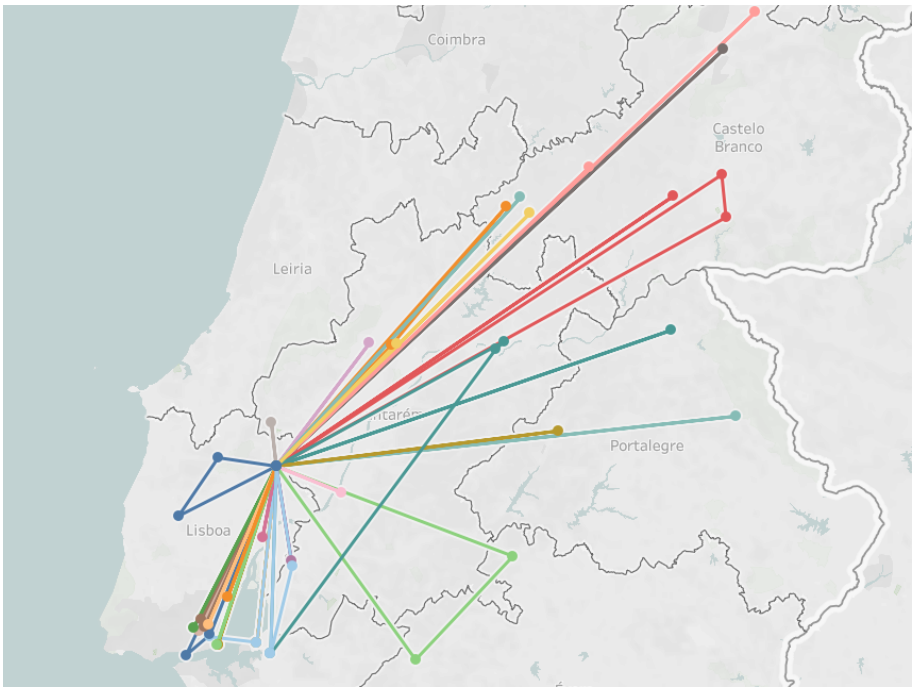
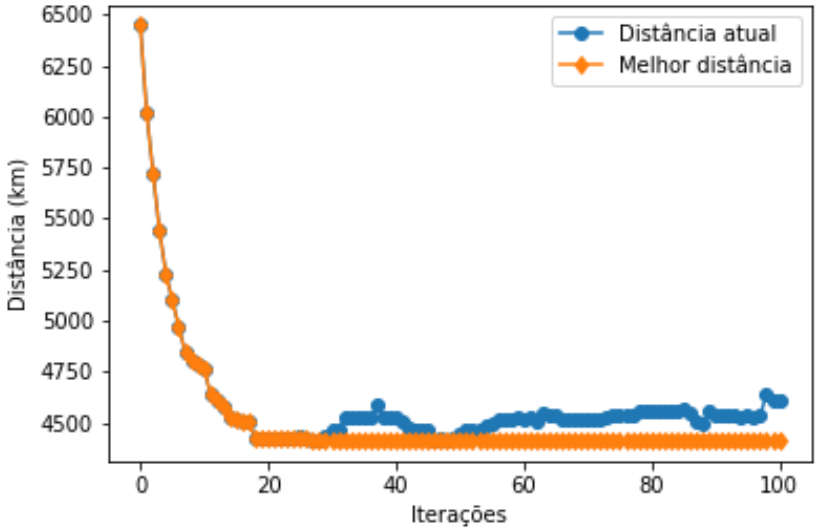
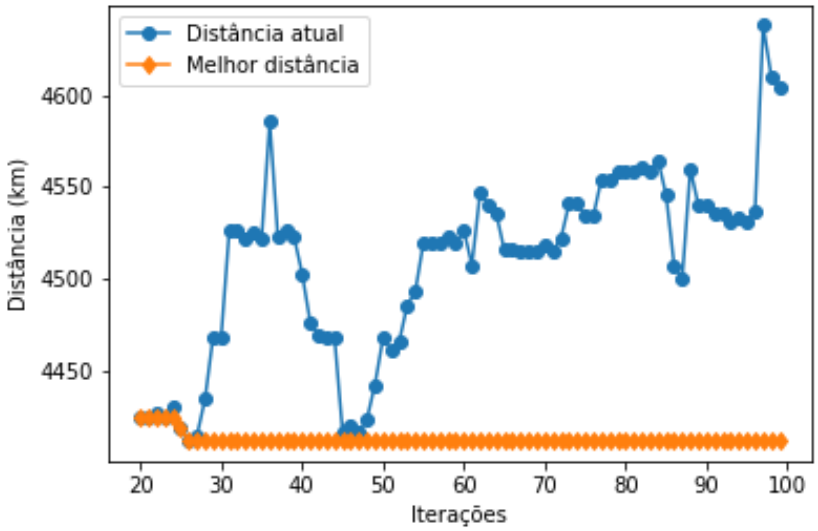


Figura 6.2: Rotas obtidas pelo algoritmo *greedy* (Solução inicial)



(a) Nas primeiras 100 iterações



(b) Da 20ª à 100ª iteração

Figura 6.3: Trajetória da pesquisa tabu

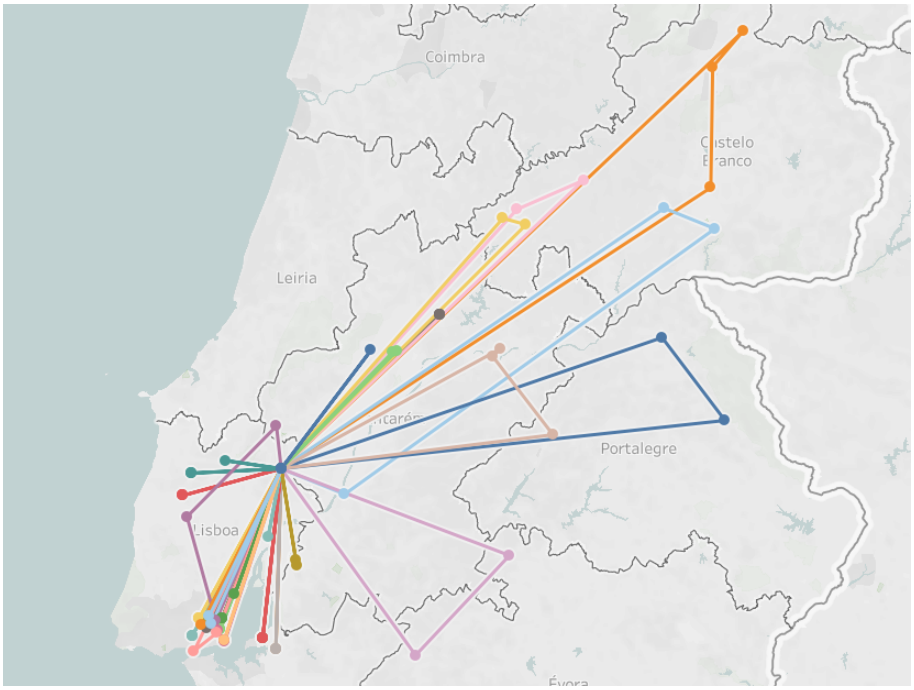


Figura 6.4: Rotas obtidas pela pesquisa tabu (Solução final)

encomendada é menor que a capacidade do veículo, neste caso também é preciso verificar se é possível distribuir os diversos tipos de produto pelos compartimentos do veículo. Como estas restrições têm que ser verificadas para todas as soluções da vizinhança, o tempo computacional tem por essa razão um largo incremento.

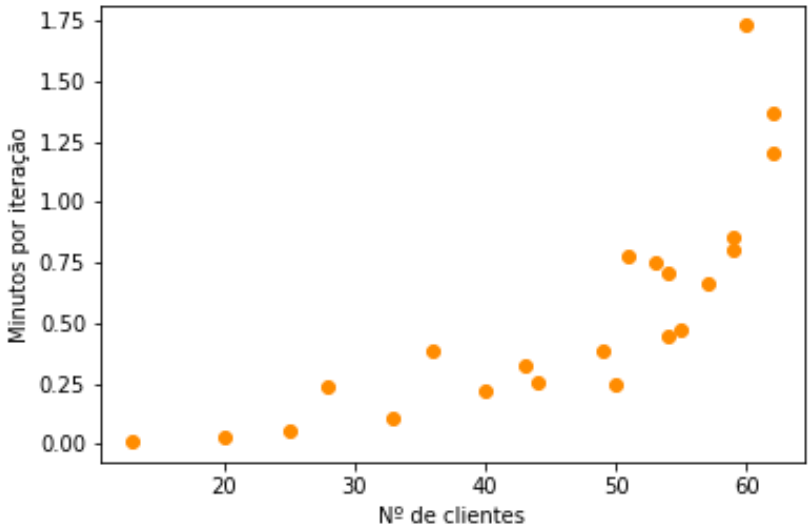


Figura 6.5: Tempo de execução *vs* número de clientes por rota

6.2 Resultados finais

Depois de definido como será o algoritmo da pesquisa tabu este foi executado para todos os dias em que a Galp realizou rotas de distribuição entre setembro de 2019 e fevereiro de 2020, o que corresponde a 157 dias de trabalho. Os resultados obtidos resultaram da execução do algoritmo da pesquisa tabu com um critério de paragem de 1000 iterações ou de 400 sem melhoria. A vizinhança implementada foi a *CROSS exchange*, com esta estrutura pode-se definir o número de clientes dos subconjuntos que são trocados, no caso do algoritmo executado o tamanho dos subconjunto foi 1, uma vez que quanto maior for o tamanho, maior será o número de soluções na vizinhança. Para este problema não era viável em termos computacionais fazer o aumento do número de clientes a considerar nos subconjuntos.

Depois da execução do algoritmo para todos os dados de teste, por comparação com o histórico, pode-se concluir que houve uma diminuição do número de rotas e da distância total percorrida em relação às rotas praticadas pela Galp. A Galp no período em estudo realizou 4709 rotas o que corresponde a uma distância total de 820 287 km. Com a solução obtida pela pesquisa tabu, estes valores desceram para 4601 rotas e 791 750 km. Isto corresponde a uma redução de 3.5% na distância percorrida e uma diminuição de 2.3% nas rotas realizadas. As capacidades e o número de clientes mantiveram-se, porque foram servidos os mesmos clientes e como as encomendas têm que ser entregues na totalidade a quantidade de produto entregue também não varia.

Para verificar a qualidade do algoritmo desenvolvido e para fazer a comparação das rotas obtidas com as das praticadas pela Galp serão utilizados os indicadores descritos na secção 1.3, sendo que na tabela 6.2 se encontram os resultados obtidos para cada um deles.

	Galp	**Algoritmo**
Lote médio por rota	30083.39	30789.55
Rácio distância/rota	174.2	172.08
Rácio distância/capacidade	0.0058	0.0056
Número médio de entregas por rota	1.451	1.485

Tabela 6.2: Comparação entre as rotas obtidas pelo algoritmo e as praticadas pela Galp

Comparando os resultados verifica-se que o lote médio por rota aumentou, o que significa que em média a quantidade de produto entregue por rota aumentou, passando de aproximadamente 30 000 L para 30 800 L. Já seria de esperar um aumento uma vez que um dos tipos de veículos utilizados para a realização destas rotas tinham menor capacidade. Houve uma ligeira diminuição no número médio de km realizados por rota, passando de 174 km para 172 km. O rácio distância/capacidade também diminuiu uma vez que houve uma redução na distância total. Por fim, pode-se verificar pelo último indicador da tabela que houve um aumento do número de clientes por rota, uma vez que

este indicador corresponde ao número médio de clientes que foram servidos por rota.

6.3 Resultados para a dupla vizinhança

No capítulo 5 foi apresentada uma nova abordagem para executar a pesquisa tabu e nesta secção serão apresentados os resultados obtidos. Os testes foram executados para uma amostra dos dados da Galp e para além disso também se testou para um problema clássico de otimização, o problema da mochila (*Knapsack problem*) é simples de formular e tem muitas aplicações práticas interessantes, como por exemplo o de otimizar investimentos.

Este é um problema de otimização muito conhecido e consiste em encher uma mochila com objetos com um determinado peso e valor. O objetivo do problema é encher a mochila com objetos com o maior valor possível, mas o peso dos objetos não pode ultrapassar o peso máximo suportado pela mochila. A formulação matemática do problema é a seguinte:

N = Número de objetos
P = Capacidade da mochila
p_i = peso do objeto i, para $i = 1,...,N$
v_i = valor do objeto i, para $i = 1,...,N$

$$x_i = \begin{cases} 1, & \text{se for colocado o objeto } i \text{ na mochila} \\ 0, & \text{caso contrário} \end{cases} \text{, para } i = 1,...,N$$

$$Maximizar \sum_{i=0}^{N} v_i x_i$$

Sujeito a:

$$\sum_{i=0}^{N} p_i x_i \leq P$$

$$x_i \in \{0,1\}, \qquad i = 1,...,n$$

Os conjuntos de dados utilizados para testar o problema da mochila foram retirados de [18], onde foram utilizados os conjuntos de pequena e grande dimensão. A vizinhança utilizada para os testes foi simples, neste problema os itens são ou não incluídos na mochila, portanto a vizinhança consistia em trocar o estado de um item. Se ele estava incluído na solução passava a não estar e vice-versa. Para a afinação do *tabu tenure* usou-se o procedimento que tinha sido utilizado para o problema de VRP, ou seja, foram testados diversos valores para o *tabu tenure* em cada um dos conjuntos de dados e foi escolhido o melhor. De notar que a segunda vizinhança tem mais um parâmetro que precisa de

ser ajustado que é o número de iterações que a vizinhança 2 deve permanecer sem melhorias quando está a operacionalizar a estratégia de selecionar as melhores soluções da vizinhança.

Durante a realização dos testes para determinar o melhor valor do *tabu tenure*, verificou-se que o valor deste parâmetro é geralmente mais baixo na vizinhança 2 do que na vizinhança 1. Na figura 6.6 pode-se observar que quando o valor do *tabu tenure* é mais baixo os resultados da vizinhança 2 são melhores do que os da vizinhança 1. O *tabu tenure* tem este comportamento porque com a estrutura da vizinhança 2 é menos provável que a pesquisa entre num ciclo, portanto não é necessário limitar tantos elementos como na vizinhança 1, já que esta tem mais tendência a criar ciclos.

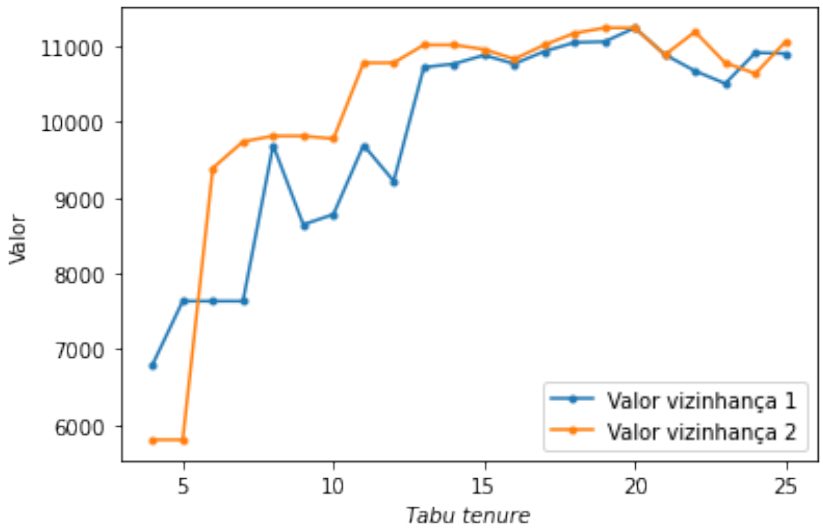


Figura 6.6: Variação do resultado obtido em função do *tabu tenure*

Os resultados obtidos para cada uma das vizinhanças como a iteração em que a melhor solução foi obtida encontram-se na tabela 6.3 e os resultados apresentados na representação gráfica anterior correspondem ao conjunto dados da instância 12 da tabela. Através da sua análise podemos concluir que ambas as vizinhanças obtiveram o valor ótimo para o exemplo apresentado, no entanto a vizinhança 1 precisou um maior número de iterações para obter a solução ótima.

Analisando os resultados obtidos pode-se concluir que a utilização da dupla vizinhança pode apresentar melhores resultados com um menor número de iterações para conjuntos de dados de grandes dimensões. Nos conjuntos de dados mais pequenos a utilização de uma pesquisa tabu comum parece ser suficiente.

Visto que se obteve dados promissores para esta nova abordagem, então também será testada para um caso real, que neste caso será o problema em estudo da distribuição na Galp.

Na tabela 6.4 encontram-se os resultados obtidos para a dupla vizinhança para um

ID	Nº de elementos	Valor ótimo	Valor Vizinhança 1	Iteração Vizinhança 1	Valor Vizinhança 2	Iteração Vizinhança 2
1	10	295	295	19	295	31
2	20	1024	1024	0	1024	0
3	4	35	35	4	35	6
4	4	23	23	0	23	0
5	15	481.07	481.07	0	481.07	0
6	10	52	52	11	52	22
7	7	107	107	0	107	0
8	23	9767	9767	32	9767	26
9	5	130	130	0	130	0
10	20	1025	1025	0	1025	0
11	100	9147	9147	127	9147	110
12	200	11238	11238	3725	11238	700
13	500	28857	25806	4968	24500	1998
14	1000	54503	42499	1219	43701	331
15	2000	110625	67743	295	67000	209

Tabela 6.3: Resultados obtidos para a dupla vizinhança no problema da mochila

conjunto de dias da distribuição da Galp. Observando a tabela pode-se concluir que para a maioria dos casos houve uma melhoria na solução final com a utilização desta abordagem. Portanto a dupla vizinhança pode ser uma nova a estratégia a ser considerada na implementação de uma pesquisa tabu, no entanto tem que ser ter em atenção, quando se utiliza duas vizinhanças o esforço computacional será maior, logo como em qualquer outra estratégia na pesquisa tabu tem que haver um equilíbrio entre os resultados e os recursos computacionais como tempo e memória.

Dia	Distância Vizinhança 1	Iteração Vizinhança 1	Distância Vizinhança 2	Iteração Vizinhança 2
1	1629.46	158	1630.43	212
2	6140.61	93	6059.73	118
3	5173.64	15	5173.64	16
4	6096.64	157	6245.82	29
5	4195.15	227	4187.8	75
6	7386.12	162	7380.91	177
7	2885.83	37	2885.83	17
8	5870.27	108	5779.39	215
9	5447.97	156	5311.9	23
10	5664.71	181	5619.28	58
11	4238.96	471	4258.82	668
12	6968.46	96	6948.04	164
13	3107.21	7	3049.22	71

Tabela 6.4: Distância total em km das rotas diárias para a pesquisa tabu e dupla vizinhança

7

Conclusões

O presente trabalho teve como objetivo criar um algoritmo para determinar as rotas de distribuição diárias da Galp, que será uma ferramenta de apoio à decisão. A implementação do algoritmo foi feita para que este fosse adaptável, ou seja, os dados de *input* podem ser alterados de acordo com as características definidas para cada uma das rotas.

O problema em estudo enquadra-se num tipo de problemas, designados por VRP e existem diversos métodos de otimização que podem ser aplicados a este tipo de problemas, sendo que se dividem em duas categorias principais, os algoritmos exatos e as heurísticas. Como os métodos exatos requerem muito esforço computacional optou-se por utilizar uma metaheurística, mais concretamente a pesquisa tabu, uma vez que é um algoritmo que tem apresentado bons resultados para os problemas de VRP.

Para que as rotas fossem viáveis e pudessem ser realizadas de acordo com a realidade da Galp tiveram que ser definidas algumas restrições no problema. As capacidades e pesos das cargas não podem ser excedidos, também se tiveram em consideração os horários dos clientes e dos motoristas, sendo que destes últimos foi aplicado de uma forma mais simplificada, uma vez que só se fez a limitação das rotas a 9 horas.

Depois de obtidos os resultados e de realizada a comparação entre as rotas obtidas e as realizadas pela Galp pode-se concluir que o algoritmo implementado apresenta boas soluções uma vez que se verificou uma diminuição de 3.5% na distância percorrida e uma redução de 2.3% no número de rotas realizadas. Com as rotas obtidas conseguiu-se aumentar a quantidade de produto transportado e o número de clientes visitados por rota. Estas melhorias são vantajosas para a Galp uma vez que com a realização destas rotas reduz os seus custos de transporte.

Para além dos objetivos propostos inicialmente ainda se testou uma nova abordagem para a pesquisa tabu, que consiste na utilização de duas vizinhanças independentes. Acabou-se por verificar que esta é uma boa estratégia e que pode ser aplicada a diversos tipos de problemas uma vez que já é uma das características da pesquisa tabu. No entanto apesar de os resultados obtidos para esta abordagem serem melhores do que com os conseguidos pela pesquisa tabu comum esta não é a melhor opção para ser aplicada ao problema em estudo uma vez que esta estratégia requer mais esforço computacional

e como o problema em estudo tem muitas restrições não seria a melhor alternativa a ser aplicada. No entanto acreditamos no mérito desta nova proposta e por essa razão submetemos para publicação numa revista da especialidade um artigo com a apresentação do trabalho desenvolvido com a dupla vizinhança na pesquisa tabu.

7.1 Trabalho futuro

Para um trabalho futuro existem diversos tópicos que podem ser explorados, uma vez que a heurística utilizada é muito flexível e existe margem para melhorar a pesquisa. Para isso podem ser estudadas outras abordagens, como por exemplo a alteração da vizinhança ou incluir estratégias de diversificação e/ou intensificação.

Relativamente ao problema em estudo só se considerou um depósito central, uma vez que se desenvolveu um algoritmo para servir apenas uma área regional, para um trabalho futuro seria interessante fazer uma extensão deste algoritmo para que fossem determinadas as rotas para toda a área de distribuição da Galp. Isso implicaria que o modelo não considerasse apenas um depósito, mas sim vários, que iriam corresponder aos parques de armazenamento da Galp. Esta alteração iria facilitar a execução e também iria criar uma maior flexibilidade, uma vez que não se está a limitar as entregas a um único depósito, por exemplo se um cliente está no limite de duas regiões, provavelmente seria equivalente ser servido por veículo proveniente de uma região ou da outra.

Outra alteração que podia ser considerada para um trabalho futuro está relacionada com a afinação dos parâmetros da pesquisa tabu. No trabalho realizado para afinação dos parâmetros utilizou-se a estratégia *offline*, onde os valores dos parâmetros são definidos antes da execução da pesquisa tabu, mas como também já foi referido, o ideal seria utilizar a estratégia *online*, onde os parâmetros seriam atualizados ao longo da pesquisa dependendo dos registos históricos.

Bibliografia

[1] J. Cheeneebash e C. Nadal. "Using Tabu Search Heuristics in Solving the Vehicle Routing Problem with Time Windows: Application to a Mauritian Firm". Em: *University of Mauritius Research Journal* 16 (2010), pp. 448–471.

[2] G. Clarke e J. Wright. "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". Em: *Operations Research* 12 (1964), pp. 568–581.

[3] J. Cordeau, M. Gendreau e G. Laporte. "A tabu search heuristic for periodic and multi-depot vehicle routing problems". Em: *Networks* 30 (1997), pp. 105–119.

[4] J. Cordeau, G. Laporte e A. Mercier. "A unified tabu search heuristic for vehicle routing problems with time windows". Em: *Journal of the Operational Research Society* 52 (2001), pp. 928–936.

[5] J. Cordeau et al. "A guide to vehicle routing heuristics". Em: *Journal of the Operational Research Society* 53 (2002), pp. 512–522.

[6] J.-F. Cordeau e G. Laporte. "Tabu Search Heuristics for the Vehicle Routing Problem". Em: *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*. Ed. por R. Sharda et al. Boston, MA: Springer US, 2005, pp. 145–163.

[7] G. Dantzig e J. H. Ramser. "The Truck Dispatching Problem". Em: *Management Science* 6 (1959), pp. 80–91.

[8] *EU Transport in figures*. Luxembourg: Publications Office of the European Union, 2021.

[9] M. Gendreau, A. Hertz e G. Laporte. "A Tabu Search Heuristic for the Vehicle Routing Problem". Em: *Management Science* 40.10 (1994), pp. 1276–1290.

[10] M. Gendreau, G. Laporte e J.-Y. Potvin. "9. Vehicle routing: modern heuristics". Em: *Local Search in Combinatorial Optimization*. Ed. por E. Aarts e J. K. Lenstra. Princeton University Press, 2018, pp. 311–336. DOI: doi:10.1515/9780691187563-012.

[11] M. Gendreau e J.-Y. Potvin. "Tabu Search". Em: *Handbook of Metaheuristics*. Springer International Publishing, 2019, pp. 37–55.

[12] F. Glover. "Future paths for integer programming and links to artificial intelligence". Em: *Comput. Oper. Res.* 13 (1986), pp. 533–549.

[13] F. Glover. "Tabu Search - Part I". Em: *INFORMS J. Comput.* 1 (1989), pp. 190–206.

[14] F. Glover. "Tabu Search - Part II". Em: *INFORMS J. Comput.* 2 (1990), pp. 4–32.

[15] F. Glover et al. "Tabu search: effective strategies for hard problems in analytics and computational science". Em: *Handbook of Combinatorial Optimization, 2nd ed, vol. XXI* (jan. de 2013), pp. 3261–3362. DOI: 10.1007/978-1-4419-7997-1_17.

[16] P. Hansen e N. Mladenović. "Variable Neighborhood Search". Em: *Handbook of Heuristics*. Ed. por R. Martí, P. M. Pardalos e M. G. C. Resende. Cham: Springer International Publishing, 2018, pp. 759–787. ISBN: 978-3-319-07124-4. DOI: 10.1007/978-3-319-07124-4_19.

[17] A. Hertz e D. Werra. "The tabu search metaheuristic: How we used it". Em: *Annals of Mathematics and Artificial Intelligence* 1 (2005), pp. 111–121.

[18] *Instances of 0/1 Knapsack Problem*. URL: http://artemisa.unicauca.edu.co/~johnyortega/instances_01_KP/.

[19] G. Laporte. "The vehicle routing problem: An overview of exact and approximate algorithms". Em: *European Journal of Operational Research* 59 (1992), pp. 345–358.

[20] G. Laporte et al. "Classical and modern heuristics for the vehicle routing problem". Em: *International Transactions in Operational Research* 7 (1999), pp. 285–300.

[21] C. Liong et al. "Vehicle routing problem: Models and solutions". Em: *Journal of Quality Measurement and Analysis* 4 (jan. de 2008), pp. 205–218.

[22] J. M. Lourenço. *The NOVAthesis Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf.

[23] F. K. Miyazawa. "Programação inteira". Em: (2003).

[24] N. Mladenović e P. Hansen. "Variable neighborhood search". Em: *Comput. Oper. Res* 24.11 (1997), pp. 1097–1100. ISSN: 0305-0548.

[25] K. Nonobe e T. Ibaraki. "A tabu search approach to the constraint satisfaction problem as a general problem solver". Em: *European Journal of Operational Research* 106.2 (1998), pp. 599–623. DOI: https://doi.org/10.1016/S0377-2217(97)00294-4.

[26] J. Ochelska-Mierzejewska. "Tabu Search Algorithm for Vehicle Routing Problem with Time Windows". Em: *Data-Centric Business and Applications*. 2020.

[27] *Openrouteservice*. URL: https://openrouteservice.org/.

[28] *OpenStreetMap*. URL: https://www.openstreetmap.org/.

[29] I. Osman. "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem". Em: *Annals of Operations Research* 41 (1993), pp. 421–451.

[30] T. C. Pais e P. Amaral. "Managing the tabu list length using a fuzzy inference system: an application to examination timetabling". Em: *Annals of Operations Research* 194 (2012), pp. 341–363. DOI: https://doi.org/10.1007/s10479-011-0867-6.

[31] H. Pirim, E. Bayraktar e B. Eksioglu. "Tabu Search: A Comparative Study". Em: *Tabu Search*. Ed. por W. Jaziri. Rijeka: IntechOpen, 2008. Cap. 1. DOI: 10.5772/5637.

[32] C. Rego e C. Roucairol. "A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem". Em: 1996.

[33] *Relatório Integrado de Gestão 2020 - Galp*. URL: https://www.galp.com/corp/Portals/0/Recursos/Investidores/SharedResources/Relatorios/pt/2020/GalpRC20RIG.pdf.

[34] Y. Rochat e É. Taillard. "Probabilistic diversification and intensification in local search for vehicle routing". Em: *Journal of Heuristics* 1 (1995), pp. 147–167.

[35] É. Taillard. "Parallel iterative search methods for vehicle routing problems". Em: *Networks* 23 (1993), pp. 661–673.

[36] E. Taillard. "Robust taboo search for the quadratic assignment problem". Em: *Parallel Computing* 17.4 (1991), pp. 443–455. DOI: https://doi.org/10.1016/S0167-8191(05)80147-4.

[37] É. Taillard et al. "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows". Em: *Transp. Sci.* 31 (1997), pp. 170–186.

[38] É. Taillard et al. "Adaptive memory programming: A unified view of metaheuristics". Em: *Eur. J. Oper. Res.* 135 (2001), pp. 1–16.

[39] E. Taillard. "Tabu Search". Em: *Metaheuristics*. Ed. por P. Siarry. Springer International Publishing, 2016, pp. 51–76. ISBN: 978-3-319-45403-0.

[40] E.-G. Talbi. *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009.

[41] P. Toth e D. Vigo. "The Granular Tabu Search and Its Application to the Vehicle-Routing Problem". Em: *INFORMS J. Comput.* 15 (2003), pp. 333–346.

[42] P. Toth e D. Vigo. "The vehicle routing problem". Em: 2001.

[43] D. Werra e A. Hertz. "Tabu search: a tutorial and an application to neural networks". Em: *Or Spektrum* 11 (1989), pp. 131–141.

[44] D. Whitley. "Next Generation Genetic Algorithms: A User's Guide and Tutorial". Em: *Handbook of Metaheuristics*. Springer International Publishing, 2019, pp. 245–274.