FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Um algoritmo adaptativo para o problema robusto de roteamento com retornos

Vitor Emanuel Marques Mendes

PARA APRECIAÇÃO POR JÚRI

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Prof. Dr. Pedro Amorim

Co-orientador: Dra. Alexandra Marques

11 de Julho de 2017

Resumo

Atualmente todas as empresas responsáveis pela sua própria logística têm como grande preocupação a eficiência da mesma. Sejam grandes empresas como a *United Parcel Service (UPS)*, a *SONAE* Indústria, ou mesmo empresas de menor dimensão. Em termos económicos, a parcela da cadeia de valor encarregue do processamento do transporte, tanto da matéria-prima como dos próprios produtos, assume um valor significativo (49%, according to the European Commission). Portanto os esforços aplicados na otimização das rotas realizadas também é grande. Estas rotas, por vezes são realizadas com o objetivo de entregar os seus produtos aos clientes satisfazendo assim as suas encomendas (rotas *outbound*), e por vezes são realizadas com o objetivo de recolher a matéria-prima nos seus fornecedores (rotas *inbound*).

Uma forma de alcançar a eficiência referida é a integração da sua logística, incorporando no planeamento da mesma os dois tipos de rotas: *inbound* e *outbound*. Por outras palavras, as empresas têm a necessidade de aproveitar as deslocações com a finalidade de transporte os seus produtos até aos clientes para no regresso proceder à recolha da matéria-prima necessária ao fabrico dos seus produtos.

O planeamento e otimização de rotas enquadra-se com o problema *Vehicle Routing Problem (VRP)*, e por esse motivo foi estudada a literatura sobre o mesmo. Num primeiro estudo breve, foi abordada toda a literatura relacionada com o objetivo de restringir o problema em questão na dissertação a uma variante do *VRP*. No final deste primeiro estudo, e como o problema da dissertação inclui fornecedores (*backhauls*), foi possível estabelecer um paralelismo com o *VRP with Backhauls (VRPB)*. Mas no problema desta dissertação, em cada rota apenas será necessária a visita de um fornecedor para satisfação de necessidades de matéria-prima. Além disso, os fornecedores só serão visitados se houver algum benefício. Este benefício traduz-se num prémio associado a cada fornecedor, que não é fixo devido às variações na qualidade da matéria-prima, e resultado em incerteza no mesmo. Por isso, um estudo posterior foi focado no *VRPB* e nos métodos de solução do problema mais adequados. No final deste estudo foi possível definir que o método de solução seria a metaheurística *Adaptative Large Neighborhood Search (ALNS)* por forma a lidar com problemas de dimensão elevada. O *VRPB* será adaptado ao nosso tipo de fornecedores e será incluída a otimização robusta para que seja tida em conta a incerteza no prémio. Resumindo, o problema culmina num *Robust VRPB (RVRPB)*. Até à data é a primeira abordagem deste género na literatura, em que a incerteza se dá num prémio atribuído a um fornecedor.

O algoritmo *ALNS* foi desenvolvido e para validação, este foi primeiro adaptado e testado para o *Capacited Vehicle Routing Problem (CVRP)* e depois ao *VRPB*. A adaptação final foi já para o *RVRPB*.

Os resultados finais demonstraram que o algoritmo *ALNS* desenvolvido se mostra mais adequado a problemas reais e de grande dimensão do que a resolução por métodos exatos. Para instâncias de menor dimensão, o algoritmo *ALNS* obteve em quase todos os casos testados o mesmo valor que o método exato, e melhores valores para todos os valores em instâncias maiores.

Abstract

Nowadays, all the companies responsible for their own logistics, wether big companies such as *United Parcel Service (UPS)*, *SONAE* Indústria or even smaller companies, have eficiency as their main concern. Economically speaking, the value represented by the products and raw material transportation in the chain value, plays a meaningful role (49%, according to the European Commission). Accordingly, the effort made for route optimization is meaningful as well. Sometimes, these routes are made aiming for product delivery (outbound routes) and sometimes are made with the goal of raw material picking (inbound routes).

Integrated logistics, with both inbound and outbound routes, is one way to accomplish cost savings. In other words, the companies take advantage of outbound routes that are mandatory to collect the necessary raw materials for their operations.

In this work, a literature review was performed on route planning and otimization, and it was possible to link the dissertation problem to the Vehicle Routing Problem (VRP). In the first phase of the review the main goal was to find the proper VRP variant. The problem includes backhauls, so a parallelism between the problem and the Vehicle Routing Problem With Backhauls was established. But in the problem approached in this thesis, only one visit to each backhaul is required to satisfy the raw material needs. Besides, they are only visited if there is an adjacent benefit from it. This benefit is subjected to uncertainty due to raw material quality fluctuation. As a result, the next review phase was focused on VRPB and on suitable resolution methods for it. In the end of this phase, the metaheuristic Adaptative Large Neighborhood Search (ALNS) was defined as the method to be used to deal with real and large dimension problems. The problem will be based on VRPB and for the uncertainty the robust optimization will be used. In summary, the problem tackled is known as Robust VRP (RVRP). To the best of our knowledge this is the first time that this problem is studied in the literature.

In order to validate the ALNS algorithm, initially it was developed for the Capacited VRP (CVRP) and then was adapted for the VRPB. Finally it was applied to the RVRPB.

The final results obtained proved that the algorithm developed is more appropriate than commercial solvers to solve real and large dimension problems. Although the optimal solution is only guaranteed in exact methods, in the majority of the tested cases the ALNS algorithm accomplished the same results as the exact method. In large dimension instances ALNS showed better performance.

Agradecimentos

Em primeiro lugar quero agradecer ao meu pai que nunca desistiu da ideia de me ver formado, sempre me incentivou e apoiou. Espero que me estejas a ver e que estejas orgulhoso de mim. Em segundo lugar gostaria de agradecer à minha mãe que, apesar de tudo, também sempre me apoiou.

Gostaria também de agradecer à minha namorada que me apoiou nos tempos mais difíceis e que nunca me deixou desmoralizar. A todos os meus camaradas da I101 (eles sabem quem são) pela diversão, entreajuda e apoio em momentos de bloqueio. Ao Pedro Alves que apesar de longe também esteve sempre "presente". A todos os meus amigos de longa data (também sabem quem são) e aos meus colegas das partidas semanais de futebol, por me proporcionarem momentos de descontração.

O meu sincero agradecimento ao meu orientador Prof. Dr. Pedro Amorim. Obrigado por ter apostado em mim, ter partilhado o seu vasto conhecimento comigo, por me ter ajudado incondicionalmente e acima de tudo pela paciência ao longo deste projeto. À minha coorientadora, Dra. Alexandra Marques, por ter acompanhado o meu trabalho e ter, igualmente, partilhado o seu conhecimento através de sugestões pertinentes.

Ao restante grupo ligado ao INESC (Maria João, Eduardo Curcio e Ricardo Soares) o meu muito obrigado pelas vossas contribuições.

Vitor Mendes

"In any moment of decision,
the best thing you can do is the right thing,
the next best thing is the wrong thing,
and the worst thing you can do is nothing."

Theodore Roosevelt

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

ACO	Ant Colony Optimization
ALNS	Adaptative Large Neighborhood Search
CFF	Commodity Flow Formulation
CVRP	Capacited and Distance-Constrained VRP
DA	Determinstc Annealing
GA	Genetic Algorithms
ILS	Iterated Local Search
INESC TEC	Instituto de Engenharia de Sistemas e Computadores - Tecnologia e Ciência
LNS	Large Neighborhood Search
MACS	Multi-ant Colony Optimization
MDVRP	Multiple Depot Vehicle Routing Problem
MILP	Mixed Integer Linear Programming
MPFIH	Modified Push-Forward Insertion Heuristic
MTZ	Miller-Tucker-Zemlin
OVRP	Open Vehicle Routing Problem
OX	Order Crossover
PSO	Particle Swarm Optimization
RAM	Random Access Memory
RCVRP	Robust Capacited and Distance-Constrained VRP
RSPPRC	Robust Shortest Path Problem with Resource Constraints
RVRP	Robust Vehicle Routing Problem
RVRPTW	Robust Vehicle Routing Problem with Time Windows
SA	Simulated Annealing
STD-VRP	Stochastic Time-Dependent Vehicle Routing Problem
SVRP	Stochastic Vehicle routing Problem
TS	Tabu Search
TSP	Travelling Salesman Person
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem
VRPB	Vehicle Routing Problem with Backhauls
VRPD	Vehicle Routing Problem with Deadlines
VRPSST	Vehicle Routing Problem with Stochastic Service Times
VRPTW	Vehicle Routing Problem with Time Windows
λ-LSD	λ-interchange Local Search Descent

Capítulo 1

Introdução

1.1 Enquadramento e motivação

O mercado de transporte é muito relevante na nossa sociedade e tem um impacto significativo na economia. Segundo dados da Comissão da União Europeia recolhidos em 2015, o transporte terrestre de carga representa 49% de entre todo o tipo de transportes terrestres (Union européenne and Commission européenne, 2015). O aumento de competitividade entre empresas de distribuição, da exigência de qualidade por parte dos clientes, dos preços do combustível e dos impostos, levam a que as empresas procurem continuamente aumentar a eficiência da sua distribuição através da melhoria do seu planeamento logístico (Juan et al, 2014). Em particular na indústria da madeira estas operações de transporte têm um peso significativo nos custos da cadeia de valor (Audy et al, 2012; Forsberg et al, 2005; Weintraub et al, 1996). Às operações florestais são associados grandes volumes bem como distâncias de transporte relativamente longínquas e preocupações ambientais no que diz respeito ao mesmo (Audy et al, 2012), o que reforça a necessidade de melhoria do planeamento mencionada anteriormente.

Uma forma de melhorar a eficiência do planeamento é através da inclusão de *backhauling* no planeamento logístico, isto é, a inclusão da passagem em fornecedores de matéria-prima nas rotas de retorno à fábrica de maneira a minimizar o número de viagens em vazio do camião, e também minimizar o número de viagens adicionais para recolha de matéria-prima. Estas viagens adicionais, que fazem parte do processo de transporte, são uma porção razoável dos custos associados à obtenção da matéria-prima e é, por essa razão, uma fonte possível de uma poupança significativa.

Este planeamento conjunto é designado de planeamento integrado da logística de *inbound* e *outbound*. O planeamento constitui um problema de roteamento, e como inclui a preocupação com a passagem nos fornecedores, é modelado num *Vehicle Routing Problem with Backhauls (VRPB)*. Apesar disso, o planeamento não é um processo trivial. Existem vários elementos que em simultâneo conferem complexidade ao mesmo, sendo estes por exemplo:

- Lista diária de clientes a visitar extensa;
- Matéria-prima e receitas de quantidades variáveis;

- Várias fábricas;
- Frota de veículos heterógenea, isto é, com capacidades variáveis.

Destes, irão ser abordados em detalhe os dois primeiros elementos do planeamento integrado da logística de uma fábrica do setor dos produtos derivados da madeira. E para que o *VRPB* se ajuste na perfeição ao problema específico abordado nesta dissertação, este sofre algumas alterações para que se tenha em conta as especificidades do mesmo.

No cado de estudo base, o planeamento diário de *inbound* e *outbound* é realizado manualmente, numa ótica regional e de forma independente entre fábricas. Além disso, o planeamento também é independente no que diz respeito às rotas de *inbound* e *outbound*, isto é, são definidas as rotas de visita aos clientes para satisfação de procura (*outbound*), sem ter em conta as rotas de visita a fornecedores para aquisição de matéria-prima (*inbound*). Esta falha na eficiência do processo será colmatada através do *VRPB*.

Visto o caso de estudo base ser um caso real e de grande escala, a forma para resolver o problema modelado no *VRPB* é resolvido através de uma metaheurística. Isto para que o problema seja resolvido em tempo útil e para que se torne vantajoso utilizar este método de planeamento de rotas em vez do método utilizado no momento. É também vantajoso utilizar uma metaheurística em detrimento de um método exato, visto que o segundo para problemas de grande escala como é o caso, se torna impraticável devido ao tempo necessário pelo método para que se resolva o problema e se obtenha uma solução. A metaheurística utilizada na dissertação é denominada de *Adaptative Large Neighborhood Search (ALNS)*.

Numa abordagem inovadora ao *VRPB*, é introduzida a ideia de um benefício na passagem pelo fornecedor através da atribuição de um prémio ao mesmo, com o objetivo de apenas visitar fornecedores que valham a pena visitar. Esta nova abordagem implica uma nova dificuldade que se traduz na incerteza do prémio atribuído. Para resolver tal dificuldade, e novamente introduzindo inovação, é aplicada a otimização robusta à metaheurística *ALNS* para resolver o *VRPB* com o objetivo de garantir, tendo em conta o nível de conservadorismo do utilizador, que uma solução seja admissível até para o pior caso. Isto é, que a solução encontrada seja admissível mesmo que se verifique que o prémio atribuído aos fornecedores seja o mais baixo possível.

1.2 Objetivos

Esta dissertação está inserida num projeto do Instituto de Engenharia de Sistemas e Computadores - Tecnologia e Ciência (*INESC TEC*) designado *EasyFlow*, que tem como finalidade aumentar a sustentabilidade de cadeias de abastecimento de base florestal, utilizando para tal ferramentas computacionais e tecnologias de sensorização para melhorar a eficiência no uso de recursos disponíveis e melhorar o planeamento logístico sustentável e colaborativo.

O propósito deste trabalho é a aplicação, em simultâneo, de técnicas de investigação operacional e de apoio à decisão para o planeamento robusto e integrado da logística de *inbound* e *outbound* em fábricas onde se constata a necessidade de tal planeamento. Para isso, será desenvolvido um

algoritmo adaptativo que para além da habitual definição de rotas diárias de *inbound* e *outbound*, irá também ter em conta na definição as incertezas inerentes e acomodando variações dos elementos que são fonte de incerteza. Esta definição de rotas *inbound* e *outbound* tentará minimizar os custos de transporte da entrega de produtos aos clientes e, quando possível, proceder à recolha de matéria-prima junto dos fornecedores no regresso à fábrica com custos mínimos.

1.3 Estrutura da dissertação

No capítulo seguinte (2) é feita a revisão bibliográfica à literatura. Numa primeira fase, é brevemente abordada a história do *Vehicle Routing Problem (VRP)* e de seguida o próprio *VRP* e os componentes que o constituem. Em fases seguintes são abordadas as variantes mais importantes para o problema em foco na dissertação, bem como os diferentes métodos de solução do mesmo. Por fim, tem-se uma secção onde as formas de lidar com a incerteza presente no problema são estudadas.

No capítulo 3, é feita a descrição do problema de forma mais detalhada onde são explicadas as suas especificidades e apresentadas as formulações matemáticas que serão utilizadas como ponto de partida para a solução do mesmo.

No capítulo 4, são explicados os passos de todo o desenvolvimento e implementação do método de solução escolhido para o problema, *Adaptative Large Neighborhood Search (ALNS)*. Na secção final deste capítulo está demonstrada a implementação da robustez no método aproximado, bem como um exemplo demonstrativo do funcionamento dos métodos que a executam.

O capítulo seguinte (5) apresenta os resultados que surgem da implementação descrita no capítulo anterior. É também realizada uma discussão dos mesmos.

No capítulo final (6) são retiradas conclusões do trabalho realizado na dissertação e indicadas propostas de possíveis trabalhos futuros.

Capítulo 2

Revisão bibliográfica

O conhecido *Vehicle Routing Problem* (*VRP*) foi escolhido como base para a resolução do problema referido na introdução. Esta escolha de base, além de se dever ao facto de ser um modelo adequado ao planeamento da logística, deve-se também ao facto do elevado sucesso obtido na aplicação prática deste problema.

> "In practice, vehicle routing may be the single biggest success story in operations research."(Golden et al, 2008)

Neste capítulo irá ser abordada a história do *VRP*, bem como o seu modelo clássico e as variantes relacionadas com o problema. Estas variantes surgiram posteriormente, fruto de necessidade de responder a diferentes aspetos do planeamento logístico sob análise. Irão também ser abordados os métodos de solução e a forma de lidar com incerteza inerente aos modelos.

2.1 História do *Vehicle Routing Problem*

Em 1959, Dantzig and Ramser (1959), introduzem pela primeira vez o problema *VRP*, na altura com a designação "*The Truck Dispatching Problem*", que consiste numa generalização do problema *Tavelling Salesman Problem* (*TSP*). O *TSP* é um problema que tenta determinar a menor rota entre um conjunto de cidades a visitar, cada uma delas apenas uma vez, e que tem de regressar obrigatoriamente à cidade de origem. A generalização do *TSP* que gerou o *The Truck Dispatching Problem* foi realizada através da adição de entregas a serem feitas em cada ponto de passagem e que teriam uma determinada quantidade associada. Os meios de entrega eram considerados homogéneos no que à capacidade dizia respeito (Dantzig and Ramser, 1959; Toro et al, 2016).

Mais tarde, em 1964, o modelo desenvolvido em Dantzig and Ramser (1959) sofreu uma alteração ao ser introduzida a condição em que os veículos de entrega são heterogéneos, isto é, têm capacidades diferentes entre si. Desde então este tema tem sido amplamente abordado surgindo diversas variantes, criadas com o objetivo de tornar este problema o mais próximo possível da realidade (Braekers et al, 2016). Nos últimos anos têm surgido também novos avanços e consequentemente novos desafios, fruto do elevado desenvolvimento tecnológico a que se tem assistido (Golden et al, 2008).

2.2 *VRP*

A solução de um VRP tem como objetivo a definição de um conjunto de rotas, em que cada uma é percorrida por um único veículo que começa e acaba num armazém. A solução tem de cumprir todos os requisitos dos clientes e satisfazer a procura (que se assume ser conhecida e fixa à priori) e o seu custo global de transporte deverá ser mínimo (Zachariadis and Kiranoudis, 2012; Cuervo et al, 2014). O problema é então constituído por um conjunto de componentes principais, um conjunto de restrições operacionais que serão impostas na construção da solução, e por objetivos a alcançar no processo de otimização (Toth and Vigo, 2002b).

2.2.1 Componentes

Os componentes constituintes do problema são os seguintes:

- Vértices ou nós:
  - Armazéns - onde está localizada a fonte de produto. É o local de onde saem e regressam os veículos destacados para realizarem as rotas definidas;
  - Clientes ou Fornecedores - onde está localizada a fonte de procura (clientes) ou de matéria-prima (fornecedores);
- Arcos - representam as estradas pelas quais os veículos se deslocam e fazem a ligação entre dois vértices.
- Veículos - representam os elementos capazes de fazer o transporte dos bens entre dois vértices através dos arcos.
- Condutores - são os responsáveis pelos elementos que transportam os bens (veículos) e estão sempre associados aos mesmos.

Tanto os clientes como os veículos têm características que dependem muito do problema em causa. Como exemplo, as características frequentemente associadas aos clientes são: procura a ser satisfeita, períodos de tempo em que a necessidade pode ser satisfeita (*time windows*), tempos limitados para carga e descarga de produtos (*unloading/loading times*) e conjunto de veículos que podem ser usados para servir o cliente. As características normalmente associadas aos veículos são: existência de armazém central para recolha do veículo ou possibilidade de terminar a rota num armazém que não o central, capacidade, possibilidade de divisão do veículo em compartimentos cada um com capacidade própria e com tipos diferentes de produtos a poderem ser carregados, conjunto de arcos do grafo pelos quais pode passar e custos associados à sua utilização (Toth and Vigo, 2002b).

2.2.2 Restrições

As características particulares dos componentes de um problema tratado pela abordagem *VRP* são o que o define e diferencia dos demais, e são traduzidas em restrições operacionais. Estas características podem ser dos veículos, dos clientes, etc.

Algumas destas características associadas aos veículos e aos clientes foram já referidas em 2.2.1. Tal como os clientes, os fornecedores têm características como: disponibilidade de produto (habitualmente matéria-prima), períodos dentro dos quais os produtos podem ser descarregados, tempos limitados para a carga dos produtos e conjunto de veículos que podem ser utilizados para efetuar tal carga.

No que diz respeito às rotas, verificam-se características tais como: limitação de tempo de realização da rota de início ao fim; a distância máxima de uma rota (que depende das restrições temporais); dentro de uma rota pode haver precedência, isto é, uma ordem definida de clientes a visitar por exemplo (Toth and Vigo, 2002b).

2.2.3 Função objetivo

O *VRP* tenta alcançar os objetivos definidos, que podem ser de diversos tipos e por vezes até divergentes. Estes objetivos são mapeados na função objetivo e são tipicamente (Toth and Vigo, 2002b):

- Minimizar o custo de transporte global, tendo em conta os custos fixos associados aos veículos utilizados para o transporte;
- Minimizar os custos variáveis associados à distância percorrida;
- Minimizar o número de veículos a utilizar;
- Balancear o tempo de viagem e carga das diferentes rotas;
- Minimizar as penalizações que resultam de um serviço parcial nos clientes ou de ausência de passagem num *backhaul*.

2.3 Variantes do *VRP* relacionadas com o problema

A variante clássica do *VRP* é a *Capacited VRP* (*CVRP*). Esta é obtida ao ser adicionada a restrição de capacidade máxima aos veículos. A partir desta variante surgem outras consoante a restrição acrescentada.

As variantes podem ser mapeadas consoante as condições específicas adicionais impostas ao *VRP*, como demonstrado na seguinte imagem (retirada de Toth and Vigo (2002b)):

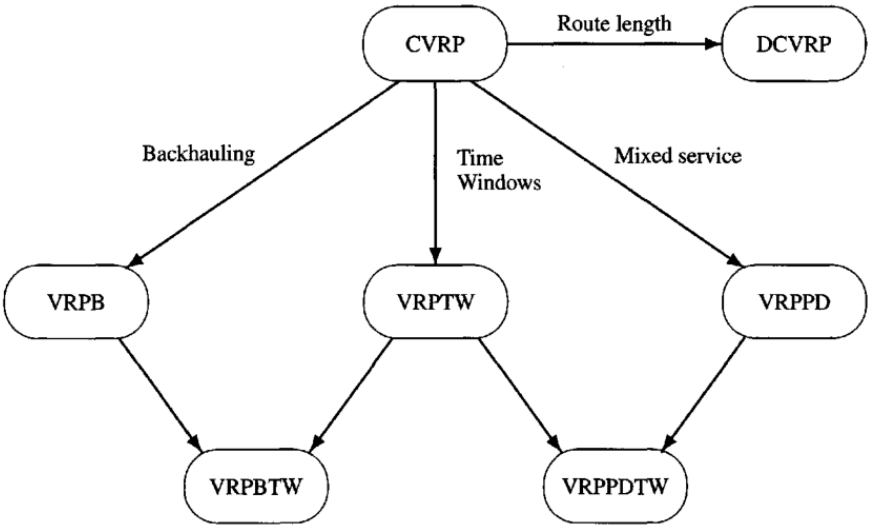

Figura 2.1: Problemas básicos da classe *CVRP* e respetivas interligações (Toth and Vigo, 2002b)

2.3.1 *Capacited Routing Vehicle Problem* (*CVRP*)

Como referido em 2.3 esta é uma variante do *VRP* que surge da adição da restrição de capacidade inerente aos veículos. Esta restrição impõe que a soma das procuras dos clientes presentes na rota não seja superior à quantidade de carga que o veículo destacado para a rota suporta. Este é um problema bastante genérico e que pode ser aplicado em diversas áreas, desde transporte de resíduos sólidos a transporte de produtos derivados da madeira (Akhtar et al, 2017; Amorim et al, 2014).

2.3.2 *Vehicle Routing Problem with Backhauls* (*VRPB*)

Uma das extensões do problema *CVRP* na Figura 2.1 é o *VRPB*, e esta variante contempla tanto pontos de entrega como de recolha. Ambos os pontos são agrupados em *subsets*, um primeiro que reúne todos os clientes que apresentam uma necessidade de produtos, e um segundo que reúne todos os fornecedores que apresentam uma disponibilidade de matéria-prima a ser recolhida. São designados de *linehaul* e *backhaul*, respetivamente.

Este tipo de problemas obedece às seguintes condições:

1. Cada veículo serve exatamente uma rota;
2. Cada rota começa e acaba no armazém;
3. A carga, tanto de *linehaul* como de *backhaul*, não poderá ser superior à capacidade do veículo em questão.
4. Cada rota satisfaz primeiro todos os clientes *linehaul* e só depois do veículo se encontrar vazio os clientes *backhaul* são visitados, se existirem;

5. A distância total percorrida pelos veículos é minimizada.

Note-se que a restrição de precedência de *linehaul* em relação a *backhaul* se deve ao controlo do fluxo de entrada e saída de carga do veículo, visto que a maioria dos veículos são carregados e descarregados pela traseira. Além disso, a larga maioria dos problemas onde há a possibilidade de aplicação do *VRPB* priorizam os clientes *linehaul* aos *backhaul*.

Uma das modelações possíveis para este problema será descrita de seguida.

2.3.2.1 Notação

Seja $G = (V, A)$ onde:

- V é o *set* de vértices $V = \{0\} \cup L \cup B$ e os *subsets* $\{0\}$, L e B são, respetivamente, o armazém e os *subsets* referentes aos clientes *linehaul* e *backhaul*;

- A é o *set* de arcos que são definidos entre dois vértices do *set* V referido.

A cada vértice pertencente a L e B estão associadas, respetivamente, uma quantidade não negativa de procura de produto e uma quantidade de produto a ser recolhido $q_i > 0$ e ao armazém uma quantidade fictícia $q_0 = 0$. Considera-se que o vértice correspondente ao armazém (V_0) contém o *set* de veículos homogéneos K, com capacidade C, de onde todos partem e onde todos regressam depois de visitados todos os clientes. Seja também c_{ij} o custo associado entre i e j e onde $c_{ij} = c_{ji}$ para cada $i, j \in V$ tal que $i \neq j$.

2.3.2.2 Variáveis de decisão

A variável de decisão deste problema é binária e define se um veículo k faz a viagem entre os nós i e j ou não.

$$x_{ij}^{k} \begin{cases} 1, \textit{ se veículo } k \textit{ faz a viagem do cliente } i \textit{ para o cliente } j \\ 0, \textit{ caso contrário} \end{cases} \tag{2.1}$$

2.3.2.3 Função objetivo

Este problema tem como objetivo minimizar o custo total do conjunto de rotas, que neste caso é obtido da seguinte maneira:

$$Min \sum_{k \in K} \sum_{\substack{i,j \in V \\ i \neq j}} c_{ij} \times x_{ij}^{k} \tag{2.2}$$

2.3.2.4 Restrições

As restrições que definem o *VRPB* são as seguintes:

$$\sum_{k\in K}\sum_{\substack{i\in V\\ i\neq j}} x_{ij}^k = 1 \quad \forall j \in V \tag{2.3}$$

$$\sum_{j\in L} x_{0j}^k = \sum_{i\in V_0} x_{i0}^k \quad \forall k \in K \tag{2.4}$$

$$\sum_{i\in V} x_{iu}^k - \sum_{j\in V} x_{uj}^k = 0 \quad \forall u \in V, \forall k \in K \tag{2.5}$$

$$\sum_{\substack{i\in B\\ j\in L}} x_{ij}^k = 0 \quad \forall k \in K \tag{2.6}$$

$$\sum_{k\in K}\sum_{j\in V_0} x_{0j}^k \leq K \tag{2.7}$$

$$\sum_{\substack{i\in L\\ j\in V}} q_i \times x_{ij}^k \leq C \quad \forall k \in K \tag{2.8}$$

$$\sum_{\substack{i\in B\\ j\in V}} q_i \times x_{ij}^k \leq C \quad \forall k \in K \tag{2.9}$$

$$x_{ij}^k \in \{0,1\} \tag{2.10}$$

A restrição 2.3 obriga a que os clientes sejam visitados exatamente uma vez, a restrição 2.4 estabelece que todos os veículos que saem do armazém têm de regressar ao mesmo. E a restrição 2.5 assegura o fluxo contínuo da rota. Na restrição 2.6 estabelece-se a precedência *linehaul-backhaul*, isto é, impede que sejam visitados clientes *backhaul* antes de serem visitados todos os clientes *linehaul*. A restrição 2.7 impede que o número de veículos que parte do armazém ultrapasse a quantidade de veículos disponíveis. As restrições 2.8 e 2.9 impedem que as capacidades máximas dos veículos sejam ultrapassadas para os clientes *linehaul* e *backhaul* respetivamente. Em 2.10 é definido o domínio da variável de decisão.

2.4 Métodos de solução para o *VRPB*

Os métodos de solução podem ser divididos em 3 grupos, nomeadamente: métodos exatos, métodos aproximados e métodos híbridos. Estes últimos são uma mistura dos dois primeiros em que numa primeira fase é utilizado um método exato como ponto de partida e numa segunda fase utilizado um método aproximado (Koç and Karaoglan, 2016).

O *VRP* é conhecido do ponto de vista da complexidade como um problema de otimização combinatória *NP-hard* visto não ser solúvel em tempo polinomial e ser um problema derivado dos problemas *TSP* e *Bin Packing Problem* que são também *NP-hard* (Caric and Gold, 2008; Lenstra and Kan, 1981). As variantes do *VRP* são também, como consequência, consideradas de complexidade *NP-hard*.

No trabalho realizado por Santos et al (2017), pode ser encontrada informação adicional a cerca dos assuntos abordados neste capítulo.

2.4.1 Métodos exatos

Estes métodos garantem a solução ótima quando aplicáveis. A grande desvantagem é que apenas são aplicáveis para problemas relativamente pequenos e na prática têm pouca aplicação devido à elevada dimensão dos problemas reais. Isto deve-se ao facto do grau de complexidade se situar em *NP-hard* como referido a cima em 2.4. Em Toth and Vigo (2002a) são revistos alguns algoritmos baseados em *branch-and-bound*. Em Gélinas et al (1995) é apresentada uma estratégia nova também baseada em *branch-and-bound*, que permite a obtenção da solução para problemas *VRPTW* com *backhauling* com uma dimensão até 100 vértices. Em relação ao *VRPB*, a primeira aplicação de um algoritmo exato foi em Yano et al (1987) com um *branch-and-bound*. Em Toth and Vigo (1997) é usado novamente o método *branch-and-bound* para resolver um *VRPB*, onde é utilizada a relaxação *Lagrangeana* no que diz respeito às restrições de capacidade.

2.4.2 Métodos aproximados

Como foi referido em 2.4.1, através de métodos exatos dificilmente se obtém a solução ótima para problemas de complexidade elevada em tempo aceitável. Acontece que com recurso a técnicas heurísticas este facto deixa de ser um problema, mas em contrapartida a solução não é garantidamente a solução ótima. Apesar disso estas técnicas conseguem apresentar soluções próximas da solução ótima e conseguem um bom *trade-off* entre qualidade da solução e tempo de resolução do problema. Este *trade-off* é justificado através de uma exploração relativamente limitada da área de pesquisa. Genericamente as heurísticas podem ser divididas em duas grandes classes, nomeadamente, heurísticas clássicas e metaheurísticas.

2.4.2.1 Heurísticas clássicas

As heurísticas clássicas têm esta designação devido à ausência de mecanismos que permitam que a função objetivo seja deteriorada a cada iteração, ao contrário das metaheurísticas que foram desenvolvidas mais recentemente (Laporte et al, 2000).

Segundo Toth and Vigo (2002b), entre as heurísticas clássicas podem-se distinguir 3 classes:

- Heurísticas de duas fases - como o próprio nome indica, divide o problema em duas fases. Uma primeira fase onde é feita a alocação dos clientes às rotas e uma segunda onde é determinada a ordem de visitas. Dentro desta classe podem ser distinguidos dois métodos: *cluster-first/route-second* e *route-first/cluster-second*.

- Heurísticas construtivas - o ponto de partida é uma solução vazia à qual vão sendo acrescentadas soluções admissíveis, isto é, são construídas gradualmente tendo sempre em conta a evolução dos custos. Incluindo o algoritmo *savings* e heurísticas de inserção.

- Heurísticas de melhoria - aperfeiçoa soluções existentes através de permutações de vértices entre rotas ou intra-rotas.

Para resolução de um *VRPB*, Goetschalckx and Jacobs-Blecha (1989) utiliza heurísticas que denomina *Spacefilling Curves Heuristics*. Também para resolução de um *VRPB*, Anily (1996) desenvolve uma heurística baseada numa já utilizada em trabalhos anteriores chamada *Modified Circular Regional Partitioning Procedure*. A heurística é utilizada primeiro para os clientes *linehaul* e depois para os clientes *backhaul*, criando regiões de *linehuals* e *backhauls*. Posteriormente seriam selecionadas as regiões a visitar. Toth and Vigo (1999), recorrem a uma heurística *cluster-first-route-second* com um método diferente de *clustering* para o mesmo problema, e que poderá ser utilizado no caso de custos assimétricos.

2.4.2.2 Metaheurísticas

As metaheurísticas, como foi referido a cima em 2.4.2.1, diferenciam-se das heurísticas clássicas ao possibilitarem que a função objetivo seja temporariamente deteriorada, com o propósito de evitar que as soluções encontradas para o problema fiquem "presas" em ótimos locais. Isto permite que sejam encontrados diferentes ótimos locais que sejam melhores que os encontrados anteriormente e, em teoria, uma maior aproximação ao ótimo global. Como consequência deste facto, em parte, estas provam ser bem sucedidas na resolução de *VRPs* complexos e de grande dimensão (Gendreau and Potvin, 2010). A partir de

- Os 6 casos apresentados em Gendreau et al (2002b) relacionados com *CVRP* (onde são aplicados *Simulated Annealing (SA), Deterministic Annealing (DA), Tabu Search (TS) e Genetic Algorithms (GA)*);
- A pesquisa feita em Parragh et al (2008a) onde mais casos são aplicados a outras variantes de *VRP*, incluindo *VRPB* (por exemplo um algoritmo baseado em *Large Neighborhood Search (LNS)*);
- Na pesquisa em Parragh et al (2008b) que cobre ainda mais variantes do *VRP*.

podemos estimar que o número de casos de sucesso tenderá a crescer quando a aplicação se alarga a todas as variantes do *VRP* (Gromicho et al, 2012).

Há várias formas de classificar e descrever os algoritmos das metaheurísticas e que dependem das suas características. Uma dessas formas tem como fundamento a comparação entre metaheurísticas baseadas em populações (*population-based*) e metaheurísticas baseadas em pesquisa individual, também designadas de métodos de trajetória (*trajetory methods*). As metaheurísticas *population-based* realizam processos de pesquisa que descrevem a evolução de um *set* de pontos no espaço de pesquisa, isto é, utilizam várias soluções em simultâneo para a otimização. Alguns exemplos de meteheurísticas *population-based* são: *GA*, *Ant Colony Optimization (ACO)* e *Particle Swarm Optimization (PSO)* (Beheshti and Shamsuddin, 2013). Já as metaheurísticas *trajectory methods* descrevem uma trajetória no espaço de pesquisa durante o processo e incluem metaheurísticas de pesquisa local tais como: *TS*, *Iterated Local Search (ILS)* e *Variable Neighborhood Search (VNS)* (Blum and Roli, 2003). Recentemente têm sido aplicados exemplos das metaheurísticas referidas em vários artigos devido ao seu sucesso em problemas complexos. Em Wassan (2007) é

proposto um algoritmo baseado em *TS* em conjunto com *Adaptative Memory Programming* para que a pesquisa seja conduzida para áreas não exploradas. Em Gajpal and Abad (2009) é utilizado um algoritmo baseado em *ACO* designado *Multi-Ant Colony Optimization System (MACS)*, conseguindo melhores resultados que os restantes algoritmos nos testes de *benchmark*. Em Cuervo et al (2014) é proposto um algoritmo *ILS* e que através de uma heurística *Oscillating Local Search (OLS)* permite que as soluções oscilem entre soluções admissíveis e não admissíveis. Em Wang et al (2010) é proposto uma metaheuristica baseada *GA* para a resolução de um *Vehicle Routing Problem with Time Windows (VRPTW)* que apresenta bons resultados, embora para instâncias de apenas 50 vértices. Em Küçükoğlu and Öztürk (2015) é apresentada uma proposta de metaheurística híbrida (*SA* e *TS*) para resolver um *VRPTW*, obtendo melhores resultados que os existentes à data em 34 das 45 instâncias testadas. Em Cordeau et al (2005) podem ser consultadas as metaheurísticas mais utilizadas até à data em aplicações de *VRP*, bem como as particularidades de cada uma delas.

2.5 Roteamento sob incerteza

Embora o *VRP* tenha vindo a ser muito estudado desde que estruturado em Dantzig and Ramser (1959), a grande parte dos problemas não considera incerteza. Normalmente, a abordagem seguida é determinística, isto é, é assumido que os dados do problema são conhecidos *à priori*, o que implica a possibilidade de uma solução considerada admissível deixar de o ser. Esta possibilidade deve-se ao facto de os dados considerados para construção da solução não serem os dados que se verificam em problemas reais. Assim, uma variação do valor dos dados da abordagem determinística, por mais pequena que seja, pode tornar a solução inadmissível quando a incerteza não é considerada. A incerteza pode surgir de diversas fontes e os principais elementos do *VRP* que são objeto de incerteza são, por exemplo: a procura dos clientes por um determinado produto, os tempos de viagem que podem sofrer atrasos das mais diversas formas, custos de transporte, entre outros.

Para que a incerteza seja tida em conta, as principais abordagens são a programação estocástica (*Stochastic Programming*) e a otimização robusta (*Robust Optimization*). A limitação encontrada à aplicação da programação estocástica é a necessidade de um conhecimento preciso das distribuições de probabilidade das variáveis aleatórias, restringindo assim o grupo de problemas aos quais a programação estocástica é adequada. Pelo contrário, a otimização robusta não tem a necessidade de fazer assunções quanto a distribuições de probabilidade e apenas faz a assunção de pertença das variáveis a um *set* determinístico de incerteza que poderá ser um poliedro (ver Bertsimas and Sim (2004)) ou uma elipsoide (ver Ben-Tal et al (2004)) (Maggioni et al, 2015).

Em programação estocástica, depois de determinadas as variáveis de decisão, é normalmente seguida uma de duas abordagens: estocástica com modelos de recurso (*stochastic with recourse models*) e de possibilidades restritas (*chance constrained*) (Ordóñez, 2010). A primeira onde é considerada uma função de recurso (*recourse function*) inserida no objetivo do problema e que contempla o custo médio de uma decisão tomada *à posteriori*. A segunda que requer a inclusão de

restrições probabilísticas no modelo e que impõe limites nas probabilidades associadas a eventos aleatórios específicos. De notar a relação existente entre *Stochastic VRP* (*SVRP*) e *dynamic VRP*, que são *VRPs* caracterizados pelo surgimento tardio de informação acerca dos dados do problema (Gendreau et al, 2002a). Podem ser consultados diferentes modelos e soluções de *dynamic VRP* em Bektas et al (2002), e em Gendreau et al (1996, 2002a) pode ser encontrada mais informação sobre a literatura em *SVRP*. Informações mais detalhadas em programação estocástica podem ser consultadas em Birge and Louveaux (2011).

A aplicação da otimização robusta é influenciada por 3 componentes muito importantes: a forma de modelação dos dados relacionados com a incerteza (dados contínuos ou discretos); a escolha do critério de otimização robusta adequado (ver referências em Solano-Charris et al (2014)); a escolha do método para geração de soluções robustas (Solano-Charris et al, 2014). Outros fatores importantes são: a formulação do problema, que deve ser feita por forma a que a mesma não seja mas difícil de resolver do que o seu modelo determinístico (Sungur et al, 2008); a escolha do *set* de incerteza adequado, o que determina se a tratabilidade do problema é preservada ou não (Han et al, 2014; Ben-Tal and Nemirovski, 1998). Este é um problema comum na otimização com incerteza devido à complexidade inerente (Sahinidis, 2004).

As soluções produzidas pela otimização robusta têm potencial para serem eficientes na realidade porque uma solução admissível pode ser admissível para todas as hipóteses do *set* de incerteza definido (Agra et al, 2013; Maggioni et al, 2015; Gounaris et al, 2013). Por outras palavras, uma solução admissível é admissível para todos os valores definidos no *set* de incerteza, valores estes que acomodam as possíveis variações que advém da incerteza. Além disso, as soluções não estão longe da solução determinística, enquanto que a superam na situação de pior caso. Informações detalhadas sobre otimização robusta tradicional podem ser encontradas em (Ben-Tal et al, 2009). Informações sobre diferentes conceitos da mesma podem ser encontradas em Ben-Tal et al (2004), Fischetti and Monaci (2009) e Liebchen et al (2009), onde são abordados respetivamente os modelos *adjustable, light* e *recovery*. Em Bertsimas et al (2011) podem ser consultadas aplicações da otimização robusta.

Um modelo híbrido em que é utilizada otimização robusta combinada com programação estocástica é usado em Erera et al (2010), no qual é elaborado um problema de duas fases em que na primeira são selecionadas rotas sem que seja tida em conta a incerteza da procura, e numa segunda fase, depois de observada a procura, uma decisão de recurso é tomada que reparam as rotas criadas anteriormente com retornos ao armazém quando é necessária a reposição de produtos.

Resumindo, podem ser encontradas algumas vantagens da otimização robusta em relação à estocástica: na maioria dos casos é mais fácil definir o *set* de incerteza do que estimar a distribuição de probabilidades; em determinadas condições, a dificuldade do problema não é aumentada de forma significativa (Bertsimas et al, 2004); mesmo no caso em que se tem apenas algum conhecimento prévio acerca da distribuição de probabilidades, o *set* de incerteza pode ser definido facilmente a partir do mesmo (Lee et al, 2012).

2.5.1 *Robust Vehicle Routing Problem* (*RVRP*)

O *RVRP* tem a vantagem de em vez de considerar probabilidades para as incertezas (como no caso da otimização estocástica) considera um *set* limitado de incerteza, o que lhe permite obter uma solução resiliente às variações nos dados consideradas no mesmo.

O *RVRP* tem o potencial apresentado pela otimização robusta, mas ainda assim são poucos os estudos sobre a aplicação da mesma ao *VRP*. Esta falta de estudo talvez se deva ao facto das abordagens determinísticas necessitarem de ser repensadas para terem em conta incerteza (Lee et al, 2012). Isto levou a que os artigos acerca do *RVRP* fossem bastante recentes na literatura. A título de exemplo, o primeiro artigo da literatura em *Robust CVRP* (*RCVRP*) é apresentado em 2008, por Sungur et al (2008) (Agra et al, 2013), no qual é feito um estudo acerca das condições necessárias para que o *RCVRP* possa ser resolvido através de métodos semelhantes aos usados no *CVRP* determinístico (Agra et al, 2013).

A maioria dos artigos em *RVRP* têm como foco a incerteza na procura de produto por parte dos clientes ou a incerteza tempos de viagem realizados pelos veículos (*service times*).

Sungur et al (2008), em 2008, estuda os *sets* de incerteza segundo os quais a formulação *Miller-Tucker-Zemlin (MTZ)* é resolvida através da otimização robusta, sem que seja acrescentada dificuldade adicional à resolução em relação à formulação determinística. É demonstrado também que a solução robusta evita procuras não satisfeitas à custa de um ligeiro aumento de custos. Para a resolução é utilizado um *Branch-and-Cut* e os resultados dos testes demonstram que a solução robusta é favorável em relação à determinística, quando a folga de capacidade dos diferentes veículos é distribuída por toda a frota, permitindo assim a partilha das mesmas em caso de incerteza. Em comparação com os modelos estocásticos, para pequenas instâncias, o modelo robusto satisfaz todas as procuras com o custo igual ou inferior.

Moghadam et al (2010), em 2010, apresenta um problema *VRP* básico onde os clientes são fornecidos através de um armazém central e onde o foco de incerteza é na procura. É utilizada uma formulação baseada na formulação *MTZ* e resolvida pelo método também utilizado em Ben-Tal and Nemirovski (1998, 1999). Os resultados desta abordagem mostram, novamente, que é evitada a procura não satisfeita à custa de um aumento no custo da solução.

Manisri et al (2011), em 2011, propõe um algoritmo de duas fases desenvolvido para tirar proveito da *Modified Push-Forward Insertion Heuristic (MPFIH)*, da λ*-interchange Local Search Descent method (*λ*-LSD)* e da metaheurística TS. O algoritmo é usado para escolher a melhor rota baseada na abordagem robusta. Na primeira fase é aplicada a *MPFIH* para a criação das rotas em que são inseridos sucessivamente clientes, tendo sempre em conta a restrição de capacidade. Na segunda fase é utilizado o método λ*-LSD* em que é feita uma troca de clientes entre duas rotas, e depois a metaheurística TS é utilizada para criar diversificação ao fazer trocas de clientes dentro das rotas e evitando que o algoritmo fique preso em ótimos locais.

Lee et al (2012), em 2012, aborda um *VRP* com objetivo de respeitar *deadlines* (*VRPD* - caso especial de *VRPTW*) e a capacidade dos veículos (*CVRP*). Nesta abordagem o foco de incerteza está nos tempos de viagem e na procura dos clientes, onde são considerados *sets* diferentes para

cada um deles. As incertezas de tempo de viagem e de procura são modelados como *Robust Shortest Path Problem with Resource Constraints (RSPPRC)* e resolvidos com um algoritmo *Dynamic Programming*. Os resultados mostram que em vários casos os ganhos produzidos pela abordagem robusta são razoáveis, com uma pequena penalização nos valores da função objetivo. Apesar disso o algoritmo proposto não pode ser aplicado diretamente a um *RVRPTW*, onde os clientes têm os tempos de início para o princípio do serviço (carga ou descarga) bem como de *deadlines*. No mesmo ano, Souyris et al (2013) resolve um problema em empresas de serviços de manutenção e reparação através de um *VRP* em que a cada serviço estão associados tempos de serviço e *deadlines*. O problema é então formulado como um *VRP with Stochastic Service Times (VRPSST)* robusto para encontrar resposta para o foco de incerteza que se verifica nos tempos de serviço. É considerado que esta incerteza pertence a um *set* convexo limitado e são apresentadas duas versões de *sets* que tornam o problema tratável. Os resultados demonstram ser uma solução capaz de resolver problemas de tamanho relevante para os situações reais, visto ter sido encontrada solução para um problema onde são considerados 41 clientes e 15 técnicos.

Han et al (2014), em 2013, considera um *VRP* com incerteza em tempos de viagem onde são conhecidas de antemão informações acerca dos tempos de viagem futuros. É utilizada a otimização robusta e resolvida através de um algoritmo *Branch-and-Cut*. Também em 2013, Gounaris et al (2013) estuda o *Robust CVRP (RCVRP)* com foco de incerteza na procura, com o objetivo de obter soluções viáveis para todas as possibilidades de procura consideradas. São apresentadas três formulações do problema: *Vehicle Flow formulation, Commodity Flow formulation e Miller-Tucker-Zemlin formulation*. Para resolução das formulações propostas é usado um algoritmo *Branch-and-Cut* em que os cortes são baseados nos *Rounded Capacity Inequalities Cuts* apresentados em Toth and Vigo (2002b), e que podem acelerar a solução. Ainda em 2013, Agra et al (2013) apresenta duas formulações do problema, uma que é baseada em *Resource Inequalities* e outra em *Path Inequalities*. A primeira em que são usadas restrições de conservação de fluxo de cada veículo, isto é, dos recursos, e a segunda em que a questão das *time windows* não é tida em conta explicitamente mas sim implicitamente. É criado um subconjunto de rotas do conjunto de rotas total, em que estão contidas as rotas nas quais não é possível garantir que as *time windows* sejam respeitadas. Para a formulação baseada em *Resource Inequalities* são usadas técnicas para redução dos pontos extremos do poliedro de incerteza e criado um algoritmo de *Column and Row Generation* para solução do problema. Este algoritmo procura selecionar as restrições necessárias à medida que a solução do problema avança para que não seja desperdiçado esforço em cenários desnecessários. Cada conjunto de restrições corresponde a um cenário. Para a formulação baseada em *Path Inequalities* é utilizado um algoritmo baseado em *Branch-and-Cut* para resolução do problema. Esta abordagem revelou ser tão fácil de resolver quanto o seu equivalente determinístico. No mesmo período, Toklu et al (2013) considera um *RCVRP* com foco de incerteza em tempos de viagem, e para a solução do mesmo é utilizado um algoritmo baseado em *ACO*. Os resultados dos testes para instâncias de 100 clientes demonstram que é, de facto, gerado um conjunto de soluções promissoras com diferentes níveis de conservadorismo.

Cao et al (2014), em 2014, resolve um problema onde a incerteza recai sobre a procura de produto por parte dos clientes, e no qual o regresso ao armazém central após a realização das entregas não é obrigatório *(Open VRP - OVRP)*. O autor propõe quatro estratégias robustas para lidar com a incerteza e um algoritmo de evolução diferencial melhorado para a solução do modelo robusto. Todas as estratégias evitam deixar procuras por satisfazer, o que implica um ligeiro custo extra. No mesmo ano, Solano-Charris et al (2014) publica um artigo em que é abordado um *VRP* básico. O foco de incerteza são os tempos de viagem e é utilizada uma formulação *Mixed Integer Linear Programming (MILP)*. Para a sua solução é proposto um GA onde é criada uma população inicial admissível, a primeira metade da mesma é construída através de uma adaptação da heurística *best insertion* e a segunda metade é gerada aleatoriamente. A partir desta população inicial é criada uma população intermédia aplicando o algoritmo *Order Crossover (OX)*, que consiste na seleção de uma sequência de clientes de uma rota e consequente inserção numa rota diferente. São também consideras mutações intra-rotas e entre rotas. Para evitar que haja uma convergência prematura na pesquisa é também usado um procedimento de renovação. Os resultados dos testes para instâncias de baixa e média escala (até 20 e até 100 vértices) mostram que o *GA* proposto para pequenas instâncias encontra 7 ótimos em 10 instâncias em menos de 8 segundos. Para instâncias médias o algoritmo demorou no máximo 12 minutos a produzir resultados.

Em 2015, Duan et al (2015) propõe dois modelos matemáticos que são desenvolvidos para resolver o *Stochastic Time-Dependent VRP (STD-VRP)*, em que um deles é baseado em otimização robusta com critério *Min-Max*. É também um problema em que são consideradas *hard time windows*, isto é, *time windows* que não podem ser relaxadas (violadas), e para resolução do mesmo é utilizado um algoritmo baseado em *ACO*. Com a otimização robusta, a solução tem um custo superior, mas são evitadas novas entregas por chegadas fora da janela definida pelas *time windows*. No mesmo período, Adulyasak and Jaillet (2015) considera um *(VRPD)* em que o foco de incerteza são os tempos de viagem (caracterizados por uma distribuição de probabilidades). É feita a comparação entre as formulações robusta e estocástica. São analisados os aspetos computacionais das soluções propostas e para solução das mesmas é desenvolvido um algoritmo baseado no *Branch-and-Cut*. Através dos testes realizados foi possível verificar que a eficiência computacional sofreu, em geral, melhorias significativas em relação às abordagens na literatura.

De seguida, na tabela (2.1) é apresentado um breve resumo do mencionado até agora.

Tabela 2.1: Resumo de informação contida nos artigos revistos sobre *RVRP*

Autores	Ano	Variante	Método de resolução			Incerteza
			Exato	Heurística	Metaheurística	
Sungur et al (2008)	2008	CVRP	Branch-and-Cut			Procura
Moghadam et al (2010)	2010	CVRP	Branch-and-Bound			Procura
Manisri et al (2011)	2011	VRPTW		Modified Push-Forward Insertion (MPFIH), Local Search Descent (LSD)	Tabu Search	Travel Times
Lee et al (2012)	2012	CVRP	Dynamic Programming			Procura e Travel Times
Souyris et al (2013)	2012	VRPSST	Branch-and-Price			Service Times
Toklu et al (2013)	2013	CVRP			Ant Colony Optimization (ACO)	Travel Times
Han et al (2014)	2013	VRP	Branch-and-Cut			Procura
Gounaris et al (2013)	2013	CVRP	Branch-and-Cut			Procura
Agra et al (2013)	2013	VRPTW	Column and Row generation, Branch-and-Cut			Travel Times
Cao et al (2014)	2014	OVRP			Differential Evolution Algorithm	Procura
Solano-Charris et al (2014)	2014	CVRP			Genetic Algorithm (GA)	Travel Times
Duan et al (2015)	2015	VRPTW			Ant Colony Optimization	Travel Times
Adulyasak and Jaillet (2015)	2015	CVRP	Branch-and-Cut			Travel Times
Este trabalho	**2017**	**VRPB**			**Adaptative Large Neighborhood Search**	**Prémios dos fornecedores**

Nesta dissertação irá ser abordado o problema *VRPB*, resolvendo-o através da metaheurística *ALNS*. Esta escolha tem por base a comparação entre métodos utilizados em vários artigos relacionados com o *VRPB*, apresentada em Santos et al (2017). A principal novidade para a literatura é a inclusão da otimização robusta para que se consiga lidar com a incerteza inerente a um dos parâmetros do problema, bem como a associação de prémios aos fornecedores que traduzem um benefício na visita aos mesmos. A otimização robusta é escassa na literatura do *VRP* e quase nula no *VRPB*, traduzindo-se numa grande mais valia.

Capítulo 3

Descrição do problema

Neste capítulo será descrito o problema. Em primeiro lugar será explicado o contexto em que o mesmo se insere, e de seguida será feita a descrição do problema em si. Serão abordadas as suas especificidades e explicadas as suas características. Serão também enumerados e caracterizados os principais componentes constituintes do problema e sem os quais este não existiria. Por fim, depois de explicado de uma forma teórica este será traduzido para a sua formulação matemática.

3.1 Contexto

O problema em estudo na dissertação procura a sustentabilidade de uma cadeia de abastecimento da indústria da madeira através da integração da logística de *inbound* e *outbound* de uma empresa. É inspirado numa empresa designada por SONAE Indústria que atua na indústria de produtos derivados da madeira. A SONAE Indústria é uma multinacional presente em 12 países distribuídos por 3 continentes e líder na Europa. Na Península Ibérica conta com várias fábricas, vários clientes (normalmente no setor secundário) e diversos fornecedores através dos quais é obtida a madeira (matéria-prima). Para a realização de todo o transporte necessário ao seu funcionamento, o serviço é contratado a terceiros.

3.2 Descrição

Como referido em 3.1 a SONAE Indústria dispõe de fábricas, vários clientes e fornecedores. Este conjunto de fábricas produz milhares de produtos, o conjunto de clientes constitui uma procura que terá que ser satisfeita (operações de *linehaul*) e o conjunto de fornecedores tem uma determinada disponibilidade de matéria-prima (operações de *backhaul*). O problema consiste em definir um conjunto de rotas, com início nas fábricas, que garantam: a passagem de veículos pelos clientes, satisfazendo as suas procuras; a passagem em fornecedores, para recolha da madeira necessária ao funcionamento das fábricas. As possibilidades de realização das rotas são as mais diversas, mas o objetivo é que, além de serem realizadas por forma a minimizar a distância percorrida pelos veículos nas operações de *linehaul*, também sejam realizadas por forma a tornar o

uso da frota eficiente, e para tal realizando as operações de *backhaul*. Adicionalmente, tem de se ter em consideração o facto de os veículos estarem limitados a uma capacidade máxima de transporte e, portanto, nem as operações de *linehaul* nem as operações de *backhaul* poderão exceder a mesma, separadamente. De notar que a visita ao fornecedor não é obrigatória e no caso da indústria em que este problema se insere, a não ser que a capacidade máxima dos veículos seja superior à disponibilidade de matéria-prima no fornecedor, bastará ao veículo passar por um fornecedor para encher completamente o veículo. De notar também que os mesmos *backhauls* poderão ser visitados por mais que uma rota.

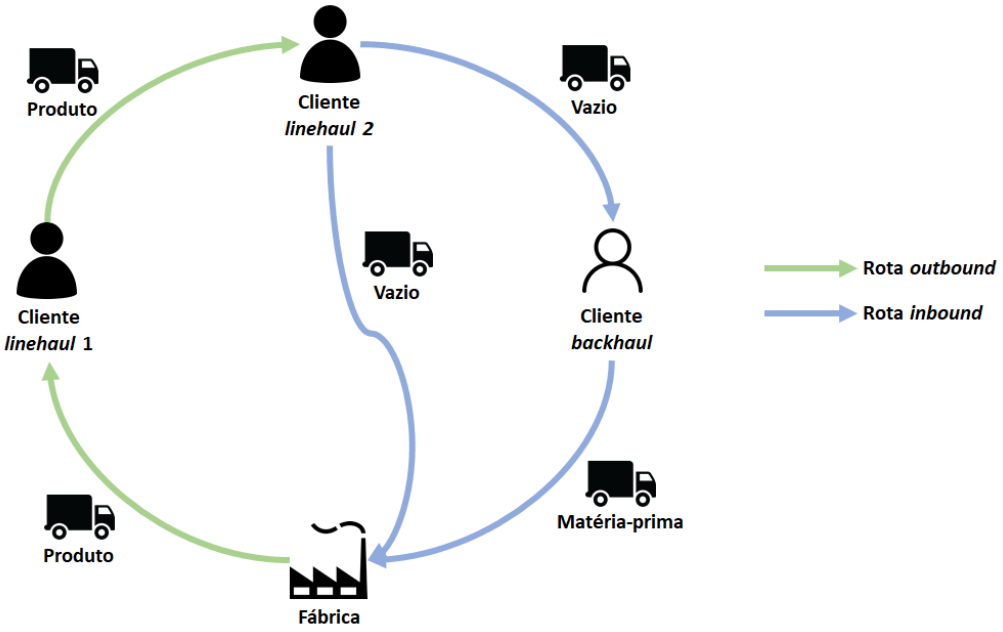

Figura 3.1: Exemplificação do problema em questão

Atualmente muitas empresas dispõem de trabalhadores exclusivamente dedicados ao planeamento, isto é, responsáveis pelo planeamento do transporte em cada fábrica, e que definem as rotas bem como o número de veículos necessários para a realização das mesmas. O serviço de transporte planeado é então contratado a empresas de transporte especializadas, com uma frota de veículos adequados ao transporte de madeira (matéria-prima) e produtos derivados da mesma. Isto implica custos adicionais com o pagamento do serviço. Em cada semana existe uma lista de clientes a serem visitados e a respetiva procura. Assim, a distribuição de clientes é feita semanalmente, tendo cada dia um conjunto de clientes atribuídos, e para que este conjunto de clientes seja satisfeito é realizado um planeamento diário. Este processo gera então dois *outputs*, um semanal e um diário. O primeiro onde constam os clientes a serem visitados por dia (as encomendas diárias a serem satisfeitas) e a respetiva fábrica que irá disponibilizar os produtos. O segundo onde constam os produtos que terão de estar disponíveis, visto que neste setor existe o princípio de que tudo o que é produzido em fábrica é imediatamente expedito, consequência da baixa capacidade

de armazenamento de produto final (*stock*). De notar que o planeamento diário é despoletado pelo planeamento semanal.

Os planeamentos referidos geram um conjunto de rotas que, individualmente, têm início numa fábrica. Este conjunto de rotas tem de conter todos os clientes presentes no planeamento semanal e cada rota pode conter um cliente, ou um conjunto de clientes. Para que a rota seja possível, a fábrica onde as rotas têm início, tem de ter disponibilidade de produto suficiente para satisfazer todas as encomendas dos clientes pelos quais a rota está destinada a passar. Em cada rota os clientes são visitados apenas uma vez e, também em cada rota, pode estar presente um fornecedor no regresso à fábrica. Desta forma é recolhida a matéria-prima essencial ao funcionamento da fábrica à qual o veículo regressa, isto é, a fábrica que finaliza a rota. A razão pela qual no regresso é apenas visitado um fornecedor reside no facto de, habitualmente, no setor em questão o veículo ser completamente carregado num fornecedor apenas. Isto deve-se ao tipo de matéria-prima em questão que é a madeira. Tanto o número de clientes como o número de fornecedores visitados tem um número máximo, que é limitado indiretamente pela capacidade máxima dos veículos da frota.

De notar que, inerente a esta viagem de regresso, se tem a incerteza no que respeita à passagem no fornecedor. Isto é, a inclusão de um fornecedor numa rota deve ser feita sempre que possível desde que não acarrete despesas desnecessárias, mas isto pode significar: não visitar nenhum, visitar fornecedores apenas em algumas das rotas, ou visitar em todas as rotas de regresso.

Tendo em conta o problema descrito e o referido anteriormente, podemos concluir que a correspondência entre o problema em questão e o *VRPB* é quase total, faltando apenas ao *VRPB* a capacidade de lidar com a incerteza consequente da inclusão de fornecedores nas rotas (tabela 3.1).

Tabela 3.1: Relação entre as características do problema e o VRPB

Características do problema	*VRPB*
Planeamento diário das rotas para os clientes atribuídos semanalmente	✓
Rotas iniciam e terminam numa fábrica	✓
Rotas de *Outbound* podem incluir mais do que um cliente	✓
Cada cliente é visitado uma vez por dia	✓
As encomendas dos clientes devem ser todas satisfeitas	✓
Depois de visitados todos os clientes, podem ser visitados fornecedores	✓
O número de clientes e fornecedores visitados é limitado pela capacidade máxima do camião	✓
Os fornecedores têm um limite máximo de matéria-prima	✓
Frota de veículos homogénea	✓
Incerteza na passagem pelo fornecedor	✗

Para lidar com a incerteza referida e como o *VRPB* simples não apresenta essa capacidade, existe a possibilidade de extender o problema. Esta extensão pode ser obtida por mais do que uma

maneira (ver 2.5), mas pelo facto de neste caso não serem conhecidas as distribuições de probabilidade do parâmetro sujeito a incerteza (consequência das variações na qualidade da matéria-prima), o caminho mais óbvio a seguir é a aplicação da otimização robusta ao problema, passando este a ser um *Robust Vehicle Routing Problem with Backhauls (RVRPB)*. Assim, garantindo uma qualquer solução admissível, garantimos também que essa solução é admissível para o pior caso *set* definido para o parâmetro sujeito a incerteza.

3.3 Formulações matemáticas

Neste subcapítulo irão ser abordadas as formulações matemáticas para os métodos exatos, primeiro do *VRPB* com a introdução do prémio, que corresponde ao problema específico da dissertação. E de seguida as adaptações para a formulação final *RVRPB*.

3.3.1 VRPB

Para o *VRPB* irá ser utilizada a formulação *Commodity Flow (CFF)* que é apresentada nos subcapítulos seguintes.

3.3.1.1 Notação

Seja $G = (V,A)$, onde:

- V é o *set* de vértices $V = \{0\} \cup L \cup B$ e os *subsets* $\{0\}$, L e B são, respetivamente, o armazém e os *subsets* referentes aos clientes *linehaul* e *backhaul*;
- A é o *set* de arcos que são definidos entre dois vértices do *set* V referido.

A cada vértice pertencente a L e B estão associadas, respetivamente, uma quantidade não negativa de procura de produto e uma quantidade de produto a ser recolhido $q_i > 0$, e ao armazém uma quantidade fictícia $q_0 = 0$. O conjunto de vértices com a excepção do armazém é designado V_0. Considera-se que o vértice correspondente ao armazém (V_0) contém o *set* de veículos homogéneos K, com capacidade C, de onde todos partem e onde todos regressam depois de visitados todos os clientes. Seja também: p_j o prémio associado a cada fornecedor; d_j a procura no caso dos clientes e disponibilidade no caso dos fornecedores; c_{ij} o custo associado entre i e j e onde $c_{ij} = c_{ji}$ para cada $i, j \in V$ tal que $i \neq j$. Os custos são considerados simétricos.

3.3.1.2 Variáveis de decisão

A variável de decisão deste problema é binária e define se um veículo k faz a viagem entre os nós i e j ou não.

$$x_{ij}^{k} \begin{cases} 1, \ \textit{se veículo } k \textit{ faz a viagem do cliente } i \textit{ para o cliente } j \\ 0, \ \textit{caso contrário} \end{cases} \tag{3.1}$$

y_{ij}, a carga do veículo entre o cliente i e j. (3.2)

3.3.1.3 Função objetivo

Este problema tem como objetivo minimizar o custo total do conjunto de rotas incluindo os prémios obtidos por passagem em fornecedores, e que neste caso é obtido da seguinte maneira:

$$Min \sum_{k \in K} \sum_{\substack{i,j \in V \\ i \neq j}} c_{ij} x_{ij}^k - \sum_{k \in K} \sum_{\substack{i \in L \\ j \in B}} p_j x_{ij}^k \tag{3.3}$$

3.3.1.4 Restrições

As restrições que definem o *VRPB* são as seguintes:

$$\sum_{k \in K} \sum_{\substack{i \in V \\ i \neq j}} x_{ij}^k = 1 \quad \forall j \in L \tag{3.4}$$

$$\sum_{k \in K} \sum_{\substack{j \in V \\ i \neq j}} x_{ij}^k = 1 \quad \forall i \in L \tag{3.5}$$

$$\sum_{j \in L} x_{0j}^k = \sum_{i \in V_0} x_{i0}^k \quad \forall k \in K \tag{3.6}$$

$$\sum_{i \in V} x_{ij}^k = \sum_{i \in V} x_{ji}^k \quad \forall j \in L, \forall k \in K \tag{3.7}$$

$$\sum_{i \in V_0} x_{ij}^k = \sum_{i \in B} x_{ji}^k + x_{j0}^k \quad \forall j \in B, \forall k \in K \tag{3.8}$$

$$\sum_{i \in V} y_{ij} = \sum_{i \in v} y_{ji} + d_j \quad \forall j \in L \tag{3.9}$$

$$y_{ij} = 0 \quad \forall i \in L, \forall j \in B \tag{3.10}$$

$$y_{i0} = 0 \quad \forall i \in L \tag{3.11}$$

$$\sum_{j \in L} y_{0j} = \sum_{j \in L} d_j \tag{3.12}$$

$$y_{i0} = \sum_{k \in K} x_{i0}^k d_i \quad \forall i \in B \tag{3.13}$$

$$y_{ij} = \sum_{k \in K} x_{ij}^k C \quad \forall i, j \in V \tag{3.14}$$

$$\sum_{i \in B} \sum_{j \in L} x_{ij}^k = 0 \quad \forall k \in K \tag{3.15}$$

$$x_{0j}^k = 0 \quad \forall j \in B, \forall k \in K \tag{3.16}$$

$$[c_{i0} - (c_{ij} - p_j + c_{j0})] x_{ij}^k \geq 0 \quad \forall i \in L, \forall j \in B, \forall k \in K \tag{3.17}$$

$$x_{ij}^k \in \{0,1\} \tag{3.18}$$

$$y_{ij}^k \geq 0 \tag{3.19}$$

As restrições 3.4 e 3.5 asseguram que cada cliente é visitado apenas uma vez e por apenas um veículo. As restrições 3.6 a 3.8 asseguram que cada rota começa e termina num armazém. As restrições 3.9 a 3.13 dizem respeito às restrições de capacidade e procura. A restrição 3.14 assegura que a capacidade do veículo não é excedida e a restrição 3.15 estabelece a precedência, onde todos os clientes têm de ser visitados antes de se poder visitar um fornecedor. A restrição 3.16 elimina a possibilidade de existência de rotas com apenas fornecedores e a restrição 3.17 é uma desigualdade válida introduzida com o objetivo de fortalecer a formulação do problema. As duas últimas equações definem os domínios das variáveis de decisão.

3.3.2 RVRPB

A formulação final terá em conta a incerteza, e para tal terá de ser introduzida a robustez (Bertsimas and Sim, 2004). Para isso, é necessário ter em conta a modelação das ocorrências de pior caso num poliedro:

$$U = \left\{ D \in R^{|B|}, p_j \in [pm_j - \dot{p}_j, pm_j], \sum_{j \in B} |pm_j - p_j|/\dot{p}_j \leq \Gamma \quad \forall j \in B \right\} \tag{3.20}$$

Onde pm_j é o prémio médio do fornecedor j e $\dot{p}_j$ é o valor de desvio. De notar que $\dot{p}_j \leq pm_j$, isto é, o valor de desvio é menor ou igual que o valor médio do prémio . Pode então ser definido $\varepsilon_j = (pm_j - p_j)/\dot{p}_j$ onde $\varepsilon \in [0,1]$ tal que:

$$p_j = pm_j - \dot{p}_j \varepsilon_j \tag{3.21}$$

E nesse caso teríamos a função objetivo seguinte:

$$Min \sum_{k \in K} \sum_{\substack{i,j \in V \\ i \neq j}} c_{ij} x_{ij}^k - Min_D \left(\sum_{k \in K} \sum_{\substack{i \in L \\ j \in B}} pm_j x_{ij}^k - x_{ij}^k \dot{p}_j \varepsilon_j \right) \tag{3.22}$$

Como esta função objetivo é não-linear, através da teoria da dualidade podemos torná-la linear formando os problemas primal e dual:

Problema primal

$$Max \sum_{k \in K} \sum_{i \in V} \sum j \in B x_{ij}^k \dot{p}_j \varepsilon_j \tag{3.23}$$

$$\sum_{j \in B} \varepsilon_j \leq \Gamma \tag{3.24}$$

$$0 \leq \varepsilon_j \leq 1 \tag{3.25}$$

Problema dual

$$Min \ \lambda \Gamma + \sum_{j \in B} \mu_j \tag{3.26}$$

$$\lambda + \mu_j \geq \sum_{k \in K} \sum_{i \in V} x_{ij}^k \dot{p}_j \quad \forall j \in B \tag{3.27}$$

$$\lambda, \mu \geq 0 \quad \forall j \in B \tag{3.28}$$

Isto implica alterações tanto na função objetivo 3.3 como na restrição 3.17, bem como a adição de novas restrições como consequência da alteração da função objetivo. Essas alterações serão ilustradas nos subcapítulos seguintes.

3.3.2.1 Função objetivo

Com as alterações, a função objetivo passa a:

$$Min \sum_{k \in K} \sum_{\substack{i,j \in V \\ i \neq j}} c_{ij} x_{ij}^k + \beta \tag{3.29}$$

3.3.2.2 Restrições

A função objetivo, além das restrições ilustradas em 3.3.1.4, sofre alteração, como referido, na restrição 3.17 e são acrescentadas duas equações. Fica então sujeita a:

$$\beta \geq -\sum_{k \in K} \sum_{i \in L} \sum_{j \in B} p_j x_j^k + \lambda \Gamma + \sum_{j \in B} \mu_j \tag{3.30}$$

$$\lambda + \mu_j \geq \sum_{k \in K} \sum_{i \in V} x_j^k \dot{p}_j \quad \forall j \in B \tag{3.31}$$

$$\lambda, \mu_j \geq 0 \quad \forall j \in B \tag{3.32}$$

A função objetivo anterior (3.3.1.3) é agora subdividida em duas (3.29 e 3.30). Isto acontece porque é de boa prática escolher um parâmetro determinístico para incluir na função objetivo, neste caso β, que atue como parâmetro de incerteza e para tal, este terá de ser definido no conjunto de restrições. A restrição 3.31 é necessária para incluir robustez como consequência do problema dual referido anteriormente. A restrição 3.32 define o domínio dos novos parâmetros utilizados.

Capítulo 4

Desenvolvimento e implementação

Neste capítulo irá ser explicado o algoritmo que irá ser desenvolvido e a sua implementação. Tudo isto terá como base a formulação matemática referida e ilustrada no capítulo anterior. Em primeiro lugar irá ser abordada a teoria acerca do algoritmo a usar e de seguida serão explicadas, em diferentes fases, os passos dados para a aplicação final do algoritmo ao problema da dissertação.

Numa primeira fase é descrito o desenvolvimento do algoritmo base e que é aplicado ao *CVRP*. Numa segunda fase são destacadas as alterações realizadas com o objetivo de tornar o algoritmo aplicável a um *VRPB*. Por fim, são descritas as novas alterações feitas para a finalizaçao do algoritmo, passando este a ser aplicável ao problema, o *RVRPB*.

4.1 *Adaptative Large Neighborhood Search (ALNS)*

A metaheurística *Adaptative Large Neighborhood Search (ALNS)* foi proposta por Ropke and Pisinger (2006) e é uma extensão da metaheurística *Large Neighborhood Search (LNS)* em que, na mesma pesquisa, é utilizado um conjunto de operadores tanto de destruição como de reparação. No *ALNS* são introduzidos pesos que são associados aos pares destrutor/reparador e que controlam a escolha dos pares a usar em cada iteração do método. Estes pesos vão sendo ajustados dinamicamente à medida que o *ALNS* progride, o que garante uma capacidade de adaptação à instância em processamento e ao estado da pesquisa.

O algoritmo *LNS* foi inicialmente proposto por Shaw (1998), e tem como ideia principal permitir que a heurística percorra o espaço de solução facilmente ao abranger uma vizinhança alargada, mesmo nos casos em que a instância tem bastantes restrições. Isto não acontece quando a vizinhança abrangida pela heurística é reduzida. A maioria dos algoritmos de pesquisa definem esta vizinhança explicitamente, enquanto que o *LNS* define a mesma implicitamente através dos operadores de destruição e reparação. O operador destrutor tem, como o próprio nome indica, a função de destruir parte da solução, isto é, remover alguns dos vértices que estão presentes na solução. Já o operador reparador tem a função de inserir de novo na solução os vértices removidos pelo operador destrutor, em posições diferentes gerando assim uma nova solução. De notar que

com a utilização dos operadores referidos a solução está constantemente a oscilar entre o domínio admissível e não admissível. Isto acontece porque ao aplicar o operador destrutor a solução deixa de ser uma solução admissível, e ao aplicar o operador reparador a solução volta a ser trazida para o domínio admissível.

Depois de aplicados os pares destrutor/reparador é então avaliada a nova solução alcançada através de um critério de aceitação. Este critério serve para guiar a pesquisa no espaço de soluções e pode ter diversas formas. Pode ser um simples critério em que apenas são aceites soluções que sejam melhores que a melhor solução encontrada até ao momento, o que tem como limitação a possibilidade do *LNS* ficar aprisionado num ótimo local. Ou podem basear-se noutros critérios como por exemplo o critério utilizado na heurística *Simulated Annealing*, em que a nova solução é sempre aceite se for melhor que a anterior e aceite se for pior com uma determinada probabilidade, definida por uma função que depende das soluções em questão e de um parâmetro T designado de temperatura. T tem um valor inicial positivo e decresce à medida que o número de iterações avança. Este critério tem como objetivo permitir que a solução se deteriore, ou seja, sejam aceites soluções piores, no início do *LNS* e que convirja próximo do final do mesmo. Isto é benéfico pois permite que o *LNS* não fique confinado a um ótimo local.

Uma fase do *LNS* com bastante importância é a da aplicação do operador destrutor, mais concretamente a escolha do grau de destruição, isto é, a quantidade de vértices a remover da solução. Se o grau de destruição for demasiado elevado a vizinhança considerada na pesquisa será grande demais para que seja possível obter convergência. No caso de o grau de destruição ser demasiado baixo, a heurística irá ter dificuldade em pesquisar no espaço de soluções admissíveis perdendo-se assim a vantagem em utilizar o *LNS*. Portanto, esta escolha tem de ser feita cuidadosamente. Algumas formas de fazer esta escolha são: aumento gradual do grau de destruição; escolha aleatória do grau de destruição em cada iteração, dentro de uma determinada gama de valores pré-definidos.

Como o *ALNS* possui capacidade de adaptação, existe a possibilidade de inclusão de operadores destrutores e reparadores que são adequados apenas em casos específicos. Isto acontece porque recorrendo ao auxílio dos pesos que estarão associados a cada par de operadores, o *ALNS* irá encarregar-se de escolher tais operadores para os casos em que são adequados, escolhendo os restantes pares para os casos que sobram. Isto, combinado com a capacidade de evasão de ótimos locais confere robustez a estas heurísticas (*LNS* e *ALNS*).

Os operadores podem também ser vistos como ferramentas para diversificação ou intensificação. Por exemplo podem ser removidos vértices aleatórios criando assim diversificação, ou os vértices podem ser removidos criteriosamente, como por exemplo remover vértices que estão relacionados de uma determinada forma, obrigando assim a uma intensificação da pesquisa. A diversificação é então obtida através da mudança de vizinhanças, que por sua vez é obtida com a utilização de pares destrutor/reparador diferentes, e evita que os mesmos estejam ciclicamente a aplicar as mesmas alterações à solução. Por vezes é necessária a introdução de ruído/aleatoriedade nos operadores para se obter a diversificação desejada.

O facto de cada um destes pares de operadores de destruição/reparação gerar uma vizinhança única faz com que o espaço de pesquisa seja maior do que noutro tipo de heurísticas, e que confira

ao *LNS* e ao *ALNS* a facilidade de pesquisa no espaço de solução, característica esta que era o objetivo na altura em que o *LNS* foi proposto. A causa da dimensão referida pode ser melhor compreendida com a ilustração da Figura 4.1.

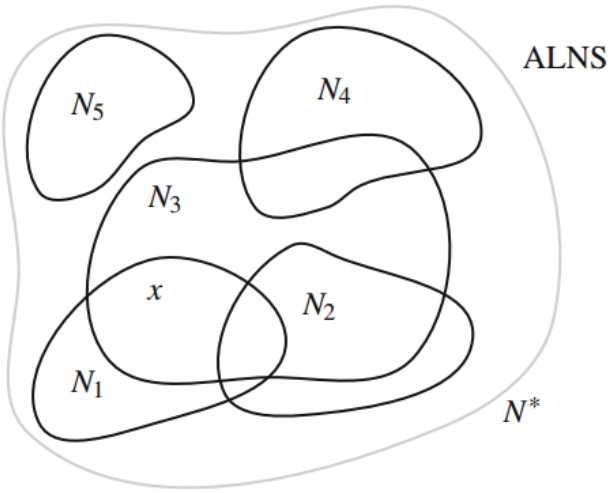

Figura 4.1: Imagem ilustrativa das vizinhanças geradas por diferentes pares de operadores destruição/construção utilizados pelo *ALNS* (Gendreau and Potvin, 2010)

Para uma apresentação mais estruturada do funcionamento do *LNS* é apresentado um pseudocódigo para a heurística no Algoritmo 1, baseado no pseudocódigo apresentado em Gendreau and Potvin (2010).

Algorithm 1

```
1:  procedure LNS(x ∈ {soluções admissíveis})
2:      x^b = x;
3:      while critério de paragem não é alcançado do
4:          x^t = r(d(x));
5:          if aceite(x^t, x) then
6:              x = x^t;
7:          end if
8:          if c(x^t) < c(x^b) then
9:              x^b = x^t;
10:         end if
11:     end while
12:     return x^b
13: end procedure
```

O *ALNS* é uma extensão do LNS, sendo que as semelhanças são grandes como se pode comprovar no pseudocódigo da heurística no Algoritmo 2, também baseado no pseudocódigo apresentado

em Gendreau and Potvin (2010).

Algorithm 2

1: **procedure** ALNS($x \in \{soluções\ admissíveis\}$)
2: $x^b = x$; $\rho^- = (1,...,1)$; $\rho^+ = (1,...,1)$;
3: **while** critério de paragem não é alcançado **do**
4: Selecionar operadores destrutor e reparador $d \in \Omega^-$ e $r \in \Omega^+$ usando ρ^- e ρ^+;
5: $x^t = r(d(x))$;
6: **if** aceite(x^t, x) **then**
7: $x = x^t$;
8: **end if**
9: **if** $c(x^t) < c(x^b)$ **then**
10: $x^b = x^t$;
11: **end if**
12: atualizar ρ^- e ρ^+;
13: **end while**
14: **return** x^b
15: **end procedure**

Os pontos que diferenciam o *ALNS* do *LNS* são os pontos 2, 4 e 12 onde, respetivamente: são inicializados os pesos (ρ^- *e* ρ^+) com valor unitário; é selecionado o par de operadores de destruição d (do conjunto de operadores destrutores Ω^-) e reparação r (do conjunto de operadores reparadores Ω^+) a usar na iteração tendo em conta os pesos dos mesmos; são atualizados os pesos.

4.2 Extensão do *ALNS - CVRP*

Tendo por base o referido em 4.1 foi implementado um algoritmo *ALNS* aplicado ao problema CVRP. Esta adaptação implica que o algoritmo consiga fazer o planeamento das rotas respeitando a regra fundamental do CVRP: a capacidade do veículo não pode ser ultrapassada em nenhuma das rotas.

Assim, a *framework* do algoritmo *ALNS* mantém-se semelhante à apresentada no final de 4.1. O novo algoritmo é apresentado de seguida no Algoritmo 3.

Algorithm 3

1: **procedure** ALNS($x \in \{soluções\ admissíveis\}$)
2: $x^b = x$; $\rho^- = (1,...,1)$; $\rho^+ = (1,...,1)$;
3: **while** critério de paragem não é alcançado **do**
4: Selecionar operadores destrutor e reparador $d \in \Omega^-$ e $r \in \Omega^+$ usando ρ^- e ρ^+;
5: $x^t = r(d(x))$;
6: **if** aceite(x^t, x) **then**
7: $x = x^t$;
8: **end if**
9: **if** $c(x^t) < c(x^b)$ **then**
10: $x^b = x^t$;
11: **end if**
12: atualizar *Scores*;
13: **if** #iterações = múltiplo de α **then**
14: Atualizar ρ^- e ρ^+ com recurso a *Scores*;
15: **end if**
16: **end while**
17: **return** x^b
18: **end procedure**

Onde a alteração se dá apenas dos pontos 12-15 e que é na atualização dos pesos. Estes passarão a ser atualizados recorrendo a *Scores* e apenas no fim de um segmento de iterações, passando os *Scores* a serem atualizados a cada iteração. Esta alteração será explicada mais à frente em 4.2.4.

A regra do CVRP é garantida num nível mais baixo do algoritmo, isto é, o código que garantirá que a regra é respeitada estará incluído, mais concretamente, nos operadores de reparação.

Alguns elementos no algoritmo são fundamentais como por exemplo: os operadores de destruição e reparação; a escolha dos mesmos; os pesos atribuídos aos pares destruição/reparação e o ajuste dos mesmos. A implementação destes elementos será descrita nos sub-capítulos seguintes.

De notar que a avaliação da solução irá ser baseada apenas na distância percorrida pelo veículo.

4.2.1 Solução inicial

Para o funcionamento do *ALNS* ser possível, o algoritmo tem de ter como *input* uma solução admissível. Então, o passo inicial é a criação de uma solução para que depois o algoritmo possa funcionar. Como esta solução inicial serve apenas como ponto de partida e não tem grande significado nesta fase a qualidade da solução, é utilizada uma heurística baseada na *Greedy Insertion* que é explicada mais à frente em 4.2.2.2.

Na heurística responsável pela criação da solução inicial, é percorrida a lista de clientes por ordem e, quando encontrada a melhor posição para cada um, estes são inseridos. A melhor posição será a posição onde a inserção do cliente acarreta o menor custo. Apenas depois de encontrada

a melhor posição para o cliente e depois deste ser inserido é que se passa ao próximo cliente. No caso de haverem dois clientes para a mesma posição ótima, da maneira que o algoritmo é implementado, é inserido o último dos dois a ser encontrado. Este processo é de novo repetido sucessivamente até que todos os clientes estejam inseridos.

4.2.2 Operadores *ALNS*

Os operadores são heurísticas desenhadas para alterar uma solução já existente para o planeamento das rotas e por isso são heurísticas de destruição e de reparação. No caso do *ALNS* e como referido em 4.1, podem-se ter ao dispôr pares de operadores que são adequados para um grupo restrito de instâncias porque o algoritmo encarrega-se de usar os mesmos nas instâncias adequadas. Nesta dissertação a escolha dos operadores foi baseada nos operadores utilizados por Ropke and Pisinger (2006) e serão apresentados a seguir.

4.2.2.1 Operadores de destruição

Os operadores de destruição têm como função remover clientes da solução resultante da iteração anterior. No desenho deste algoritmo foram utilizadas três heurísticas como operadores de remoção: *Shaw Removal*, *Random Removal* e *Worst Removal*.

Shaw Removal Esta heurística foi proposta em Shaw (1997) pelo próprio Shaw, e tem como objetivo a remoção de clientes que estão relacionados segundo um critério que é implementado. Este critério pode ser baseado em: distância entre clientes, capacidade dos clientes, tempos de serviço, etc. Isto é aplicado por ser expectável que clientes semelhantes sejam fáceis de permutar e que, como consequência, se obtenha a criação de novas e melhores soluções. Visto de outro prisma, ao não escolher clientes díspares evitamos que aconteça o caso de estarmos a remover clientes que só consigamos reinserir nas mesmas posições, não ganhando nada com a operação.

A semelhança entre dois clientes pode ser definida através de uma medida de semelhança representada por R_{ij}, e que significa a afinidade entre o cliente i e j. Neste caso, a medida de semelhança é obtida através de dois termos: a procura dos dois clientes e a distância geográfica entre os mesmos. Estes termos são pesados pelos respetivos índices a_1 e a_2 respetivamente. A medida de semelhança é então dada por:

$$R_{ij} = a_1 * \left(\left|p_i - p_j\right|\right) + a_2 * d_{ij} \tag{4.1}$$

Onde o termo pesado por a_1 mede a semelhança entre as procuras p_i do cliente i e p_j do cliente j. O termo pesado por a_2 mede a distância entre os clientes i e j, d_{ij}. De notar que os pesos a_1 e a_2 são normalizados e pertencem ao intervalo $[0,1]$.

A heurística tem como *inputs* três elementos: uma solução s pertencente ao conjunto de soluções admissíveis; uma quantidade q que quantifica o número de clientes a remover pela heurística; um parâmetro p que irá introduzir alguma aleatoriedade na seleção dos clientes a remover, sendo que um valor baixo de p significa uma maior aleatoriedade e a um valor alto de p corresponde uma

menor aleatoriedade. No Algoritmo 4, em forma de pseudocódigo, podemos ver o funcionamento da mesma.

Algorithm 4

1: **procedure** SHAWREMOVAL(s $\in \{$*soluções admissíveis*$\}$, q $\in \mathbb{N}$, p$\in \mathbb{R}_+$)
2: cliente: c = cliente aleatório de s removido;
3: conjunto de clientes removidos: D=$\{c\}$;
4: **while** $D < q$ **do**
5: c_t = cliente aleatório em D;
6: Vetor: L = clientes da solução por remover;
7: ordenar L tal que: $i < j \Rightarrow R(c_t, L[i]) < R(c_t, L[j])$;
8: escolher número aleatório n do intervalo $[0,1[$
9: $c = \{L[n^p |L|]\}$;
10: $D = D \cup \{c\}$;
11: **end while**
12: **end procedure**

Resumindo, como se pode verificar nos pontos 2 e 3, é escolhido inicialmente um cliente aleatório presente na solução s. Este cliente é colocado num vetor D que terá, no fim da heurística, o conjunto de clientes removidos da solução. De seguida temos um ciclo no ponto 4 no qual é escolhido novamente um cliente aleatório. Este cliente (c_t) é então comparado com os clientes da solução s ainda por remover, presentes no vetor L. No ponto 7 é feita a comparação e, em simultâneo, o vetor L é ordenado segundo a condição $R(c_t, L[i]) < R(c_t, L[j])$. Por outras palavras, o vetor L estará ordenado de forma crescente de R, tendo os clientes mais parecidos aos já removidos em primeiro lugar. Nos pontos 8-10 é introduzida alguma aleatoriedade segundo o valor do *input* p. Isto porque a posição do vetor de clientes ordenado L à qual se vai selecionar o próximo cliente a remover, depende de um número aleatório n compreendido entre 0 e 1, elevado a p. No fundo, ao termos um valor entre 0 e 1 elevado ao valor p: se p for elevado, o valor resultante será baixo, o que resultará na seleção dos clientes mais próximos do início do vetor L, o que por sua vez implica uma aleatoriedade baixa ou mesmo nula; se p for baixo, o valor resultante será alto o que resultará na seleção dos clientes mais próximos do final do vetor L, implicando uma aleatoriedade elevada.

Random Removal Esta heurística é bastante simples. Como *inputs* tem apenas s pertencente ao conjunto de soluções admissíveis e q, a quantidade de clientes a remover da solução. A ideia é que os clientes a ser removidos sejam removidos aleatoriamente. O *Random Removal* pode ser visto como o caso especial do *Shaw Removal* em que o p toma o valor 1, deixando assim que a seleção do cliente a remover seja completamente aleatório. Mas ao ser implementado separadamente, e sendo uma heurística simples, pode ser feito de forma a que seja mais rápido e consuma menos recursos que o *Shaw Removal*.

Worst Removal A heurística tem como ideia base a remoção dos clientes cuja remoção diminui mais acentuadamente o valor da função objetivo. Para isso é utilizada uma função que mede o quanto a função objetivo é diminuída com a remoção do cliente selecionado. Isto tem como objetivo melhorar a solução, retirando clientes que estão colocados numa posição na qual o seu custo, para a solução final, é elevado. A função pode ser escrita da seguinte forma: $custo(i,s) = f(s) - f_{-i}(s)$, onde $f(s)$ é o custo da solução e $f_{-i}(s)$ o custo da solução já sem o cliente i.

Algorithm 5

1: **procedure** WORSTREMOVAL(s $\in \{soluções\}$, q $\in \mathbb{N}$, p$\in \mathbb{R}_+$)
2: Vetor: L = clientes da solução por remover;
3: ordenar L por ordem decrescente de $custo(i,s)$;
4: $\#removidos = 0$;
5: **while** $\#removidos < q$ **do**
6: escolher número aleatório n do intervalo $[0,1[$
7: $c = \{L[n^p |L|]\}$;
8: remover c da solução;
9: $\#removidos = \#removidos + 1$;
10: **end while**
11: **end procedure**

Em certa medida o *Worst Removal* é semelhante ao *Shaw Removal*, tendo mesmo bastantes passos aproveitados do mesmo, como por exemplo nos pontos 2, 3, 6 e 7. A principal diferença está no ponto 3, em que o vetor é ordenado pelo valor de $custo(i,s)$ decrescente em vez da condição $R(c_t, L[i]) < R(c_t, L[j])$. Além dessa diferença, também as ideias em que os dois algoritmos assentam são diferentes. Enquanto a ideia do *Shaw Removal* é a de retirar da solução clientes que são fáceis de trocar uns com os outros, a ideia do *Worst Removal* é de retirar da solução os clientes que provavelmente estão em posições erradas da solução. O ponto 7 evita a situação em que são removidos sempre os mesmos clientes, ao ter em conta o valor p. O valor p tem o mesmo significado do que o valor p do *Shaw Removal* referido em 4.2.2.1.

4.2.2.2 Operadores de reparação

Os operadores de reparação, como o próprio nome indica, têm como função reparar a solução que foi destruída anteriormente pelos operadores destrutores, inserindo os clientes removidos. Daí a ligação entre os mesmos (par operadores destrutor/reparador), referida até agora em vários locais ao longo do capítulo 4. Estes operadores são bastante importantes no algoritmo, uma vez que é na implementação dos mesmos que é garantido que são respeitadas as restrições.

No desenho deste algoritmo foram utilizadas duas heurísticas como operadores de reparação: Greedy Insertion e K-Regret Insertion. Esta última heurística é considerada como apenas uma, mas pode ser desdobrada em K heurísticas visto que depende do K como irá ser explicado mais à frente em 4.2.2.2.

Greedy Insertion Esta heurística, tal como a *Random Removal* em 4.2.2.1, é uma heurística bastante simples. Como *input* tem apenas a solução s, e a ideia principal na qual a heurística assenta é também simples: inserir os clientes removidos anteriormente pelos operadores de destruição, por forma a que o valor da função objetivo aumente o mínimo possível.

Para tal, e considerando que o custo de inserir o cliente i numa dada posição é c_i, são percorridas todas as posições das rotas existentes e calculado este valor em cada uma delas. À medida que este processo vai avançando, vai sendo guardado o valor mínimo do mesmo e a posição na qual tal mínimo foi obtido, de maneira a que no final se tenha a posição na qual o cliente i tem o custo mínimo de inserção. De notar que estes valores apenas são guardados se o cliente poder ser inserido nesta posição, isto é, se consoante a procura do cliente, a capacidade máxima do veículo não for ultrapassada ao inseri-lo na posição. Por outras palavras, os valores são guardados se a soma da procura do cliente e da capacidade atual da carga do veículo não ultrapassar a capacidade máxima do mesmo. Caso não seja encontrada uma posição em que seja possível inserir o cliente em questão, é criada uma nova rota para esse cliente. No final, inserimos o cliente i na posição na qual a inserção do mesmo aumenta o menos possível o valor da função objetivo da solução ou, caso não tenha sido encontrada posição, numa nova rota. De seguida repetimos de novo o processo para o cliente seguinte.

***K*-Regret Insertion** A heurística *Regret* foi já usada por Potvin and Rousseau (1993) num problema *Vehicle Routing Problem and Scheduling Problem with Time Windows* (VRSPTW), em que usa uma medida generalizada de *regret* em vez de uma medida de impacto da inserção na função objetivo.

Esta heurística é um bocado mais complexa do que a maioria das heurísticas utilizadas e referidas anteriormente em 4.2.2.1 e 4.2.2.2. Procura ter uma visão mais abrangente na decisão de inserção dos clientes, isto é, tenta prever o impacto de tomar a decisão de inserir numa rota específica em vez de outra qualquer. Por outras palavras ainda, esta heurística tenta inserir em primeiro lugar os clientes que irão custar demasiado se forem inseridos posteriormente, ou seja, clientes que nos iremos arrepender se não forem inseridos já e daí o nome da heurística. Isto ficará percetível mais à frente.

Assumindo $x_{ik} \in \{1,...,m\}$, em que m é o número total de rotas e x_{ik} é a rota onde o cliente i tem o k^{o} de custo mais baixo de inserção. Por exemplo: x_{11} é a rota onde o cliente 1 tem o 1^{o} valor mais baixo de inserção, x_{15} é a rota onde o cliente 1 tem o 5^{o} valor mais baixo de inserção e assim sucessivamente. A função que define o valor de *regret* pode ser definida da seguinte maneira:

$$c_i^* = \Delta f_{i,x_{i2}} - \Delta f_{i,x_{i1}} \tag{4.2}$$

Isto significa que o valor *regret* é a diferença entre o custo de inserir o cliente na melhor rota e o custo de inserir o cliente na sua segunda melhor rota. No caso mais geral podemos definir o valor de *regret* como:

$$c_i^* = \sum_{j=1}^{k} \left(\Delta f_{i,x_{ij}} - \Delta f_{i,x_{i1}}\right) \tag{4.3}$$

E o objetivo final da heurística, independentemente do valor de k, será:

$$max\ c_i^* \tag{4.4}$$

Algorithm 6

1: **procedure** K-REGRETINSERTION(s $\in \{soluções\ admissíveis\}$, K $\in \mathbb{N}$)
2: Vetor de estruturas: L = melhores posições em cada rota de clientes ainda por inserir, # de rotas onde clientes podem ser inseridos;
3: **while** tamanho de $L \neq 0$ **do**
4: Ordenar crescentemente L por # de rotas onde clientes podem ser inseridos;
5: c = cliente em L com maior valor de *regret*;
6: Inserir c em s;
7: Remover c de L;
8: Atualizar L;
9: **end while**
10: **end procedure**

De notar o vetor L no ponto 2 é um vetor de estruturas constituídas essencialmente por: identificação do cliente, por um multimap e pelo número de rotas em que o cliente pode ser inserido. Um multimap é um *container* associativo e os *containers* são estruturas de dados. Um multimap é então uma estrutura de dados associativos, isto é, não mantêm a ordem de inserção dos dados. Os dados inseridos têm associados uma *key* pela qual o multimap será, por defeito, ordenado crescentemente. Desta forma ao ter uma estrutura em que temos a informação acerca do cliente, juntamente com um multimap em que o valor key é o custo mínimo de inserção e o valor associado é um *pair* que interliga a rota onde este custo mínimo foi encontrado e a posição da rota onde inserir o cliente tem esse custo, temos acesso ao seguinte: o cliente a inserir, a posição na rota onde inserir e o custo de inserir. Além disso, temos esta informação ordenada por custo mínimo. Ou seja, temos o custo mínimo de inserir um dado cliente em cada uma das rotas da solução e a respetiva posição. Resumindo, a 1ª melhor rota, a 2ª melhor rota, etc, de cada cliente ainda por inserir na solução.

A razão pela qual é necessário que cada estrutura do vetor de estruturas L contenha o número de rotas nas quais o cliente respetivo pode ser inserido assenta na ideia seguinte: se não se der prioridade aos clientes que têm menos rotas onde podem ser inseridos, pode-se correr o risco de inserir outros clientes nas únicas rotas onde os primeiros podiam ser inseridos, fazendo assim com que os mesmos deixem de ter rotas possíveis. Isto significa que se dá prioridade aos clientes nos quais se tem menos margem de manobra para inserir. Isto é precavido no ponto 4.

No ponto 5 é utilizada a função representada na equação 4.2, que é o ponto fundamental da heurística como é possível de entender tendo em conta o referido ao longo desta explicação. Esta função serve para encontrar o cliente que irá ser inserido, recorrendo a L. Os pontos 6 e 7 são pontos simples onde o cliente escolhido no ponto anterior é inserido na solução e removido de L.

De notar que se não houver rota onde o cliente possa ser inserido, é criada uma rota para este. Após a remoção no ponto 7, L tem de ser atualizado como consequência da alteração na solução, mais concretamente na rota onde o cliente foi inserido. Isto é realizado no ponto 8.

4.2.3 Escolha do par destrutor/reparador

A escolha do par de operadores destrutor/reparador tem um papel importante no ALNS. Parte da capacidade de adaptação do *ALNS* deve-se à mesma, isto porque se poderia escolher os operadores aleatoriamente, não tirando partido do ajuste do valor dos pesos ao longo do *ALNS*. Este ajuste irá ser explicado mais à frente em 4.2.4. Mas uma grande vantagem do *ALNS* advém de tirar proveito desse valor de ajuste, ao escolher pares de operadores destrutores/reparadores que têm tido uma melhor performance e consequentemente pesos mais elevados. Assim, escolhemos o par que mais se adequa no momento para que se tenha uma maior possibilidade de obter soluções melhores.

Assim, e tendo usado nesta dissertação três operadores destrutores (*Shaw Removal*, *Random Removal* e *Worst Removal*) e quatro reparadores (*Greedy Insertion*, *2-Regret*, *3-Regret* e *4-Regret*), tem-se ω_{ij} como o peso do par destrutor/reparador ij, e a probabilidade do par ser escolhido é dada por:

$$\frac{\omega_{ij}}{\sum_{i=1}^{3}\sum_{j=1}^{4}\omega_{ij}} \tag{4.5}$$

De notar que a escolha dos operadores é feita ao par, tal como o ajuste dos scores. Isto porque se acredita que o benefício de ser considerado o par, em oposição a escolher um operador destrutor e um reparador individualmente, será maior.

4.2.4 Ajuste dos pesos

A ideia principal por trás da associação de pesos a cada par de operadores destruição/reparação referida em 4.1, está relacionada com a escolha do par mais apropriado a cada situação. Isto é, inicialmente os pesos são escolhidos aleatoriamente e ao longo do desenvolvimento do *ALNS* os pesos vão sendo atualizados conforme a prestação do par, o que permite ao *ALNS* escolher os pares com melhores resultados para a instância em processamento. O *ALNS* é constituído por vários ciclos, e cada um deles é considerado uma iteração. Em cada uma dessas iterações é usado um par destrutor/reparador, e para avaliar a sua performance é atualizado um score que também está associado ao par. O número total de iterações do *ALNS* é dividido em segmentos de 100 iterações, e no fim de cada segmento os pesos são atualizados com base nos scores obtidos por cada par. Os scores vão sendo incrementados de um parâmetro consoante a sua performance e segundo a seguinte tabela:

Tabela 4.1: Parâmetros incrementais dos scores

Parâmetro	Descrição
σ_a	O par aplicado criou uma nova melhor solução global
σ_b	O par aplicado criou uma solução melhor que a anterior
σ_c	O par aplicado criou uma solução pior

Assim, o *ALNS* tem a capacidade de acompanhar a performance de cada par. A atribuição do valor a cada um dos parâmetros é feita por forma a premiar os pares que obtêm uma melhor performance, sendo que pares cuja performance não seja tão boa continuam a ter um incremento de um parâmetro σ_c. O facto de serem premiados pares de operadores com boa performance parece lógico. Premiar pares que levem a soluções piores do que as que já foram atingidas não tem um sentido tão óbvio, mas o objetivo desta atribuição de prémio prende-se com a diversificação desejada no ALNS. Ao premiar soluções piores permite-se que novos locais do espaço de soluções admissíveis sejam pesquisados, bem como se permite que a pesquisa não fique presa em ótimos locais.

No final de cada segmento, os incrementos dos scores em cada iteração são tidos em conta para a atualização dos pesos segundo a seguinte fórmula:

$$\omega_{ij,k+1} = \omega_{ij,k} * (1-r) + r * \frac{\pi_{ij}}{\theta_{ij}} \tag{4.6}$$

Onde k é o segmento de iterações, π_{ij} é o score do par de operadores ij obtido no segmento anterior, e θ_{ij} é o número de vezes que esse par foi utilizado no segmento anterior. De notar que o valor de θ_{ij} será realmente o mínimo entre o valor unitário e o valor registado por θ_{ij}, evitando assim ter um valor nulo como denominador. O elemento que resta, r, controla a rapidez de reacção dos pesos aos scores, ou seja, à performance dos operadores. Quanto mais baixo o valor de r, menos os pesos têm em conta os scores. No caso de $r = 0$, os pesos não têm em conta os scores obtidos no segmento anterior, e os pesos do segmento anterior definem completamente os novos pesos. No caso de $r = 1$, os pesos são totalmente definidos pelos scores obtidos no segmento anterior.

Isto significa que os pesos a utilizar no segmento seguinte são definidos pelos scores do segmento anterior, com um nível determinado pelo valor do parâmetro de rapidez de reacção r e tendo em conta o número de vezes que o operador foi utilizado.

4.2.5 Critério de aceitação

Como referido em 4.1, depois de aplicados os pares de operadores destrutores/reparadores que foram escolhidos, a nova solução alcançada é avaliada através de um critério de aceitação. A solução é então:

1. Sempre aceite se for melhor do que a melhor solução já encontrada até ao momento;
2. Sempre aceite se for melhor que a solução anterior;

3. Aceite de acordo com um critério de aceitação se for pior que a anterior.

Nesta dissertação o critério escolhido é baseado no critério utilizado no SA, tendo por base o artigo Ropke and Pisinger (2006). A nova solução será, por isso, aceite com uma probabilidade definida por:

$$e^{-\frac{(f(s')-f(s))}{T}} \tag{4.7}$$

Onde s' é a nova solução, s a solução atual e T a temperatura, com $T > 0$. T é definido inicialmente tendo em conta a solução inicial e por forma a que uma solução que seja pior $x\%$ do que a solução atual, tenha y% probabilidade de ser aceite. No caso desta dissertação foi definido $x = 2\%$ e $y = 50\%$. Depois, e à medida que o *ALNS* vai evoluindo e as iterações aumentando, T vai sendo decrementado com uma razão a (ritmo de arrefecimento) aplicando $T = a * T$ com $a \in [0, 1]$.

Assim, nos momentos iniciais do *ALNS* é permitida uma grande diversificação e, posteriormente, quanto maior for o número de iterações, menor será a probabilidade de ser aceite uma solução pior. Isto que significa que o *ALNS* começa a convergir.

4.3 Extensão do *ALNS* para *CVRP - VRPB*

A principal diferença entre o desenho do algoritmo para ser aplicado a um *CVRP* e a um *VRPB* é a existência de *backhauls* neste último. A restrição de capacidade mantém-se, mas tem de ser aplicada, independentemente, nas partes *linehaul* e *backhaul* de cada rota. Isto porque, como referido em 3.2, os clientes *linehaul* de uma rota são todos visitados antes dos clientes *backhaul*, tendo assim a capacidade total do veículo como restrição de capacidade na rota *linehaul*. Como depois de todos visitados o veículo foi esvaziado, tem-se de novo a capacidade total como restrição para a realização da rota *backhaul*.

Como consequência desta diferença fundamental, são alterados: a criação da solução inicial, todos os operadores destrutores, todos os operadores reparadores e a avaliação da solução.

4.3.1 Solução inicial

A forma como a solução inicial estava a ser realizada pôde ser toda ela aproveitada, sendo apenas necessário acrescentar os fornecedores. Assim, a solução inicial é criada da mesma forma que em 4.2.1, e no final da mesma são incluídos os fornecedores. Como foi explicado em 3.2, no setor onde o caso de estudo se enquadra, o veículo apenas precisa de visitar um fornecedor para ficar totalmente lotado. Além disso, não há a obrigatoriedade de visitar todos os *backhauls*, nem a obrigatoriedade de todas as rotas incluírem um *backhaul*.

Como resultado é necessária a inclusão de um benefício de visita a um determinado *backhaul* para determinar se o mesmo fornecedor é inserido ou não. Este benefício será daqui em diante referido como prémio e irá ser explicado mais à frente em 4.3.3.

4.3.2 Operadores *ALNS*

Tal como na solução inicial, também nos operadores é agora necessário ter em conta os fornecedores (*backhauls*), bem como o prémio de visita aos mesmos.

Para tal, os operadores sofrem alterações. O operador *Greedy Insertion*, e sendo ele a base para a heurística de criação da solução inicial, é o único que se comporta da mesma forma. Isto é, o algoritmo é todo aproveitado sendo apenas acrescentado no final do mesmo a inserção de fornecedores, tendo em conta o prémio de visita aos mesmos além da capacidade do veículo. Todos os restantes são aproveitados de maneira diferente. Isto porque todos os outros não são alterados apenas no final, mas sim como um todo: os clientes a inserir ou remover são escolhidos e só depois é tido em conta se se trata de um tipo de clientes ou outro (*linehaul* ou *backhaul*), ao contrário da solução inicial e da heurística *Greedy Insertion*, onde como referido anteriormente, são tratados primeiro os clientes e depois os fornecedores. Conforme o tipo de cliente, é de novo tido em conta (ou não) o prémio de visita ao fornecedor além da distância entre clientes.

Para uma melhor compreensão será dado de seguida uma explicação de cada um dos tipos de aproveitamento.

Greedy Insertion Esta heurística é toda aproveitada sendo apenas acrescentada uma fase final. Com as alterações este passa a ter duas fases: a primeira fase é igual à original, onde são inseridos todos os clientes; a segunda fase é a novidade, onde são inseridos todos os fornecedores para os quais exista benefício em inserir em qualquer uma das rotas.

Random Removal Esta heurística é aproveitada como um todo, isto porque mesmo com as alterações continua a existir apenas uma fase. Anteriormente esta fase era constituída pela escolha de clientes e consequente remoção da solução. Com as alterações passa a ser escolhido um vértice (independentemente de ser cliente ou fornecedor) que é posteriormente removido. A remoção é efetuada consoante o tipo de vértice.

4.3.3 Avaliação da solução

Foi referido anteriormente em 4.3.1 que para se saber se um *backhaul* iria ser visitado era associado um prémio p_j ao mesmo. Este dado é conhecido e fornecido pelo utilizador ao *ALNS* como *input*. A cada *backhaul* é então associado um prémio para que se possa concluir se compensa inserir esse cliente na rota ou não. A forma para tomar esta decisão é explicada a seguir em 4.3.3.1.

A avaliação da solução continua tal como na extensão para *CVRP*, com o acréscimo do prémio de cada *backhaul* na solução.

4.3.3.1 Prémio de visita a *backhaul*

Considerando que estamos no final de rota e temos o custo c_{i0}, onde i é o último cliente *linehaul* na rota e 0 o armazém, este seria o custo incluido no custo total da solução final na extensão *CVRP*.

Agora com as alterações iríamos ter um *backhaul* j que iria ser inserido entre i e o armazém, e que sem a inclusão do prémio para o problema *VRPB* o custo passaria de c_{i0} para:

$$c_{ij} + c_{j0} \tag{4.8}$$

Como o prémio tem uma conotação positiva iria-se refletir neste custo da seguinte maneira:

$$c_{ij} - p_j + c_{j0} \tag{4.9}$$

Onde como é fácil de entender p_j é o prémio associado ao cliente *backhaul* j e o custo resultante será inferior ao custo obtido na fórmula 4.8.

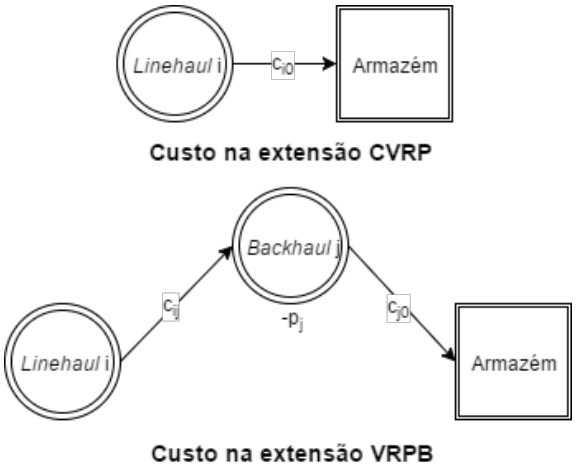

Figura 4.2: Ilustração da inclusão do prémio

De notar de novo que o *backhaul* j apenas será inserido se compensar que seja inserido, isto é conseguido ao inserir j apenas se:

$$c_{i0} > c_{ij} - p_j + c_{j0} \tag{4.10}$$

Cumprindo assim o objetivo de haver um benefício associado à passagem por um cliente *backhaul* j inserido na solução.

4.4 Extensão do *ALNS* para *VRPB - VRPB* robusto

Como explicado anteriormente o valor do prémio pode sofrer uma variação o que implica que este elemento irá estar sujeito a uma incerteza. Ao utilizar a otimização robusta garante-se que se tem uma boa solução para o problema, mesmo no caso em que o prémio é mais baixo do que o esperado inicialmente, ou seja, garante-se que se tem uma boa solução mesmo nos piores casos. De seguida em 4.4.1 será explicada a forma encontrada para introduzir robustez no *ALNS*.

4.4.1 Perturbação dos custos

Partindo de uma solução em que apenas é tido em conta um valor fixo para o prémio (p_j) atribuído aos fornecedores, o custo total poderá ser alterado se este valor sofrer uma variação. Neste sentido, o pior caso será quando a alteração que p_j sofre faz com que este tome o valor mais baixo possível. Seria o caso em que o benefício de passagem no mesmo seria também o mais baixo. Sendo assim, e dependendo do nível de conservadorismo do utilizador, a ideia é de permitir uma maior ou menor perturbação no custo correspondente ao conservadorismo. Um maior nível de conservadorismo corresponderia à ocorrência de uma maior quantidade de piores casos, isto é, de uma maior quantidade de benefícios baixos na passagem pelos fornecedores. Pelo contrário, um nível de conservadorismo mais baixo traduzir-se-á numa menor quantidade de piores casos. O nível de conservadorismo é representado pelo parâmetro Γ introduzido pelo utilizador. De notar que quando Γ toma um valor nulo, estamos na presença do caso determinístico. Quando Γ toma valor unitário, é aplicado um desvio inteiro no fornecedor com maior desvio possível no prémio. Quando $\Gamma = 1.5$, é aplicado um desvio tal como no caso unitário, mais meio desvio no fornecedor com o segundo maior desvio possível no prémio. E assim sucessivamente.

Este valor gama, está também presente na formulação matemática do problema *RVRPB* apresentada em 3.3.2, mais concretamente na equação 3.30.

Para introdução desta robustez no algoritmo *ALNS*, este tem de ser alterado. Terá de ser acrescentado um estado onde será introduzida a robustez na solução antes desta ser avaliada. Esta alteração pode ser vista no Algoritmo 7.

Algorithm 7

1: **procedure** ALNSROBUSTO($x \in \{soluções\ admissíveis\}$, Γ)
2: $x^b = x$; $\rho^- = (1,...,1)$; $\rho^+ = (1,...,1)$;
3: **while** critério de paragem não é alcançado **do**
4: Selecionar operadores destrutor e reparador $d \in \Omega^-$ e $r \in \Omega^+$ usando ρ^- e ρ^+;
5: $x^t = r(d(x))$;
6: InsereRobustez(Γ);
7: ReavaliaFornecedores(Γ);
8: **if** aceite(x^t, x) **then**
9: $x = x^t$;
10: **end if**
11: **if** $c(x^t) < c(x^b)$ **then**
12: $x^b = x^t$;
13: **end if**
14: atualizar *Scores*;
15: **if** #iterações = múltiplo de α **then**
16: Atualizar ρ^- e ρ^+ com recurso a *Scores*;
17: **end if**
18: **end while**
19: **return** x^b
20: **end procedure**

É adicionado o ponto 6 em relação ao algoritmo 2 bem como o *input* Γ. Neste ponto o objetivo da função $InsereRobustez(\Gamma)$ é encontrar os fornecedores que irão sofrer perturbações, e a ordem e nível de perturbação a aplicar aos mesmos. Tal como referido anteriormente, dependendo do nível de conservadorismo (consoante o Γ), irão ser afetados por perturbações mais ou menos fornecedores. Isto irá tornar a solução robusta. Uma ideia semelhante foi utilizada por Toklu (2014). A função é detalhada no pseudocódigo a seguir:

Algorithm 8

```
1:  procedure INSEREROBUSTEZ(Γ)
2:      Vetor: L = fornecedores na solução;
3:      dp_i = desvio de prémio do fornecedor i * #vezes que fornecedor i está presente na solução;
4:      Ordenar L por dp_i decrescente;
5:      while L não estiver vazio do
6:          if Γ ≥ 1 then
7:              custo_solução = custo_solução + dp_i;
8:              Γ = Γ − 1;
9:              Eliminar fornecedor i de L;
10:         else if Γ > 0 ou Γ < 1 then
11:             custo_solução = custo_solução + dp_i ∗ Γ;
12:             Γ = 0;
13:             Eliminar fornecedor i de L;
14:         else
15:             Eliminar fornecedor i de L;
16:         end if
17:     end while
18: end procedure
```

É importante notar que no ponto 3 e no caso de um fornecedor estar em mais do que uma rota, o desvio é multiplicado pelo número de vezes que o fornecedor está presente e é esse o valor que é tido em conta na ordenação (ponto 4).

Nos pontos 2-4 é definido um vetor que terá os fornecedores presentes na solução e é definido o critério de ordenação do mesmo. Este critério é o desvio do prémio do fornecedor i (dp_i) multiplicado pelo número de vezes que o mesmo é visitado na solução, e a ordenação do vetor L é descrescente segundo dp_i.

No ciclo, e dependendo do valor de *input* Γ, é realizado um de três processamentos:

1. Se $\Gamma \geq 1$, o custo da solução sofre um incremento igual a um dp_i, ou seja, o desvio do fornecedor i é acrescentado o número de vezes que o mesmo está na solução. De seguida o valor de Γ é decrementado de uma unidade e o fornecedor i é eliminado do vetor.

2. Se Γ estiver entre 0 e 1, o custo da solução sofre um incremento igual a um dp_i multiplicado pelo valor de Γ. De seguida o valor de Γ é anulado e o fornecedor i é eliminado do vetor.

3. Se Γ não se enquadrar em nenhum dos outros casos, apenas é eliminado o fornecedor.

Todo este processamento se resume a acrescentar o desvio correspondente ao valor de Γ, introduzido pelo utilizador, aos fornecedores. Este desvio tem de ser aplicado primeiro aos fornecedores que causarão um maior desvio na solução, e o valor de Γ significa o número de fornecedores aos quais este desvio irá ser aplicado. De notar que o desvio é aplicado ao fornecedor, e por isso

se este é visitado mais que uma vez, sofrerá esse desvio de todas as vezes. Para se perceber melhor a aplicação, se se tiver dois fornecedores e um $\Gamma = 1.5$:

1. Irá ser aplicado um $\Gamma = 1$, ou seja, um desvio, de todas as vezes que o fornecedor com maior desvio total (desvio multiplicado pelo número de vezes que fornecedor é visitado) é visitado.

2. Irá ser aplicado um $\Gamma = 0.5$, ou seja, meio desvio, de todas as vezes que o fornecedor com segundo maior desvio total é visitado.

Depois de aplicado este algoritmo à solução, o valor do prémio atribuído a alguns dos fornecedores é alterado. Como consequência pode-se dar o caso em que a condição presente na equação 4.10 deixe de se verificar devido à alteração do termo p_j. Então e para que esta nova possibilidade seja detetada, e se necessário corrigida, a solução é reavaliada no que diz respeito à situação dos fornecedores no ponto 7 do algoritmo 7. A função *ReavaliaFornecedores*(Γ) é detalhada no pseudocódigo apresentado de seguida:

Algorithm 9

```
1:  procedure REAVALIAFORNECEDORES(Γ)
2:      Vetor: L = fornecedores na solução que sofreram alteração;
3:      Ordenar L por dimensão da alteração decrescente;
4:      while Houver rotas não verificadas do
5:          if Rota tem fornecedor then
6:              if fornecedor está em L then
7:                  Verifica a condição de fornecedor i;
8:                  if Condição não se verifica then
9:                      Remove fornecedor i da solução;
10:                 end if
11:             end if
12:         end if
13:     end while
14: end procedure
```

A ideia principal em que este pseudocódigo assenta, é percorrer as rotas da solução à procura de fornecedores que tenham sofrido uma alteração, e para esses voltar a verificar se ainda existe benefício em tê-los presentes na rota. Em caso negativo, estes fornecedores terão de ser removidos.

4.4.1.1 Exemplo de aplicação de métodos de robustez

Nesta secção irá ser exposta uma iteração do algoritmo na qual são aplicados os métodos de robustez e onde se sente o efeito do valor Γ, ao deixar de valer a pena ter um fornecedor quando aplicada a variação correspondente ao valor de Γ ao mesmo.

A instância escolhida para a exemplificação foi uma instância dos autores Goetschalckx e Jacobs-Blecha, retirada de Mikiela (2014). Esta instância é designada GA1 e tem os seguintes vértices:

- Armazém - vértice com a identificação 0;
- Clientes - vértices com a identificação de 1 a 20;
- Fornecedores - vértices com a identificação de 21 a 25;

Associados a estes 5 fornecedores temos os prémios e variações da tabela 4.2, e a correspondência entre os fornecedores e os dados da tabela é a da tabela 4.3.

Tabela 4.2: Valores de prémio e variação máxima utilizados no exemplo de aplicação dos métodos *InsereRobustez*(Γ) e *ReavaliaFornecedores*(Γ)

Elemento	Valores				
	#1	#2	#3	#4	#5
Prémio	438	404	454	392	464
Variação	242	194	239	231	229

Tabela 4.3: Correspondência entre valores de prémio e variação e a identificação dos fornecedores

	Valores				
	#1	#2	#3	#4	#5
ID Fornecedor	21	22	23	24	25

Com recurso aos valores apresentados nas tabelas 4.2 e 4.3 ilustradas anteriormente, foi executado o teste. No teste realizado, e que resulta nas figuras 4.3 e 4.4 que irão ser apresentadas, foi utilizado um valor de $\Gamma = 1.5$.

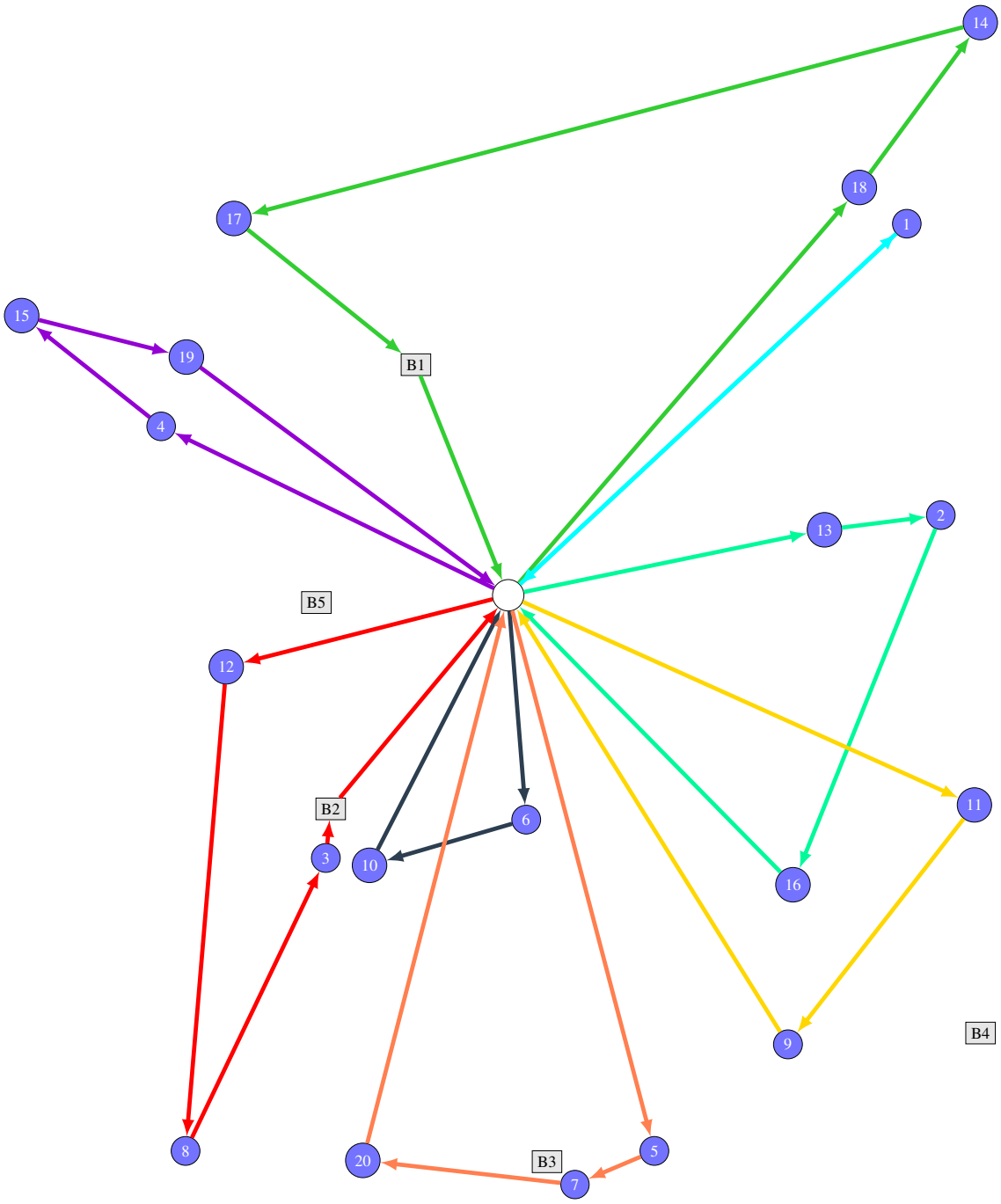

Figura 4.3: Imagem da solução antes da aplicação dos métodos *InsereRobustez*(Γ) e *ReavaliaFornecedores*(Γ)

Na figura 4.3, ilustrativa da solução antes da aplicação dos métodos *InsereRobustez*(Γ) e *ReavaliaFornecedores*(Γ), como se pode constatar, temos na solução dois fornecedores (B1, fornecedor 21; B2, fornecedor 22). Até este ponto o algoritmo apenas se limitou a criar solução inicial e aplicar os operadores destrutores/reparadores. Neste ponto são aplicados os métodos robustos.

Ao aplicar o método *InsereRobustez*(Γ), e visto que $\Gamma = 1.5$ irá ser aplicado um desvio total no fornecedor com maior variação possível e meio desvio no segundo fornecedor com maior variação

possível. Segundo as tabelas 4.2 e 4.3, é possível verificar que, nos casos dos fornecedores 21 e 22, temos: em primeiro lugar o fornecedor 21 com uma variação máxima possível de 242 e prémio 438; em segundo o fornecedor 22 com uma variação máxima possível de 194 e prémio 404. Sendo assim, e segundo o método no algoritmo 8, será aplicada uma variação de 242 ao fornecedor 21 e uma variação de $\frac{194}{2}$ ao fornecedor 22.

Como, neste caso, apenas se necessita de analisar a situação dos fornecedores 21 e 22, temos a seguinte tabela:

Tabela 4.4: Distâncias entre vértices

#	Ligação	Distâncias
	Rota número 1	
1	3 ->0	7642.35
2	3 ->22	1178.72
3	22 ->0	6637.44
	Rota número 4	
4	17 ->0	11137.9
5	17 ->21	5580.25
6	21 ->0	5934.82

Com o auxílio desta tabela, e conjugando com a fórmula 4.10 em 4.3.3.1, temos as seguintes situações:

Fornecedor 21

- Com prémio:

$$11137,9 > 5580,25 - 438 + 5934,82 \tag{4.11}$$
$$\Leftrightarrow 11137,9 > 11077,07 \tag{4.12}$$

  Com a verificação da equação, é comprovado o facto de termos o fornecedor 21 inserido na rota em questão, isto é, o facto de valer a pena inserir o fornecedor.

- Com prémio e desvio:

$$11137,9 > 11077,07 + 1 \times 242 \tag{4.13}$$
$$\Leftrightarrow 11137,9 > 11319,07 \tag{4.14}$$

  Com a inclusão da robustez, e consequentemente a introdução da variação, verifica-se que a equação é violada. Isto significa que com a variação deixa de ser benéfica a visita ao fornecedor.

Fornecedor 22

- Com prémio:

$$7642,35 > 1178,72 - 404 + 6637,44 \tag{4.15}$$
$$\Leftrightarrow 7642,35 > 7412,16 \tag{4.16}$$

  Com a verificação da equação, é também comprovado o facto de termos o fornecedor 22 inserido na rota em questão.

- Com prémio e desvio:

$$7642,35 > 7412,16 + 0,5 \times 194 \tag{4.17}$$
$$\Leftrightarrow 7642,35 > 7509,16 \tag{4.18}$$

  Com a inclusão da robustez, e consequentemente a introdução da variação, a equação continua a ser verificada. Sendo assim, continua a valer a pena o fornecedor estar presente na solução.

Sendo assim, a solução apresentada em 4.3 não poderia continuar igual. Teria de ser alterada, sendo que o fornecedor 21 teria de ser eliminado. Isto é conseguido com a aplicação do método *ReavaliaFornecedores*(Γ), obtendo-se a seguinte solução:

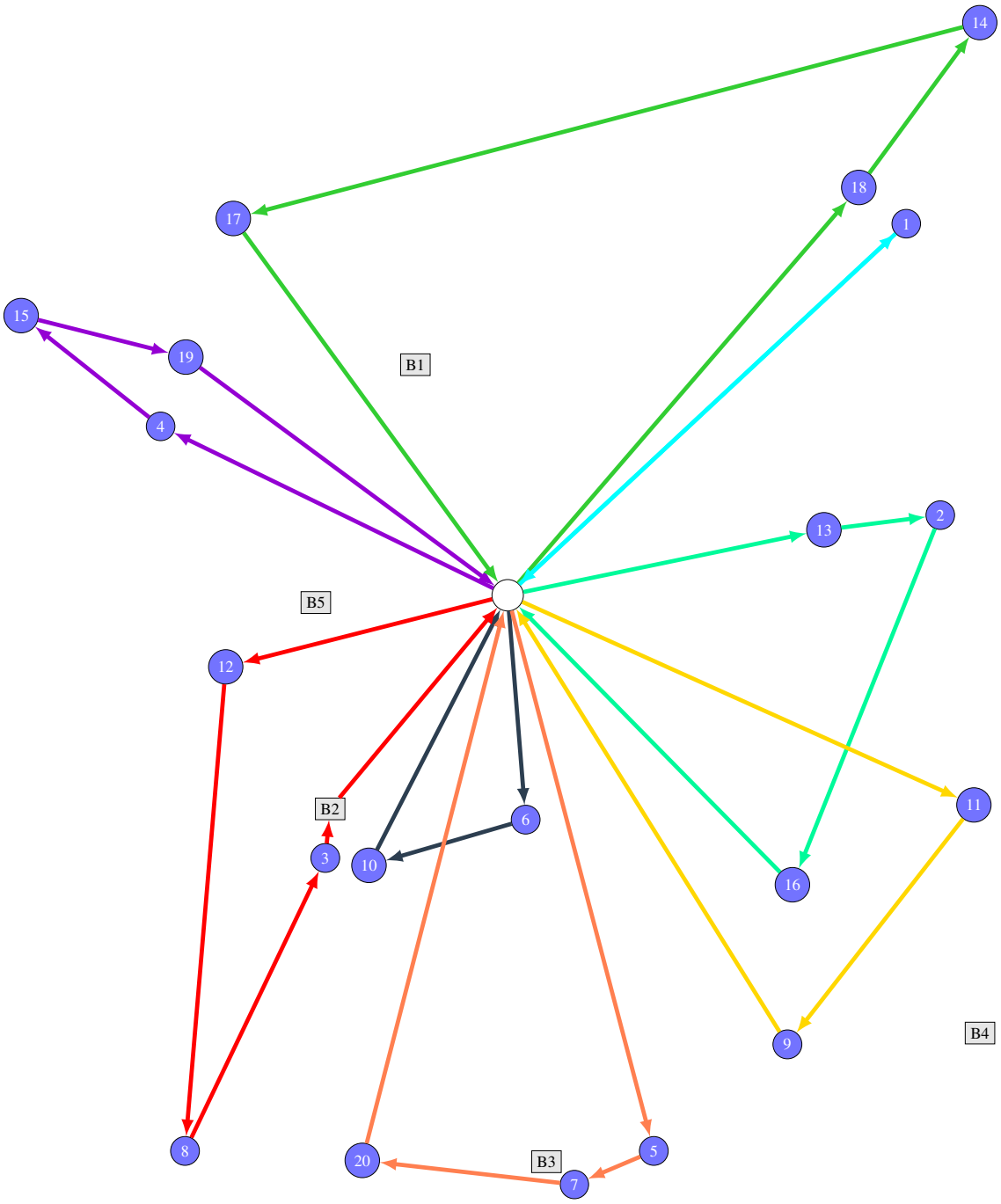

Figura 4.4: Imagem da solução depois da aplicação dos métodos *InsereRobustez*(Γ) e *Reavalia-Fornecedores*(Γ)

Como se pode comprovar pela figura, o fornecedor 21 (B1) deixa mesmo de ser visitado. Assim, é obtido o resultado final desejado com a introdução dos métodos de robustez.

Capítulo 5

Resultados e discussão

Neste capítulo serão apresentados os resultados de validação dos vários passos referidos no capítulo anterior e, no fim, apresentados os resultados do algoritmo final juntamente com uma breve discussão dos mesmos. Portanto, terá a mesma ordem do capítulo anterior, onde serão expostos os resultados primeiro do algoritmo aplicável ao *CVRP*, depois ao *VRPB* e por fim, ao *RVRPB*. Ao longo do capítulo serão também mencionadas as parametrizações utilizadas.

Na obtenção de todos os resultados apresentados foram utilizados dois computadores. O computador utilizado em todos os resultados, com a exceção dos resultados obtidos para o algoritmo final referentes ao método exato, possui um processador Intel(R) Core(TM) i7-4700MQ 2,40GHz, com 8GB RAM num sistema operativo Windows 10 Home de 64 *bits*. O computador utilizado na resolução do método exato, com o qual a aplicação do método aproximado ao *RVRPB* irá ser comparado, possui um processador Intel(R) Core(TM) i7-4790 3,60GHz, com 16GB de RAM num sistema operativo Windows de 64 *bits*. Para implementação do algoritmo foi utilizado o software *Microsoft Visual Studio 2017* e a linguagem escolhida foi *C++*. Para os resultados obtidos no método exato, foi utilizado o software *CPLEX* e a linguagem específica de modelação *OPL*.

5.1 *ALNS CVRP*

Como ficou percetível no capítulo anterior, este foi o primeiro algoritmo a ser preparado com o objetivo final de se obter um algoritmo para o problema da dissertação. Ao ser elaborado o algoritmo final em vários passos tem-se a possibilidade de ir validando os mesmos à medida que são terminados. Estas validações, por sua vez, são realizadas para que seja assegurada a qualidade do algoritmo ao longo do seu desenvolvimento, assim como para permitir uma rápida deteção de falhas e uma célere correção das mesmas.

Neste subcapítulo irão ser apresentadas as parametrizações utilizadas e os resultados obtidos na validação. Estes resultados são comparados com os *benchmark* da literatura para algumas instâncias, com o objetivo de verificar que o algoritmo conseguia obter os resultados ótimos. Foram escolhidas três instâncias retiradas de NEO Research Group (2013): as instâncias *A-n32-k5* (com

32 clientes e 5 veículos) e *A-n53-k7* (com 53 clientes e 7 veículos) de Augerat et al.; a instância *F-n45-k4* (com 45 clientes e 4 veículos) de Fisher.

Na tabela seguinte é apresentada a parametrização.

Tabela 5.1: Tabela de parametrização utilizada no algoritmo para *CVRP*

Parâmetro	Descrição	Valor
α	Número de iterações de um segmento	100
a_1	Pesos dos termos na fórmula da heurística *Shaw Removal*	0.5
a_2		0.5
σ_a	*Score* de obtenção de nova melhor solução global	15
σ_b	*Score* de obtenção de melhor solução que a anterior	10
σ_c	*Score* de pior solução	5
r	Rapidez de reacção dos pesos aos Scores	0.99
p	Grau de aleatoriedade de destruição	3.5
c	Ritmo de arrefecimento	0.99
-	Critério de paragem	1000

Utilizando estes valores para os parâmetros, foi testado o algoritmo obtendo-se os resultados apresentados de seguida.

Tabela 5.2: Resultados obtidos pelo *ALNS CVRP*

Instância	Resultado ótimo	Resultado obtido
A-n32-k5	784	784
A-n53-k7	1010	1010
F-n45-k4	724	724

Através dos resultados obtidos foi possível validar o algoritmo. Foram obtidos resultados satisfatórios ao serem obtidos os resultados ótimos nas três instâncias testadas. Tendo definido como critério de paragem a corrida de 1000 iterações, estes resultados foram obtidos em menos de 30 segundos. Isto permite a aprovação da qualidade do algoritmo e o avanço para a próxima validação com a segurança de que o algoritmo apresentava a performance desejada.

5.2 *ALNS VRPB*

Depois de validado o algoritmo elaborado para aplicação num *CVRP*, o passo seguinte foi a adaptação do mesmo para um problema *VRPB*. Após a adaptação referida, também este novo algoritmo teria de ser validado. Para tal, foram utilizadas adaptações de duas instâncias criadas pelos autores Goetschalckx e Jacobs-Blecha: a instância GA1 (26 vértices constituídos por 1 armazém, 19 clientes, 6 fornecedores e veículos com capacidade máxima de 1550Kg) e a GA2 (26 vértices constituídos por 1 armazém, 19 clientes, 6 fornecedores e veículos com capacidade máxima de

2550Kg). Como este algoritmo já seria semelhante ao algoritmo final foram utilizadas apenas estas duas instâncias, sendo que se fossem validadas, já seria um indício de que as alterações ao algoritmo estariam corretamente implementadas. Uma validação mais aprofundada seria realizada no algoritmo final.

A parametrização utilizada foi a ilustrada na tabela seguinte.

Tabela 5.3: Tabela de parametrização utilizada no algoritmo para *VRPB*

Parâmetro	Descrição	Valor
α	Número de iterações de um segmento	100
a_1	Pesos dos termos na fórmula da heurística *Shaw Removal*	0.5
a_2		0.5
σ_a	*Score* de obtenção de nova melhor solução global	15
σ_b	*Score* de obtenção de melhor solução que a anterior	10
σ_c	*Score* de pior solução	5
r	Rapidez de reacção dos pesos aos Scores	0.99
p	Grau de aleatoriedade de destruição	3.5
c	Ritmo de arrefecimento	0.99
-	Critério de paragem	10000

Onde a única alteração feita em relação à tabela 5.2 foi efetuada no critério de paragem onde se passa a usar um valor de 10000 iterações. Esta alteração foi levada a cabo para que fosse permitido ao algoritmo realizar iterações suficientes com a finalidade de aumentar a probabilidade de chegada ao melhor resultado.

De seguida, e com esta parametrização, irá ser apresentada a tabela com os resultados obtidos e como referência para os resultados ótimos foram utilizados os valores encontrados no trabalho realizado em Wassan et al (2017).

Tabela 5.4: Comparação de resultados obtidos pelo *ALNS VRPB*

Instância	Resultado ótimo	Resultado obtido
GA1	229885,65	229886
GA2	180119,21	180119

Mais uma vez, através dos resultados obtidos foi possível validar o novo algoritmo. Os resultados obtidos foram semelhantes aos resultados presentes em Wassan et al (2017), sendo que a pequena diferença se deve a arredondamentos. Embora nesta fase não fosse relevante o tempo despendido para a chegada ao resultado e razão pela qual foram utilizadas 10000 iterações como critério de paragem, os resultados foram, ainda assim, obtidos em tempos a rondar os 2 minutos. Com estes resultados estava aprovada a qualidade do algoritmo e era permitido avançar para a adaptação e implementação do algoritmo final ao *RVRPB*.

5.3 *ALNS RVRPB*

Depois das validações necessárias nos passos de elaboração do algoritmo para *CVRP* e posterior adaptação ao *VRPB*, estava-se na presença do passo final. Este passo seria a adaptação do algoritmo *ALNS VRPB* para o nosso problema *ALNS RVRPB*. Nesta fase e para validação final, foi realizada uma maior quantidade de testes. Para tal foram utilizadas 7 instâncias adaptadas, dos autores Goetschalckx e Jacobs-Blecha, retiradas de Mikiela (2014). As instâncias são as seguintes:

1. GA1 - 26 vértices, dos quais 1 armazém, 20 clientes e 5 fornecedores. A capacidade máxima dos veículos é de 1550Kg.

2. GA2 - 26 vértices, dos quais 1 armazém, 20 clientes e 5 fornecedores. A capacidade máxima dos veículos é de 2550Kg.

3. GA4 - 26 vértices, dos quais 1 armazém, 20 clientes e 5 fornecedores. A capacidade máxima dos veículos é de 4050Kg.

4. GB3 - 31 vértices, dos quais 1 armazém, 20 clientes e 10 fornecedores. A capacidade máxima dos veículos é de 4000Kg.

5. GD1 - 39 vértices, dos quais 1 armazém, 30 clientes e 8 fornecedores. A capacidade máxima dos veículos é de 1700Kg.

6. GD4 - 39 vértices, dos quais 1 armazém, 30 clientes e 8 fornecedores. A capacidade máxima dos veículos é de 4075Kg.

7. GN5 - 151 vértices, dos quais 1 armazém, 100 clientes e 50 fornecedores. A capacidade máxima dos veículos é de 8500Kg.

Como se pode ver a nomeação das instâncias apresentam um padrão na nomenclatura. Esse padrão é GXY e pode ser descrito da seguinte forma: o primeiro caractere (G) refere-se ao nome do autor da instância, Goetschalckx; o segundo caractere (X) refere-se a uma letra que agrupa as instâncias no que diz respeito ao número de vértices da mesma; o terceiro caractere (Y) refere-se à capacidade máxima dos veículos utilizados, e consequentemente, na capacidade máxima das rotas *linehaul* e *backhaul* individuais.

A parametrização utilizada é apresentada na tabela seguinte.

Tabela 5.5: Tabela de parametrização utilizada no algoritmo para *RVRPB*

Parâmetro	Descrição	Valor
α	Número de iterações de um segmento	100
a_1	Pesos dos termos na fórmula da heurística *Shaw Removal*	0.5
a_2		0.5
σ_a	*Score* de obtenção de nova melhor solução global	15
σ_b	*Score* de obtenção de melhor solução que a anterior	10
σ_c	*Score* de pior solução	5
r	Rapidez de reacção dos pesos aos Scores	0.99
p	Grau de aleatoriedade de destruição	3.5
c	Ritmo de arrefecimento	0.99
-	Critério de paragem	100000*

Uma vez mais, a diferença na parametrização encontra-se no critério de paragem. O valor apresentado de 100000 é o valor máximo de iterações que o algoritmo correrá até que pare. Para uma maior eficiência, temos um valor mínimo de iterações que seriam corridas de 10000. Isto é, neste caso, as 10000 iterações seria o número de iterações que teriam de ocorrer sem que houvesse alteração da melhor solução global encontrada. Ao contrário do algoritmo *ALNS VRPB* onde as 10000 iterações seriam simples e sem critério, ou seja, 10000 iterações seguidas apenas.

Esta particularidade tem como consequência a corrida de 10001 iterações no melhor caso, que seria o caso em que a primeira aplicação do par de operadores destrutor/reparador resultaria na melhor solução global, sendo que as seguintes 10000 seriam soluções piores do que esta.

Neste algoritmo é importante salientar que devido à sua especificidade (é um problema diferente de um *RVRPB* clássico), a comparação de resultados apenas se poderá fazer com os valores obtidos pelo método exato.

Para o correto funcionamento do algoritmo, e como mencionado no capítulo anterior, é necessário o fornecimento dos dados relativos ao prémio associado ao fornecedor por parte do utilizador. Estes dados seriam fornecidos tendo por base informação transata disponível. Nesta dissertação, e para teste, foram criados 10 grupos de valores aleatórios dos quais foram extraídos posteriormente 10 valores médios e 10 valores de desvio padrão. Os valores médios seriam o valor do prémio e os valores de desvio padrão seriam a variação máxima que cada um dos prémios poderia sofrer. Estes valores estão ilustrados na seguinte tabela.

Tabela 5.6: Valores de prémio e variação máxima utilizados

Elemento	Valores									
	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Prémio	438	404	454	392	464	380	384	355	459	415
Variação	242	194	239	231	229	201	235	222	229	253

Para instâncias cuja quantidade de fornecedores é menor do que 10, seriam utilizados os primeiros valores de prémio e variação começando por #1 até que todos os fornecedores tivessem valores atribuídos. Para quantidades de fornecedores maiores que 10, seriam utilizados os 10 primeiros valores por ordem, e a partir daí, voltariam a ser utilizados os valores a partir de #1. Por exemplo: numa instância com 7 fornecedores iriam ser utilizados os primeiros 7 valores da tabela; numa instância com 13 seriam utilizados os primeiros 10 valores para os primeiros 10 fornecedores e os últimos 3 fornecedores repetiriam os primeiros 3 valores da tabela.

De seguida irão ser apresentados em tabelas individuais os resultados obtidos para cada instância testada, tanto do método exato como da metaheurística.

Tabela 5.7: Comparação de resultados obtidos pelo *ALNS RVRPB* para a instância GA1

Instância	Gama	Método exato					Método aproximado				
		Custo final	Distância Percorrida (Km)	Retorno	Tempo (s)	GAP1*	Custo final	Distância Percorrida (Km)	Benefício	Tempo (s)	GAP2*
A1	0	218566	220276	1710	>600	**8%**	218566	220276	1710	98	0,000%
	0.5	218760	220276	1516	>600	**8%**	218760	220276	1516	56	0,000%
	1	218954	220276	1322	>600	**8%**	218954	220276	1322	146	0,000%
	1.5	**219075**	**220276**	**1201**	**>600**	**9%**	**219130**	**219899**	**769,5**	**244**	**0,025%**
	2	**219196**	**220285**	**1080**	**>600**	**9%**	**219206**	**219626**	**420**	**120**	**0,005%**
	2.5	**219197**	**219618**	**420**	**>600**	**10%**	**219244**	**219899**	**655**	**110**	**0,021%**
	3	219197	219618	420	>600	**9%**	219197	219617	420	80	0,000%
	3.5	219197	219618	420	>600	**10%**	219197	219617	420	134	0,000%
	4	219197	219618	420	>600	**11%**	219197	219617	420	111	0,000%
	4.5	219197	219618	420	>600	**13%**	219197	219617	420	158	0,000%

Tabela 5.8: Comparação de resultados obtidos pelo *ALNS RVRPB* para a instância GA2

Instância	Gama	Método exato					Método aproximado				
		Custo final	Distância Percorrida (Km)	Retorno	Tempo (s)	GAP1*	Custo final	Distância Percorrida (Km)	Benefício	Tempo (s)	GAP2*
A2	0	164457	165763	1306	>600	**8%**	164457	165763	1306	111	0,000%
	0.5	**164578**	**165763**	**1185**	**>600**	**8%**	**164632**	**165386**	**753,5**	**121**	**0,033%**
	1	**164699**	**165763**	**1064**	**>600**	**7%**	**164747**	**165386**	**639**	**145**	**0,029%**
	1.5	**164813**	**165763**	**949.5**	**>600**	**9%**	**164844**	**165386**	**542**	**111**	**0,019%**
	2	164894	165104	210	>600	**9%**	164894	165104	210	142	0,000%
	2.5	164894	165104	210	>600	**8%**	164894	165104	210	130	0,000%
	3	164894	165104	210	>600	**6%**	164894	165104	210	128	0,000%
	3.5	164894	165104	210	>600	**8%**	164894	165104	210	134	0,000%
	4	164894	165104	210	>600	**8%**	164894	165104	210	132	0,000%
	4.5	164894	165104	210	>600	**8%**	164894	165104	210	141	0,000%
	5	164894	165104	210	>600	**7%**	164894	165104	210	145	0,000%

Tabela 5.9: Comparação de resultados obtidos pelo *ALNS RVRPB* para a instância GA4

Instância	Gama	Método exato					Método aproximado				
		Custo final	Distância Percorrida (Km)	Retorno	Tempo (s)	GAP1*	Custo final	Distância Percorrida (Km)	Benefício	Tempo (s)	GAP2*
A4	0	136068	136532	464	63.95	0	136068	136532	464	91	0,000%
	0.5	136182	136532	349.5	60.63	0	136182	136532	349.5	144	0,000%
	1	136250	136250	0	51.718	0	136250	136250	0	123	0,000%
	1.5	**136250**	**136250**	**0**	**25.08**	**0**	**136297**	**136532**	**235**	**134**	**0,034%**
	2	136250	136250	0	20.42	0	136250	136250	0	117	0,000%
	2.5	136250	136250	0	55.33	0	136250	136250	0	123	0,000%
	3	136250	136250	0	43.05	0	136250	136250	0	111	0,000%
	3.5	136250	136250	0	40.22	0	136250	136250	0	127	0,000%
	4	136250	136250	0	44.41	0	136250	136250	0	126	0,000%
	4.5	136250	136250	0	27.23	0	136250	136250	0	95	0,000%
	5	136250	136250	0	14.50	0	136250	136250	0	124	0,000%

Tabela 5.10: Comparação de resultados obtidos pelo *ALNS RVRPB* para a instância GB3

Instância	Gama	Método exato					Método aproximado				
		Custo final	Distância Percorrida (Km)	Retorno	Tempo (s)	GAP1*	Custo final	Distância Percorrida (Km)	Benefício	Tempo (s)	GAP2*
B3	0	132731	134061	1330	2.5	0%	132731	134061	1330	153	0,000%
	0.5	132907	133678	771	2.5	0%	132907	133678	771	147	0,000%
	1	133028	133678	650	2.0	0%	133028	133678	650	180	0,000%
	1.5	133148	133679	530.5	2.1	0%	133148	133678	530.5	163	0,000%
	2	133267	133678	411	1.8	0%	133267	133678	411	157	0,000%
	2.5	133267	133678	411	6.7	0%	133267	133678	411	168	0,000%
	3	133267	133678	411	2.4	0%	133267	133678	411	206	0,000%
	3.5	133267	133678	411	4.2	0%	133267	133678	411	162	0,000%
	4	133267	133678	411	7	0%	133267	133678	411	150	0,000%
	4.5	133267	133678	411	8	0%	133267	133678	411	147	0,000%
	5	133267	133678	411	7	0%	133267	133678	411	180	0,000%
	5.5	133267	133678	411	7.7	0%	133267	133678	411	138	0,000%
	6	133267	133678	411	8.1	0%	133267	133678	411	137	0,000%
	6.5	133267	133678	411	8.3	0%	133267	133678	411	131	0,000%
	7	133267	133678	411	9.7	0%	133267	133678	411	147	0,000%
	7.5	133267	133678	411	5.4	0%	133267	133678	411	170	0,000%
	8	133267	133678	411	8.4	0%	133267	133678	411	155	0,000%
	8.5	133267	133678	411	7.8	0%	133267	133678	411	142	0,000%
	9	133267	133678	411	8.4	0%	133267	133678	411	157	0,000%
	9.5	133267	133678	411	9.1	0%	133267	133678	411	153	0,000%
	10	133267	133678	411	8.6	0%	133267	133678	411	156	0,000%

Tabela 5.11: Comparação de resultados obtidos pelo *ALNS RVRPB* para a instância GD1

Instância	Gama	Método exato					Método aproximado				
		Custo final	Distância Percorrida (Km)	Retorno	Tempo (s)	GAP1*	Custo final	Distância Percorrida (Km)	Benefício	Tempo (s)	GAP2*
D1	0	302872	304102	1230	>600	**12%**	289761	291428	1667	355	**-4,329%**
	0.5	302149	303260	1110.5	>600	**11%**	289979	290683	704.5	475	**-4,028%**
	1	304745	305260	607	>600	**12%**	298917	299507	590	431	**-1,912%**
	1.5	298210	299084	873.5	>600	**9%**	292082	292948	865.5	384	**-2,055%**
	2	303346	304102	756	>600	**12%**	292393	293144	751	380	**-3,611%**
	2.5	299205	299420	215	>600	**11%**	293211	293505	294	430	**-2,003%**
	3	299623	299838	215	>600	**11%**	293211	293505	294	536	**-2,140%**
	3.5	298421	298636	215	>600	**10%**	293211	293505	294	442	**-1,746%**
	4	298421	298636	215	>600	**11%**	293144	293305	161	385	**-1,768%**
	4.5	298421	298636	215	>600	**10%**	290235	290235	0	558	**-2,743%**
	5	301524	301739	215	>600	**10%**	292339	292500	161	615	**-3,046%**
	5.5	303745	303960	215	>600	**12%**	293144	293305	161	387	**-3,490%**
	6	298446	298661	215	>600	**11%**	293144	293305	161	524	**-1,777%**
	6.5	298421	298636	215	>600	**11%**	292339	292500	161	357	**-2,038%**
	7	303246	303461	215	>600	**12%**	290235	290235	0	471	**-4,291%**
	7.5	298421	298636	215	>600	**10%**	292339	292500	161	349	**-2,038%**
	8	298421	298636	215	>600	**11%**	293144	293305	161	419	**-1,768%**

Tabela 5.12: Comparação de resultados obtidos pelo *ALNS RVRPB* para a instância GD4

Instância	Gama	Método exato					Método aproximado				
		Custo final	Distância Percorrida (Km)	Retorno	Tempo (s)	GAP1*	Custo final	Distância Percorrida (Km)	Benefício	Tempo (s)	GAP2*
D4	0	176313	176697	384	>600	**9%**	166187	166542	355	207	**-5,743%**
	0.5	173324	173946	621.5	>600	**7%**	166298	166542	244	324	**-4,054%**
	1	176497	176497	0	>600	**9%**	166409	166542	133	346	**-5,716%**
	1.5	176497	176497	0	>600	**9%**	167307	167440	133	268	**-5,207%**
	2	176497	176497	0	>600	**9%**	166409	166542	133	261	**-5,716%**
	2.5	176497	176497	0	>600	**10%**	166280	166280	0	307	**-5,789%**
	3	176337	176337	0	>600	**9%**	167240	167240	0	325	**-5,159%**
	3.5	176497	176497	0	>600	**9%**	166280	166280	0	366	**-5,789%**
	4	176546	176546	0	>600	**10%**	167240	167240	0	315	**-5,271%**
	4.5	176497	176497	0	>600	**9%**	167240	167240	0	313	**-5,245%**
	5	176497	176497	0	>600	**9%**	166280	166280	0	352	**-5,789%**
	5.5	176546	176546	0	>600	**9%**	166280	166280	0	411	**-5,815%**
	6	176497	176497	0	>600	**10%**	166280	166280	0	304	**-5,789%**
	6.5	176497	176497	0	>600	**9%**	167240	167240	0	324	**-5,245%**
	7	176497	176497	0	>600	**9%**	167240	167240	0	340	**-5,245%**
	7.5	176497	176497	0	>600	**9%**	166280	166280	0	313	**-5,789%**
	8	176497	176497	0	>600	**10%**	166280	166280	0	276	**-5,789%**

Tabela 5.13: Comparação de resultados obtidos pelo *ALNS RVRPB* para a instância GN5

Instância	Gama	Método exato					Método aproximado				
		Custo final	Distância Percorrida (Km)	Retorno	Tempo (s)	GAP1*	Custo final	Distância Percorrida (Km)	Benefício	Tempo (s)	GAP2*
N5	0	-	-	-	1800	-	273126	274458	1332	1301	-%
	0.5	-	-	-	1800	-	275418	277057	1639	707	-%
	1	-	-	-	1800	-	279324	280198	874	2431	-%
	1.5	-	-	-	1800	-	274850	275392	542	2576	-%
	2	-	-	-	1800	-	276874	277728	854	2671	-%

De notar que temos dois GAPs referidos nas tabelas, um no método exato e outro no método aproximado. No método exato o GAP (GAP1) refere-se à diferença entre *Upper Bound* e *Lower Bound*, enquanto que no método aproximado o GAP (GAP2) refere-se à diferença entre os valores obtidos no próprio método e no método exato. Estes GAPs são obtidos da seguinte forma:

$$GAP1 = \frac{\textit{Lower Bound - Upper Bound}}{\textit{Lower Bound}} \times 100 \tag{5.1}$$

$$GAP2 = \frac{\textit{Custo Final (Método Aproximado) - Custo Final (Método Exato)}}{\textit{Custo Final (Método Exato)}} \times 100 \tag{5.2}$$

Nem em todos os casos para os quais os métodos (exato e aproximado) são comparados, são obtidos os mesmos valores. Estes casos apresentam apenas 7% (aproximadamente) da totalidade dos casos testados. Mas apesar disso, isto acontece para valores em que o método exato não consegue alcançar a solução ótima. Aproximadamente, em 50% dos casos foi obtida a mesma solução em ambos os métodos, e nas instâncias de maior dimensão que correspondem a aproximadamente 42% de todos os casos foram obtidos melhores resultados. De seguida irão ser apresentados, de forma mais detalhada, os dados retirados das tabelas anteriores.

Nos resultados obtidos na instância GA1, um dado a realçar é o facto de o método exato não conseguir chegar a um GAP1 (5.1) nulo em 600 segundos de corrida. Além disso, existem 3 valores de Γ para os quais o método aproximado não consegue obter o mesmo resultado que o método exato. Estes valores são $\Gamma = 1.5$, $\Gamma = 2$ e $\Gamma = 2.5$, para os quais se obtém os valores de GAP2 (5.2) 0.025%, 0.005% e 0.021% respetivamente. Estes valores apesar de não serem favoráveis, são irrisórios e são todos obtidos com um tempo máximo de 244 segundos.

Na instância GA2, mais uma vez, o método exato não consegue atingir um GAP1 de valor nulo, dificultando a comparação dos resultados obtidos. Existem,mais uma vez, 3 valores de Γ para os quais o método aproximado não consegue obter o mesmo resultado que o método exato. Neste caso, os valores são $\Gamma = 0.5$, $\Gamma = 1$ e $\Gamma = 1.5$, para os quais se obtém os valores de GAP2 (5.2) 0.033%, 0.029% e 0.019% respetivamente. Os valores de GAP2 voltam a ser desprezíveis.

Já na instância GA4 o método exato consegue obter um valor nulo para o GAP1. Nesta instância o método aproximado apresenta apenas um valor de Γ para o qual o mesmo resultado não é obtido. Este valor é $\Gamma = 1.5$, onde se verifica um valor de GAP2 de 0.034%. Mais uma vez o valor de GAP2 é ínfimo.

Na instância seguinte, GB3, os valores obtidos pelo método exato apresentam um GAP1 nulo, o que significa que estamos na presença da solução ótima para o problema. Quanto aos valores obtidos no método aproximado no que diz respeito ao GAP2, também eles têm todos o valor nulo. Sendo assim, tanto o método exato como o método aproximado, obtêm a solução ótima. Os resultados obtidos por ambos têm uma correspondência perfeita.

Até ao momento, todas as instâncias referidas têm um número máximo de 31 vértices como foi mencionado anteriormente, aquando da enumeração das instâncias e suas características. Daqui em diante as instâncias GD1, GD4 e GN5 têm todas mais de 38 vértices.

O método exato, na instância GD1, volta a não alcançar valores de GAP1 nulos. O que revela que o método exato volta a fracassar no que di respeito à obtenção da solução ótima. Os valores obtidos através do método aproximado para GAP2 são negativos, com uma média de -2.634% e que representam um valor já considerado significativo. Isto revela que o método aproximado obtém resultados melhores do que o método exato.

É a partir da quantidade de vértices utilizados nesta instância (inclusive) que o método exato deixa de funcionar como o desejado.

Na instância GD4, volta-se a verificar o mesmo que na instância anterior. Os valores de GAP1 não nulos e valores de GAP2 negativos com uma média de -5.479%. Este valor verifica-se ainda maior que na instância GD1, o que se traduz em resultados ainda melhores do método aproximado em relação ao que se verificou na instância GD1.

Nestas duas instâncias (GD1 e GD4) constata-se que quanto maior capacidade máxima apresentada pelos veículos, maior a diferença obtida pelo método aproximado em relação ao método exato, traduzida no GAP2. Pode-se verificar na conjugação entre a média de melhoria e a capacidade máxima do veículo, por instância, que é corroborada a ideia referida de correlação entre maior capacidade do veículo e maior média de melhoria.

Na instância GN5, que apresenta 151 vértices, o método exato deixa de conseguir apresentar resultados por completo. Com tempo de processamento de 1800 segundos não consegue chegar sequer a uma solução. O método aproximado, em contrapartida, alcança resultados, apesar de apresentar tempos elevados com um máximo de 2671 segundos para $\Gamma = 2$ devido ao complexo critério de paragem (onde no melhor caso o mínimo de iterações é de 10001), apresenta soluções.

Os valores obtidos para tempo médio de processamento do algoritmo são mais estáveis. Isto acontece porque o tempo de processamento do método aproximado está maioritariamente relacionado com o número de vértices que constituem a instância, apenas apresentando uma pequena dependência dos operadores utilizados durante o mesmo. Isto porque em instâncias nas quais os operadores destrutores/reparadores mais adequados são os mais pesados em termos de processamento, isso acaba por se refletir no tempo de processamento global do algoritmo.

Já no que diz respeito aos valores apresentados pelo método exato, conclui-se que os mesmos não apresentam a mesma estabilidade. Isto acontece porque além do tempo de processamento estar excessivamente dependente do número de vértices, também está igualmente dependente da capacidade máxima apresentada pelos veículos. Sendo que quanto maior a capacidade máxima do veículo para instâncias de tamanho adequado, maior a facilidade apresentada pelo método exato para garantir uma solução ótima.

Além disso, os únicos valores dos quais se pode ter a certeza que o processamento não demoraria mais do que o que se acabou por se suceder são os valores das instâncias GA4 e GB3. Isto porque como já foi referido anteriormente, estas instâncias são as únicas nas quais o método exato alcançou a solução ótima, ou seja, GAP1 nulo. Nas restantes instâncias, os valores para o tempo de processamento, referem-se ao limite máximo de tempo de processamento imposto.

De notar que a instância GN5 não se encontra com valor nenhum representado porque mesmo

com um tempo máximo de processamento imposto de 1800s (triplo do imposto nas outras instâncias), o método exato não conseguiu sequer chegar a uma solução.

Por fim, e com o objetivo de comparar a robustez obtida no caso determinístico, que se verifica para $\Gamma = 0$, e a robustez para os Γ robustos, foi realizada uma simulação Monte Carlo para a variação do prémio. Para tal, foi escolhida a instância GB3, visto ser a instância para a qual os dois métodos chegam à mesma solução, e com um GAP1 nulo, o que significa que estamos na presença da solução ótima ao longo de todos os valores de Γ para a instância em causa. Os valores testados foram dois: $\Gamma = 0$ e $\Gamma = 0.5$. Foram usados apenas estes dois valores por serem os valores onde a solução é alterada no que respeita a fornecedores. Na solução determinística ($\Gamma = 0$) tem-se: 2 vezes o fornecedor B1 e 1 vez o fornecedor B3. Na solução com $\Gamma = 0.5$ tem-se: 1 vez o fornecedor B1 e 1 vez o fornecedor B3. Através do processamento dos resultados da simulação Monte Carlo, onde como referido anteriormente se variou o prémio, obtiveram-se as seguintes curvas normais (5.1).

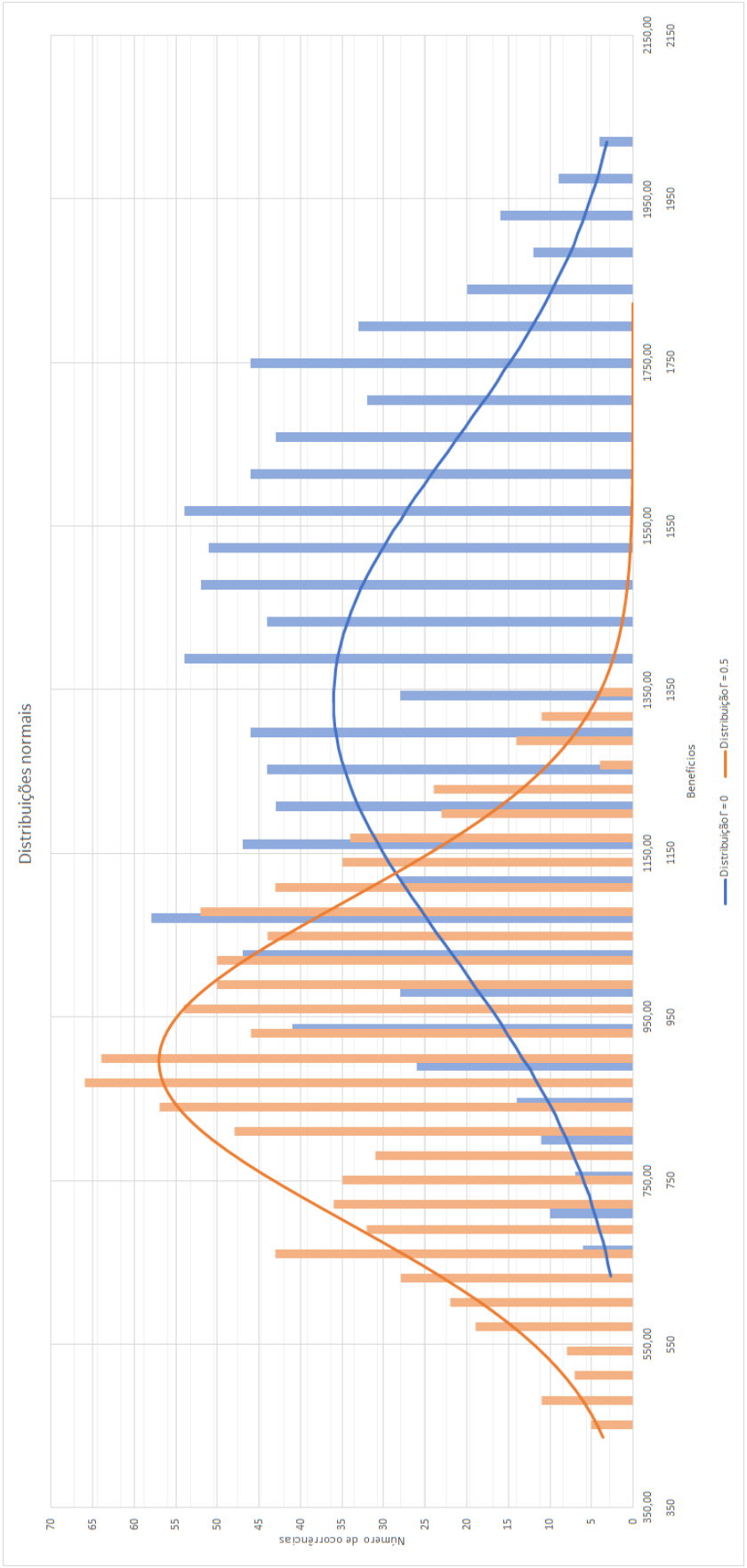

Figura 5.1: Distribuições normais aproximadas da função objetivo resultantes da simulação Monte Carlo

Tal como seria de esperar, os valores obtidos para o resultado determinístico ($\Gamma = 0$) apresentam-se mais dispersos do que no caso robusto. Os resultados obtidos no caso determinístico, devem-se ao facto de neste caso não ser considerada a incerteza. Isto significa que com a introdução de robustez no algoritmo, os valores obtidos serão mais concentrados, ou seja, sofrem menos variações. Estes resultados comprovam que, efetivamente, é alcançada uma maior robustez do que no caso determinístico, tal como o desejado. É também possível corroborar, através deste teste, a ideia de que quanto maior o valor de Γ, maior o conservadorismo. Este facto pode ser constatado através da análise do conjunto de valores de benefício alcançado no caso robusto, que se verifica menor do que no caso determinístico.

Capítulo 6

Conclusões e trabalho futuro

As conclusões expostas nestes capítulos têm por base todos os resultados apresentados no capítulo anterior, bem como os objetivos traçados no início deste trabalho.

O objetivo fulcral traçado para este projeto desde o começo, era a elaboração de um algoritmo para o problema de planeamento de rotas com fornecedores (retornos) tendo em conta a incerteza inerente a um prémio associado a cada fornecedor, com origem nas variações de qualidade da matéria-prima disponibilizada pelos mesmos. Este objetivo foi alcançado. Resultante deste objetivo, surge o desejo de obter resultados ótimos para todo e qualquer valor testado no algoritmo. Apesar de isto não acontecer em todos os valores comparados com os do método exato, os valores para os quais não se obtiveram os mesmos resultados que o método exato, são valores em que o método exato não conseguiu alcançar a solução ótima. Isto significa que é difícil de tirar conclusões com toda a segurança. Além disso, estes casos representam apenas 7% da totalidade de casos testados. Pelo contrário, em aproximadamente 50% dos casos testados foi obtida a mesma solução pelo método aproximado, que no método exato. Para instâncias de maior dimensão, que correspondem a 42% do total de casos testados, e onde o objetivo seria termos melhores soluções (ou em tempo de processamento ou no valor da própria solução), foram obtidos melhores resultados. Em 87% destes 42%, foi obtida uma melhor solução pelo método exato com um valor médio de melhoria de 4%. Nos restantes 13% dos 42% referidos, e que correspondem a uma instância com 151 vértices, o método exato não conseguiu sequer obter uma solução, ao contrário do algoritmo desenvolvido.

Por fim, com uma simulação de Monte Carlo para uma das instâncias foi possível comprovar a robustez para valores de Γ mais conservadores, bem como corroborar a ideia de que quanto maior o valor de Γ, maior o conservadorismo.

No futuro podem ser tomadas diversas medidas com o objetivo de melhorar o algoritmo. A primeira das quais poderia ser apenas a melhoria do código através da análise cuidada do mesmo. Isto porque, o algoritmo foi sendo adaptado por forma a passar por várias fases de validação. Além disso, conforme se foi tendo mais noção do problema, à medida que o desenvolvimento do algoritmo foi avançando, foi possível chegar à conclusão que alguns métodos poderiam ser mais eficientes se fossem destinados de origem ao algoritmo final.

Outra medida que se poderia ter, seria a adição de novos operadores através de diferentes heurísticas, tornando assim possível ao algoritmo criar novas regiões de pesquisa. Isto, além de permitir que a vizinhança pesquisada fosse maior, criaria maior diversificação aumentando, como consequência, a probabilidade do algoritmo ter à sua disposição mais heurísticas eficientes e adequadas a um maior leque de situações.

No algoritmo presente as distâncias entre vértices são simétricas, isto é, a distância do vértice *a* ao *b* é a mesma que do *b* ao *a*. Como consequência deste facto, uma rota *linehaul* terá o mesmo custo tanto numa determinada ordem como na ordem inversa. Portanto, no caso em que temos de planear rotas apenas para clientes a ordem é indiferente. O mesmo não se verifica quando além do planeamento de rotas para clientes também temos de planear rotas que incluem fornecedores, graças à imposição de visita a todos os clientes antes que seja visitado qualquer fornecedor. Isto porque quando se avalia a inserção de fornecedores tem de se ter em conta o cliente presente no fim da rota *linehaul*, e ter a mesma numa ordem ou na ordem inversa irá ter impacto na avaliação. Logo, seria interessante, na fase de avaliação da inserção do fornecedor, inverter a rota temporariamente e analisar ambas as ordens da rota *linehaul*, por forma a poder comparar as duas situações e decidir se em alguma das duas seria benéfica a inserção de um fornecedor.

Uma medida diferente que poderia ser tomada seria a alteração de alguns parâmetros do algoritmo. Por exemplo, com a finalidade de aumentar a diversificação, poderia ser utilizada uma maior quantidade de vértices a remover. Isto faria com que a solução sofresse alterações mais acentuadas, evitando que os pares de operadores destrutores/reparadores estivessem a realizar várias vezes as mesmas modificações na solução. Como consequência, seria avaliado um maior número de soluções distintas. Outros parâmetros que poderiam ser alterados seriam os valores atribuídos aos *scores* segundo a sua performance, dando valores mais altos do que os que foram dados aos pares de operadores com melhor performance, e mais baixos aos operadores com pior performance. Isto criaria uma discrepância maior entre *scores* e consequentemente seria esperado que fossem utilizados mais vezes os pares que realmente têm melhor performance. Expectavelmente esta medida traria melhores soluções.

O parâmetro presente em alguns operadores destrutores que controla a aleatoriedade de escolha do cliente a remover, poderia também ser alterado numa perspetiva de criar maior diversificação novamente. Mais uma vez, isto permitiria a pesquisa de novas vizinhanças.

Outro exemplo de um parâmetro a alterar, seria o parâmetro que controla a rapidez de ajuste dos pesos às performances obtidas pelos pares de operadores, isto é, se os pesos do segmento de iterações anterior definem completamente ou em parte os pesos a utilizar no segmento seguinte.

Adicionalmente, poderia ser utilizada uma forma diferente para que a robustez fosse incluída no algoritmo. Por exemplo a inclusão de robustez nas heurísticas que constituem os operadores. Depois de realizada esta alteração, seria interessante a comparação de resultados entre esta forma de inclusão de robustez e a forma utilizada nesta dissertação.

Finalmente, poderia ser criada uma interligação com a Google Maps API por forma a criar uma apresentação eficiente e clara das rotas da solução obtida. Assim, a apresentação poderia ser

feita numa página web ou até mesmo numa aplicação para telemóvel. Isto facilitaria a utilização da perspetiva do utilizador.

Bibliografia

Adulyasak Y, Jaillet P (2015) Models and algorithms for stochastic and robust vehicle routing with deadlines. Transportation Science 50(2):608–626

Agra A, Christiansen M, Figueiredo R, Hvattum LM, Poss M, Requejo C (2013) The robust vehicle routing problem with time windows. Computers & Operations Research 40(3):856–866

Akhtar M, Hannan M, Begum R, Basri H, Scavino E (2017) Backtracking search algorithm in CVRP models for efficient solid waste collection and route optimization. Waste Management

Amorim P, Marques A, Oliveira B (2014) Wood-based products distribution planning with wood returns in a iberian company. In: MOSIM 2014, 10ème Conférence Francophone de Modélisation, Optimisation et Simulation

Anily S (1996) The vehicle-routing problem with delivery and backhaul options. Naval Research Logistics 43(3):415–434

Audy J, D'Amours S, Rönnqvist M (2012) An empirical study on coalition formation and cost/savings allocation. International Journal of Production Economics 136(1):13–27

Beheshti Z, Shamsuddin S (2013) A review of population-based meta-heuristic algorithms. International Journal of Advances in Soft Computing and its Application 5(1):1–35

Bektas T, Repoussis P, Tarantilis C (2002) Dynamic Vehicle Routing Problem. In: The vehicle routing problem, SIAM monographs on discrete mathematics and applications, Society for Industrial and Applied Mathematics, Philadelphia, Pa

Ben-Tal A, Nemirovski A (1998) Robust convex optimization. Mathematics of Operations Research 23(4):769–805

Ben-Tal A, Nemirovski A (1999) Robust solutions of uncertain linear programs. Operations Research Letters 25(1):1–13

Ben-Tal A, Goryashko A, Guslitzer E, Nemirovski A (2004) Adjustable robust solutions of uncertain linear programs. Mathematical Programming 99(2):351–376

Ben-Tal A, El Ghaoui L, Nemirovski A (2009) Robust Optimization. Princeton University Press

Bertsimas D, Sim M (2004) The Price of Robustness. Operations Research 52(1):35–53

Bertsimas D, Pachamanova D, Sim M (2004) Robust linear optimization under general norms. Operations Research Letters 32(6):510–516

Bertsimas D, Brown D, Caramanis C (2011) Theory and Applications of Robust Optimization. SIAM Review 53(3):464–501

Birge J, Louveaux F (2011) Introduction to stochastic programming. Springer Science & Business Media

Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys (CSUR) 35(3):268–308

Braekers K, Ramaekers K, Van Nieuwenhuyse I (2016) The vehicle routing problem: State of the art classification and review. Computers & Industrial Engineering 99:300–313

Cao E, Lai M, Yang H (2014) Open vehicle routing problem with demand uncertainty and its robust strategies. Expert Systems with Applications 41(7):3569–3575

Caric T, Gold H (2008) Vehicle routing problem. Vienna, Austria

Cordeau J, Gendreau M, Hertz A, Laporte G, Sormany J (2005) New heuristics for the vehicle routing problem. In: Logistics systems: design and optimization, Springer, pp 279–297

Cuervo D, Goos P, Sörensen K, Arráiz E (2014) An iterated local search algorithm for the vehicle routing problem with backhauls. European Journal of Operational Research 237(2):454–464

Dantzig G, Ramser J (1959) The truck dispatching problem. Management Science 6:80–91

Duan Z, Sun S, Sun S, Li W (2015) Stochastic time-dependent vehicle routing problem: Mathematical models and ant colony algorithm. Advances in Mechanical Engineering 7(11):168781401561,863

Erera A, Morales J, Savelsbergh M (2010) The Vehicle Routing Problem with Stochastic Demand and Duration Constraints. Transportation Science 44(4):474–492

Fischetti M, Monaci M (2009) Light robustness. In: Robust and online large-scale optimization, Springer, pp 61–84

Forsberg M, Frisk M, Rönnqvisty M (2005) FlowOpt - a decision support tool for strategic and tactical transportation planning in forestry. International Journal of Forest Engineering 16(2):101–114

Gajpal Y, Abad P (2009) Multi-ant colony system (MACS) for a vehicle routing problem with backhauls. European Journal of Operational Research 196(1):102 – 117

Gendreau M, Potvin J (eds) (2010) Handbook of Metaheuristics, International Series in Operations Research & Management Science, vol 146. Springer US, Boston, MA

Gendreau M, Laporte G, Séguin R (1996) Stochastic vehicle routing. European Journal of Operational Research 88(1):3 – 12

Gendreau M, Jabali O, Rei W (2002a) Stochastic Vehicle Routing Problems. In: The vehicle routing problem, SIAM monographs on discrete mathematics and applications, Society for Industrial and Applied Mathematics, Philadelphia, Pa, pp 213 – 239

Gendreau M, Laporte G, Potvin J (2002b) Metaheuristics for the Capacited VRP. In: The vehicle routing problem, SIAM monographs on discrete mathematics and applications, Society for Industrial and Applied Mathematics, Philadelphia, Pa

Goetschalckx M, Jacobs-Blecha C (1989) The vehicle routing problem with backhauls. European Journal of Operational Research 42(1):39–51

Golden B, Raghavan S, Wasil E (eds) (2008) The vehicle routing problem: latest advances and new challenges. No. 43 in Operations research/Computer science interfaces series, Springer, New York, NY

Gounaris C, Wiesemann W, Floudas C (2013) The Robust Capacitated Vehicle Routing Problem Under Demand Uncertainty. Operations Research 61(3):677–693

Gromicho J, van Hoorn J, Kok A, Schutten J (2012) Restricted dynamic programming: A flexible framework for solving realistic VRPs. Computers & Operations Research 39(5):902–909

Gélinas S, Desrochers M, Desrosiers J, Solomon M (1995) A new branching strategy for time constrained routing problems with application to backhauling. Annals of Operations Research 61(1):91–109

Han J, Lee C, Park S (2014) A Robust Scenario Approach for the Vehicle Routing Problem with Uncertain Travel Times. Transportation Science 48(3):373–390

Juan A, Faulin J, Pérez-Bernabeu E, Jozefowiez N (2014) Horizontal Cooperation in Vehicle Routing Problems with Backhauling and Environmental Criteria. Procedia - Social and Behavioral Sciences 111:1133–1141

Koç C, Karaoglan I (2016) The green vehicle routing problem: A heuristic based exact solution approach. Applied Soft Computing 39:154–164

Küçükoğlu I, Öztürk N (2015) An advanced hybrid meta-heuristic algorithm for the vehicle routing problem with backhauls and time windows. Computers & Industrial Engineering 86:60 – 68

Laporte G, Gendreau M, Potvin J, Semet F (2000) Classical and modern heuristics for the vehicle routing problem. International Transactions in Operational Research 7(4-5):285–300

Lee C, Lee K, Park S (2012) Robust vehicle routing problem with deadlines and travel time/demand uncertainty. Journal of the Operational Research Society 63(9):1294–1306

Lenstra J, Kan A (1981) Complexity of vehicle routing and scheduling problems. Networks 11(2):221–227

Liebchen C, Lübbecke M, Möhring R, Stiller S (2009) The concept of recoverable robustness, linear programming recovery, and railway applications. In: Robust and online large-scale optimization, Springer, pp 1–27

Maggioni F, Potra F, Bertocchi M (2015) Stochastic versus robust optimization for a transportation problem. Optimization Online pp 03–4805

Manisri T, Mungwattana A, Janssens G (2011) Minimax optimisation approach for the Robust Vehicle Routing Problem with Time Windows and uncertain travel times. International Journal of Logistics Systems and Management 10(4):461–477

Mikiela D (2014) Vrp-rep: the vehicle routing problem repository. URL http://www.vrp-rep.org/datasets/item/2016-0005.html, [Accessed 10/05/17]

Moghadam B, Sadjadi S, Seyedhosseini S (2010) An empirical analysis on robust vehicle routing problem: a case study on drug industry. International Journal of Logistics Systems and Management 7(4):507–518

NEO Research Group (2013) Known Best Results | Vehicle Routing Problem. URL http://neo.lcc.uma.es/vrp/vrp-instances/, [Accessed 04/05/17]

Ordóñez F (2010) Robust Vehicle Routing. In: Hasenbein J, Gray P, Greenberg H (eds) Risk and Optimization in an Uncertain World, INFORMS, pp 153–178

Parragh S, Doerner K, Hartl R (2008a) A survey on pickup and delivery problems: Part I: Transportation between customers and depot. Journal für Betriebswirtschaft 58(1):21–51

Parragh S, Doerner K, Hartl R (2008b) A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations. Journal für Betriebswirtschaft 58(2):81–117

Potvin J, Rousseau J (1993) A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. European Journal of Operational Research 66(3):331–340

Ropke S, Pisinger D (2006) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40(4):455–472

Sahinidis N (2004) Optimization under uncertainty: state-of-the-art and opportunities. Computers & Chemical Engineering 28(6):971–983

Santos M, Amorim P, Marques A, Carvalho A, Póvoa A (2017) A survey on the vehicle routing problem with backhauls with emphasis in uncertainty and sustainability issues [Working paper]

Shaw P (1997) A new local search algorithm providing high quality solutions to vehicle routing problems. APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK

Shaw P (1998) Using constraint programming and local search methods to solve vehicle routing problems. In: International Conference on Principles and Practice of Constraint Programming, Springer, pp 417–431

Solano-Charris E, Prins C, Santos A (2014) A Robust optimization approach for the Vehicle Routing problem with uncertain travel cost. In: Control, Decision and Information Technologies (CoDIT), 2014 International Conference on, IEEE, pp 098–103

Souyris S, Cortés C, Ordóñez F, Weintraub A (2013) A robust optimization approach to dispatching technicians under stochastic service times. Optimization Letters 7(7):1549–1568

Sungur I, Ordóñez F, Dessouky M (2008) A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty. IIE Transactions 40(5):509–523

Toklu N (2014) Matheuristics for robust optimization. PhD thesis, Università della Svizzera italiana

Toklu N, Montemanni R, Gambardella L (2013) An ant colony system for the capacitated vehicle routing problem with uncertain travel costs. In: Swarm Intelligence (SIS), 2013 IEEE Symposium on, IEEE, pp 32–39

Toro O, Eliana M, Escobar Z, Antonio H, Granada E, et al (2016) Literature review on the vehicle routing problem in the green transportation context. Luna Azul pp 362–387

Toth P, Vigo D (1997) An exact algorithm for the vehicle routing problem with backhauls. Transportation Science 31(4):372–385

Toth P, Vigo D (1999) A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. European Journal of Operational Research 113(3):528–543

Toth P, Vigo D (2002a) Models, relaxations and exact approaches for the capacitated vehicle routing problem. Discrete Applied Mathematics 123(1–3):487 – 512

Toth P, Vigo D (eds) (2002b) The vehicle routing problem. SIAM monographs on discrete mathematics and applications, Society for Industrial and Applied Mathematics, Philadelphia, Pa

Union européenne, Commission européenne (2015) EU transport in figures 2016. Publications office of the European Union, Luxembourg

Wang Y, Zhang Y, Fuh J (2010) A genetic search algorithm for a vehicle routing problem with time windows. In: Manufacturing Automation (ICMA), 2010 International Conference on, IEEE, pp 172–177

Wassan N (2007) Reactive tabu adaptive memory programming search for the vehicle routing problem with backhauls. Journal of the Operational Research Society 58(12):1630–1641

Wassan N, Wassan N, Nagy G, Salhi S (2017) The multiple trip vehicle routing problem with backhauls: Formulation and a two-level variable neighbourhood search. Computers & Operations Research 78:454–467

Weintraub A, Epstein R, Morales R, Seron J, Traverso P (1996) A truck scheduling system improves efficiency in the forest industries. Interfaces 26(4):1–12

Yano C, Chan T, Richter L, Cutler T, Murty K, McGettigan D (1987) Vehicle Routing at Quality Stores. Interfaces 17:52–63

Zachariadis E, Kiranoudis C (2012) An effective local search approach for the Vehicle Routing Problem with Backhauls. Expert Systems with Applications 39(3):3174–3184