Micael Mateus Mesquita

Um problema de otimização de rotas com janelas temporais

Dissertação de Mestrado em Matemática, Área de Especialização em Estatística, Otimização e Matemática Financeira, orientada pelo Professora Doutora Marta Margarida Braz Pascoal e apresentada ao Departamento de Matemática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Janeiro de 2018

· U C ·

Universidade de Coimbra

Um problema de otimização de rotas com janelas temporais

Micael Mateus Mesquita

UNIVERSIDADE DE COIMBRA

Mestrado em Matemática

Master in Mathematics

Dissertação de Mestrado | MSc Dissertation

Agradecimentos

Quero agradecer à minha orientadora, pela disponibilidade, ajuda e sobretudo paciência ao longo do ano letivo.

Aos meus pais, por todo o apoio e carinho que me têm dado. Devo-lhes tudo! Não posso esquecer-me de todos os meus familiares, que mesmo não estando presentes fisicamente estão sempre comigo. Um grande obrigado a todos eles.

Ao Dinis Oliveira, António Pena, Gonçalo Firmo, Rúben Caetano, Diana Fonseca, Marisol Martins, Gonçalo Saraiva, André Martins e Diana Macedo, por me demonstrarem todos os dias que tenho bons amigos e pelo companheirismo, quase diário. Não poderia esquecer-me também da Silvie Santos, para além de ser uma pessoa incrível é também uma excelente amiga com quem eu posso contar e sempre disposta a dar umas gargalhadas. A ela o meu muito obrigado. Era impensável esquecer-me da pessoa que me recebeu em Coimbra. O meu "padrinho" André Oliveira, com quem eu passei muitos bons momentos durante este meu percurso e claro, quero continuar a passar. Além disso, está sempre disponível para uma boa conversa. Claro, falta a "madrinha" Inês Loureiro. Sim, não me esqueci de ti apesar de te ter conhecido mais tarde mas sei que és um pessoa incrível. Tenho os melhores "padrinhos" e amigos do mundo! Muito obrigado por fazerem parte da minha vida.

Queria agradecer à fabulosa equipa que me acolheu nestes últimos tempos. Sei que levo grandes amizades, tais como o Paulo Antunes, José Gonçalves, Pedro Monteiro e José Pina, entre outros. Obrigado a todos pelos bons momentos que se passaram e pelas belas gargalhadas que, mesmo em dias menos bons, conseguem alegrar qualquer um. Vocês são fenomenais! Um obrigado do fundo do coração e... *"fire walk with us"*.

Por fim, queria agradecer à Isabelle Caetano. Obrigado por ter tido a oportunidade de te conhecer, mas quero agradecer sobretudo pelos momentos que passamos, pelas conversas fabulosas que temos, pelos conselhos e sobretudo pelo apoio que me tens dado. És uma pessoa incrível! Não mereces apenas um obrigado, mas sim um obrigado do tamanho do universo.

Resumo

O problema de recolha de produtos (PRP) é um problema de otimização cujo objetivo é determinar a ordem pela qual visitar todos os vértices de um grafo, utilizando n_v veículos, minimizando a distância total a percorrer. Além disso, a visita a cada cliente deve realizar-se durante um intervalo de tempo definido *a priori* e a duração total de cada viagem não deve exceder um determinado limite. Neste trabalho o PRP é formulado como um programa linear inteiro misto. Essa formulação é exemplificada e, em seguida, são descritos alguns métodos heurísticos clássicos para este problema. Por fim, estuda-se o comportamento do PRP quando resolvido através da utilização de *software* para programação matemática e recorrendo a métodos heurísticos implementados, em problemas gerados de forma aleatória com várias dimensões[1].

Palavras chave: Programação linear inteira mista; Problema de recolha de produtos; Heurísticas.

[1] A imagem da capa foi retirada de: Wikimidea Commons, "Vehicle Routing Problem". Acedido a 30 de Dezembro de 2017. https://upload.wikimedia.org/wikipedia/commons/thumb/e/e0/Vehicle_Routing_Problem_Example.svg/492px-Vehicle_Routing_Problem_Example.svg.png.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O Problema de Otimização de Rotas de Veículos, ou simplesmente Problema de Rotas de Veículos (PRV), é um dos mais conhecidos e estudados problemas de otimização combinatória, em que se pretende calcular um conjunto ótimo de rotas para uma frota de veículos, por forma a servir um dado conjunto de clientes. O problema foi introduzido em finais dos anos 50, motivado por uma aplicação industrial concreta. Desde então tem sido alvo da atenção de muitos investigadores, que se têm dedicado não apenas ao problema orignal, mas também a variantes do problema que incluem novos tipos de restrições.

Uma das principais variantes do PRV é o Problema de Recolha de Produtos (PRP), em que cada cliente impõe um intervalo de tempo em que pode ser servido. Além das muitas aplicações, óbvias, também a sua dificuldade em termos computacionais tem desafiado muitos autores, que se têm dedicado, tanto a desenvolver algoritmos exatos para o resolver, como a desenvolver outros métodos, que permitem obter soluções admissíveis suficientemente próximas da ótima, em tempos computacionais compatíveis com muitas das aplicações.

O presente trabalho aborda o PRP e adapta algumas heurísticas clássicas para o PRV a este problema. O resto do texto está dividido em cinco outros capítulos. No segundo são apresentadas algumas noções sobre Teoria de Grafos, onde são explicados alguns conceitos importantes e úteis ao longo do trabalho. O terceiro capítulo tem por tema o Problema do Caixeiro Viajante, que é estudado e formulado matematicamente. No mesmo capítulo, são também apresentados alguns métodos que permitem resolver este problema. O quarto capítulo é dedicado ao PRP. O problema é enunciado, formulado e exemplificado. É também incluída uma restrição que limita o tempo de rota e são descritos alguns métodos que permitem resolver problemas relacionados, com especial atenção a métodos heurísticos. No quinto capítulo são apresentados resultados de testes feitos ao PRP, utilizando o CPLEX como ferramenta de resolução. São também descritos métodos heurísticos baseados nos conhecidos para o PRV e que foram implementados. Os resultados da aplicação dos métodos heurísticos a problemas gerados aleatoriamente são discutidos e comparados com as soluções encontradas utilizando o CPLEX e utilizando os métodos heurísticos. Por último, apresentam-se algumas conclusões.

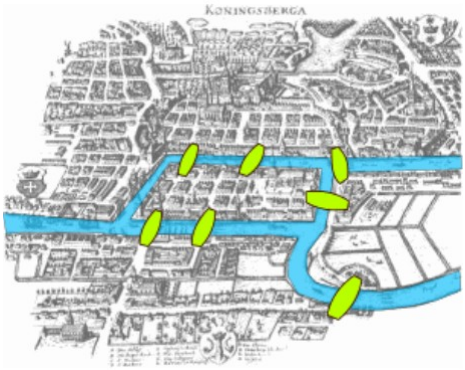
Capítulo 2

Breve introdução à Teoria de Grafos

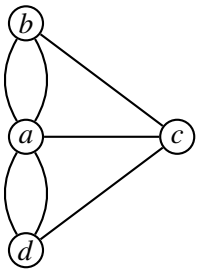
2.1 Introdução

A teoria de grafos foi introduzida em 1735 pelo matemático Leonhard Euler (1707-1783) ao tentar resolver o problema das sete pontes de Königsberg. Esta cidade, atualmente russa e chamada Kaliningrad, tinha sete pontes, seis das quais interligavam duas ilhas e as margens do rio Pregel e uma fazia a ligação entre as duas ilhas, como ilustra a Figura 2.1a. Durante muito tempo, os habitantes dessa cidade questionaram-se se seria possível percorrer todas as pontes uma única vez.

Em 1736, Euler apresentou uma solução para este problema, provando que não é possível fazer a travessia por todas as pontes uma e uma só vez. Para chegar a esta conclusão, foi necessário formular o problema de forma abstrata, utilizando para tal grafos.



(a) Representação da cidade (Fonte: https://upload.wikimedia.org/wikipedia/commons/5/5d/Konigsberg_bridges.png)



(b) Representação da cidade como um grafo

Figura 2.1 Problema das sete pontes de Königsberg

A Figura 2.1b representa a abstração introduzida por Euler. Aí os vértices denotam a massa terrestre (a, b, c e d) e as arestas que os ligam representam as sete pontes. Euler não só mostrou que não seria possível fazer o percurso uma única vez, como também estabeleceu a condição para que, num caso geral, tais travesssias pudessem ser feitas com sucesso. Nomeadamente, este percurso só existe quando o grafo tem zero ou exatamente dois vértices com grau ímpar, como se verá adiante.

Apesar dos avanços de Euler, só no início da segunda metade do Século XIX esta área começou a ser mais desenvolvida, com a resolução de problemas de fórmulas de estruturas químicas levada a cabo por Arthur Cayley.

No que se segue, formalizamos alguns conceitos de teoria de grafos mencionados acima e outros que serão utilizados no decorrer do trabalho.

2.2 Teoria de grafos

Nesta secção introduzem-se alguns conceitos e resultados relacionados com grafos.

Definição 2.2.1. *Um grafo é um par $G = (V,E)$, onde V (ou $V(G)$) é um conjunto de elementos, chamados vértices, e E (ou $E(G)$) é um conjunto de pares de vértices, chamados arestas.*

A Figura 2.1b representa graficamente um grafo. Nesse exemplo, os vértices são $V = \{a,b,c,d\}$ e as arestas $E = \{(ab)_1,(ab)_2,ac,(ad)_1,(ad)_2,bc,cd\}$. Note-se que mais do que uma aresta se liga aos vértices a e b, o mesmo se passando relativamente a a e d. Neste caso as arestas distinguem-se pelos índices, $(ab)_1$ e $(ab)_2$, assim como $(ad)_1$ e $(ad)_2$, e dizem-se paralelas. Note-se também que as arestas do grafo relacionam os dois vértices extremos, mas não impõem nenhuma ordem aos mesmos. Por esse motivo diz-se que grafos em que todas as arestas são deste tipo são grafos não orientados. No entanto, os grafos podem ter as suas arestas orientadas.

Definição 2.2.2. *Um grafo G diz-se orientado, ou um digrafo, se o conjunto E, das arestas, é constituido por pares ordenados de vértices. Neste caso, é habitual designar-se esse conjunto por A e as arestas por arcos.*

Não é obrigatório que a orientação de arcos seja nos dois sentidos. Por exemplo, dados os vértices a e b, pode existir um arco com início em a e fim em b, mas não existir um arco de b para a. As duas situações estão ilustradas nas Figuras 2.2a e 2.2b.

(a) Exemplo 1 (b) Exemplo 2

Figura 2.2 Grafos orientados

A partir de um grafo pode considerar-se apenas um seu subconjunto de vértices e um seu subconjunto de arestas. Ao par que daí resulta dá-se o nome de subgrafo.

Definição 2.2.3. *Seja $G = (V,E)$ um grafo. O par $G' = (V',E')$ é chamado de subgrafo de G se $E' \subseteq E$ e $V' \subseteq V$.*

Por exemplo, um possível subgrafo do grafo na Figura 2.1b é o da Figura 2.3. Neste caso $V' = V = \{a,b,c,d\}$ e $E' = \{(ab)_1,bc,ac,(ad)_1\} \subset E$.

Definição 2.2.4. *Seja $G = (V,E)$ um grafo. O grau do vértice $i \in V$, denotado $d(i)$, é o número de arestas que incidem nesse vértice.*

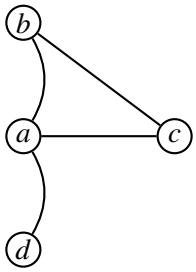


Figura 2.3 Exemplo de um subgrafo do grafo na Figura 2.1b

Por exemplo, no grafo da Figura 2.1b, incidem três arestas no vértice c, portanto $d(c) = 3$.

Define-se em seguida uma classe especial de grafos.

Definição 2.2.5. *Um grafo G diz-se completo se para todo o par de vértices existir uma aresta que os ligue. Estes grafos são denotados por K_n, onde n representa o seu número de vértices.*

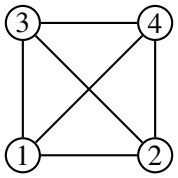


Figura 2.4 Grafo completo K_4

A Figura 2.4 mostra um exemplo de um grafo completo com quatro vértices.

Seja G um grafo e v_i, um vértice inicial, e v_f, um vértice final, dois quaisquer dos seus vértices. Um percurso de v_i para v_f em G é uma sequência alternada de vértices e arestas de G, que permite ligar v_i, o primeiro dos vértices, a v_f, o último. Estes trajetos podem ter algumas formas diferentes. Quando a sequência de vértices e arestas pode conter repetições de ambos, o percurso chama-se um *passeio*. A partir deste conceito podemos acrescentar restrições, obtendo assim outras formas de percurso. Caso nenhuma aresta se repita, mas podendo repetir-se os vértices, o percurso é chamado *trilho*. Se não houver qualquer repetição, nem de vértices nem de arestas, é chamado de *caminho*. Adicionalmente, se os vértices inicial e terminal de um caminho coincidirem, designa-se como caminho fechado (ou circuito), caso contrário diz-se aberto. Na Tabela 2.1 mostram-se alguns destes percursos, relativamente ao grafo da Figura 2.4. Utilizam-se os símbolos "$\langle$" e "$\rangle$" para delimitar o percurso e números para representar a visita dos vértices correspondentes. Além disso, omite-se a referência às arestas que ligam pares de vértices consecutivos, no pressuposto de não existirem arestas paralelas.

Tabela 2.1 Percursos no grafo da Figura 2.4

Passeio	$\langle 2,4,2,1\rangle$
Trilho	$\langle 1,3,4,3\rangle$
Caminho	$\langle 1,2,3,4\rangle$
Caminho fechado	$\langle 1,2,3,4,1\rangle$

Observando o grafo da Figura 2.4, verifica-se que se escolhermos quaisquer dois vértices existe sempre um caminho que os une.

Definição 2.2.6. *Um grafo G diz-se conexo se entre quaisquer dois vértices de G existir um caminho. Caso contrário, o grafo G diz-se desconexo.*

O grafo K_4 é conexo. A Figura 2.5 é um exemplo de um grafo desconexo. Por exemplo, não existe nenhum caminho que ligue os vértices 1 e 4.

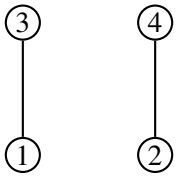
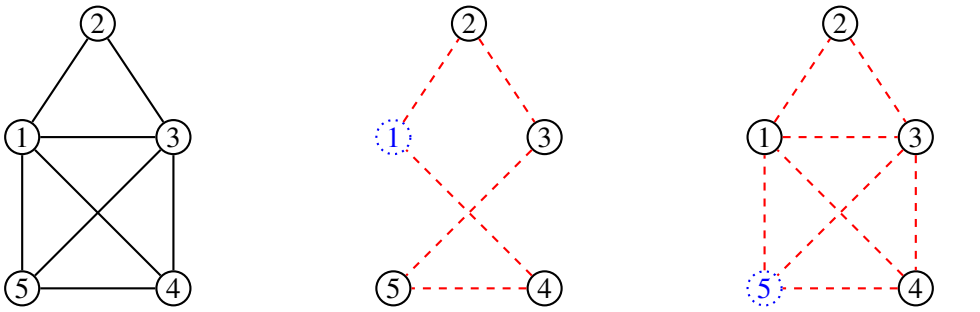


Figura 2.5 Grafo desconexo

Os grafos conexos podem ainda ser classificados como contendo um circuito Hamiltoniano ou um trilho Euleriano, conforme as definições a seguir.

Definição 2.2.7. *Um grafo conexo G é Hamiltoniano se existir um circuito que inclua todos os vértices de G. Este circuito é chamado circuito Hamiltoniano.*

O grafo na Figura 2.6a contém cinco vértices. Partindo do vértice 1 é possível encontrar um circuito conexo que passe por todos os vértices, sem os repetir, e regressa ao vértice inicial. Por exemplo, $\langle 1,2,3,5,4,1\rangle$, como ilustrado a vermelho (tracejado) na Figura 2.6b. Então, este circuito é Hamiltoniano e o grafo é, igualmente, Hamiltoniano.



(a) Grafo G (b) Circuito Hamiltoniano em G (c) Trilho Euleriano em G

Figura 2.6 Exemplo de um grafo com circuito Hamiltoniano e um trilho Euleriano

Definição 2.2.8. *Um grafo conexo G diz-se Euleriano se existir um trilho aberto que inclua todas as arestas de G. Este percurso é chamado trilho aberto Euleriano.*

A Figura 2.6c é um exemplo de um grafo Euleriano. Com efeito, $\langle 5,1,2,3,5,4,1,3,4\rangle$ é um trilho aberto que permite percorrer todas as arestas.

Nem todos os grafos contêm um trilho aberto (ou fechado) Euleriano, como é o caso do grafo da Figura 2.1b. Os dois resultados apresentados a seguir caracterizam os grafos que contêm este tipo de circuitos.

Teorema 2.2.1. *Um grafo conexo $G = (V,E)$ contém um trilho fechado Euleriano se e só se todos os vértices de G têm grau par.*

Demonstração. ($\Rightarrow$) Seja C um trilho Euleriano fechado de um grafo G. Cada vez que um vértice v ocorre no caminho C, há uma contribuição de duas unidades para o grau de v (uma aresta a chegar a v e outra a sair). Isto é válido não só para os vértices intermédios de C, mas também para o vértice final, uma vez que este é simultaneamente o vértice inicial e o final. Como cada aresta ocorre exatamente uma vez em C, cada vértice possui grau par.

($\Leftarrow$) Utilizemos agora indução sobre o número de arestas. Suponhamos que o grau de cada vértice de G é par. Como G é conexo, $d(v) \geq 2$, para todo $v \in V$, logo o grafo G contém um ciclo P. Se P contém todas as arestas de G, a demostração está concluída. Caso contrário, removemos de G as arestas do ciclo P, resultando um novo grafo, J, não necessariamente conexo mas com menos arestas do que G. Todos os vértices de J têm grau par, logo, pela hipótese de indução, cada componente conexa de J tem um trilho Euleriano fechado. Além disso, como G é conexo, cada componente de J tem pelo menos um vértice em comum com P, logo é possível construir um trilho fechado Euleriano em G. □

Como referido no início desta secção, e demonstrado por Euler, num grafo com no máximo dois vértices com grau ímpar existe um caminho aberto Euleriano.

Teorema 2.2.2. *Um grafo conexo contém um trilho aberto Euleriano se e só se existem exatamente dois vértices com grau ímpar, que serão os extremos do circuito.*

Demonstração. Sejam u e v os vértices de grau ímpar e acrescentemos ao grafo a aresta uv. Então os vértices ficam todos com grau par e, pelo Teorema 2.2.1, existe um trilho fechado de Euler, o qual passa pela aresta uv. Se a esse trilho fechado retirarmos a aresta uv, obtemos o trilho aberto desejado. □

É possível obter um caminho, ou um circuito, de Euler a partir de um grafo conexo qualquer, aplicando o algoritmo de Fleury [9]. A ideia geral deste método é escolher um vértice inicial qualquer e, em seguida, escolher outro ao qual o primeiro esteja ligado por uma aresta. Este passo é repetido até esgotar as arestas do grafo. Caso incida mais do que uma aresta no vértice mais recentemente acrescentado ao circuito, é dada a preferência a uma cuja remoção não ponha em causa a conexidade do grafo. Este algoritmo implica a análise de todas as arestas do grafo, pelo que tem um número de operações de $\mathcal{O}(m)$.

Outro tipo de grafos especiais são as árvores.

Definição 2.2.9. *Uma árvore é um grafo conexo e sem ciclos. Quando o grafo é constituído por várias componentes conexas, todas elas árvores, então diz-se uma floresta.*

Definição 2.2.10. *Uma árvore T que seja subgrafo de um grafo G diz-se geradora de G se contém todos os vértices de G.*

Na Figura 2.7 representa-se uma árvore, obtida a partir do grafo K_4 da Figura 2.4. Esta árvore é também uma árvore geradora de K_4, isto é, uma árvore que abrange todos os vértices de K_4.

Defina-se em seguida o conceito de rede, de grande utilidade em problemas de otimização, uma vez que permite modelar bem uma variedade de objetivos.

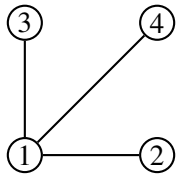


Figura 2.7 Árvore obtida a partir de K_4

Definição 2.2.11. *Uma rede é um grafo a cujas arestas (ou vértices) estão associados valores númericos, denominados pesos (ou custos, distâncias ou ainda tempos).*

A Figura 2.8 é um exemplo de uma rede, uma vez que as arestas são afetadas a pesos.

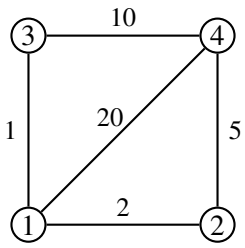
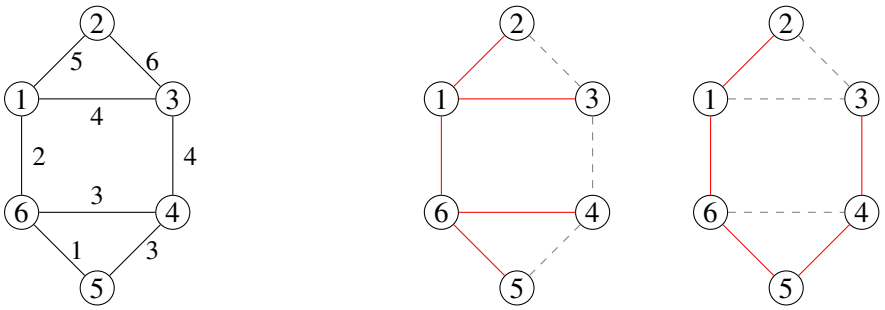


Figura 2.8 Rede G

Suponhamos que $G = (V,E)$ é uma rede e que $d_{ij} \in \mathbb{R}$ é o peso associado a qualquer aresta $ij \in E$. Então pode definir-se peso de uma árvore geradora de G como sendo a soma dos pesos da suas arestas. O objetivo do problema da árvore geradora com peso mínimo, ou árvore geradora mínima, é determinar uma árvore geradora de G que tenha peso mínimo relativamente a todas as árvores geradoras de G. Naturalmente este problema é admissível se e só se o grafo tiver pelo menos uma árvore geradora, ou seja, se e só se o grafo for conexo.

Suponhamos dada a rede representada na Figura 2.9a. A Figura 2.9b ilustra duas árvores geradoras nesta rede, representada pelas linhas vermelhas (sólidas). Ambas têm peso 15 e são árvores geradoras com peso mínimo.



(a) Rede G (b) Duas árvores geradoras de G com peso mínimo

Figura 2.9 Árvores geradoras mínimas

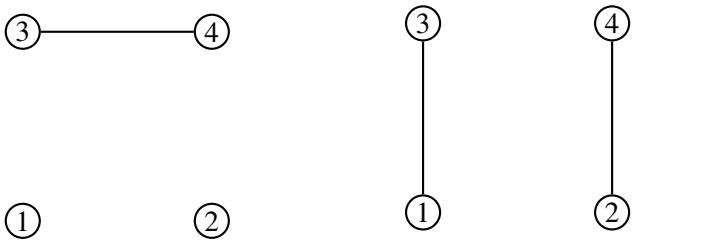
A árvore geradora com peso mínimo de um grafo conexo pode ser calculada em tempo polinomial, por exemplo pelos algoritmos de Kruskal [12] ou de Prim [20]. Ambos são métodos gulosos[1] em

[1] Do inglês, *greedy algorithm*.

que as arestas a incluir na solução são escolhidas por ordem (crescente) de peso, sob determinadas condições. O algoritmo de Kruskal tem tempo de $\mathcal{O}(m \log n)$, enquanto que o algoritmo de Prim [20] resolve o mesmo problema em tempo de $\mathcal{O}(n^2)$.

Definição 2.2.12. *Seja $G = (V,E)$ um grafo. Um subgrafo M de G é chamado emparelhamento se em cada vértice $u \in V$ incide no máximo uma aresta de M, ou seja, se em M $d(u) \leq 1$. O emparelhamento M diz-se um emparelhamento perfeito se qualquer que seja o vértice u em M existe uma aresta de M que incide em M, isto é, se em M se tem $d(u) = 1$.*

Seja $G = (V,E)$ uma rede com pesos $d_{ij} \in \mathbb{R}$ associados às arestas $ij \in E$. O peso de um emparelhamento é definido pela soma dos pesos das suas arestas. Então, o objetivo do problema do emparelhamento perfeito com peso mínimo é determinar um emparelhamento perfeito de G que minimize o peso. Este emparelhamento pode ser determinado através do algoritmo de Edmonds [8], que tem tempo de $\mathcal{O}(mn)$, que no caso de um grafo completo corresponde a $\mathcal{O}(n^3)$.



(a) Emparelhamento

(b) Emparelhamento perfeito com peso mínimo

Figura 2.10 Emparelhamentos na rede da Figura 2.8

A Figura 2.10 mostra dois emparelhamentos na rede da Figura 2.8. O emparelhamento na Figura 2.10a não é perfeito, uma vez que nenhuma aresta incide nos vértices 1 e 2. O emparelhamento na Figura 2.10b é um emparelhamento perfeito e também um emparelhamento perfeito com peso mínimo, tendo peso 6. De facto, existe apenas outro emparelhamento perfeito nessa rede, constituído pelas arestas $\{\{1,2\},\{3,4\}\}$, que tem peso 12.

Capítulo 3

O Problema do Caixeiro Viajante

3.1 Introdução

São conhecidas referências a problemas relacionados com o problema do caixeiro viajante (PCV) desde final do século XIX. Contudo, sua autoria parece permanecer incerta [1]. Os primeiros registos de uma formulação matemática do PCV datam da década de 1930, no entanto, o problema só começou a ser reconhecido como PCV, e a ganhar algum crédito, a partir de meados do século passado. Desde então o PCV tornou-se um dos problemas mais emblemáticos na área de Otimização Combinatória.

A ideia base do PCV é que um vendedor pretende visitar um conjunto de clientes uma única vez, e que deseja fazê-lo de modo a gastar o menos possível. Além da aplicação mais imediata a problemas de otimização de rotas, o PCV tem sido utilizado em inúmeros outros contextos, tais como o desenho de sistemas digitais, ou o agendamento e sequenciação de atividades, entre outros. No entanto, até à data presente ainda não se conseguiu desenvolver um algoritmo eficiente, capaz de encontrar a solução ótima de um qualquer destes problemas. Isto porque o PCV possui uma elevada ordem de complexidade computacional, dizendo-se um problema $\mathscr{NP}-$difícil. No entanto, existem métodos exatos e aproximados que permitem resolver este problema. Os exatos permitem encontrar uma solução ótima, mas podem demorar muito tempo, mesmo para problemas com dimensões limitadas, como por exemplo o método *branch-and-bound* ou a programação dinâmica. Já os métodos aproximados, ou heurísticos, são, em geral, bastante rápidos e podem fornecer uma resposta relativamente próxima da ótima para o problema.

Neste capítulo apresenta-se formalmente o PCV e alguns métodos heurísticos conhecidos para o resolver.

3.2 Formulação matemática

Seja $G = (V,E)$ um grafo completo, sem perda de generalidade, onde $V = \{1,2,\ldots,n\}$ é o conjunto de vértices e com $m = |E|$ arestas. Seja ainda $D \in \mathbb{R}^{n\times n}$ uma matriz, cujas componentes d_{ij} representam os custos associados à deslocação do vértice i ao vértice j. Em muitos casos, tais custos podem corresponder a distâncias ou tempos. No caso do PCV, $d_{ij} \in \mathbb{R}_0^+$ representa a distância a percorrer para visitar o vértice j após visitar o vértice i. O objetivo do PCV é determinar um circuito que passe por todos os vértices uma única vez, voltando ao inicial, e que tenha distância total mínima. De acordo

com o capítulo anterior, este circuito diz-se Hamiltoneano, mas pode também denominar-se de rota. Para modelar um problema quando G não for um grafo completo, é suficiente tomar-se o custo das arestas em falta como um valor arbitrariamente grande.

Para formular o PCV, consideremos as variáveis binárias

$$x_{ij} = \begin{cases} 1, & \text{se na solução o vértice } i \text{ e o vértice } j \text{ estão ligados diretamente} \\ 0, & \text{caso contrário} \end{cases}, \quad \forall i, j \in V.$$

Utilizando estas variáveis podemos definir a função objetivo do PCV, que deve representar o custo total das arestas escolhidas para o circuito,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$$

e que se pretende minimizar. Como restrições do problema é necessário considerar

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \in V, \tag{3.1}$$

para garantir que exatamente uma aresta abandona cada vértice i e

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j \in V, \tag{3.2}$$

para garantir que exatamente uma aresta chega a cada vértice j.

As restrições (3.1) e (3.2) são restrições de afetação dos vértices, pelo que permitem definir os vértices que antecedem e sucedem cada um na solução. Soluções que satisfaçam estas restrições podem incluir vários circuitos desconexos, usualmente denominados sub-rotas. A Figura 3.1 apresenta um exemplo desta situação, recordando-se que no PCV se procura um único circuito que visite todos os vértices. Para evitar soluções como a ilustrada, é necessário acrescentar as restrições

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \quad \forall S \subset V. \tag{3.3}$$

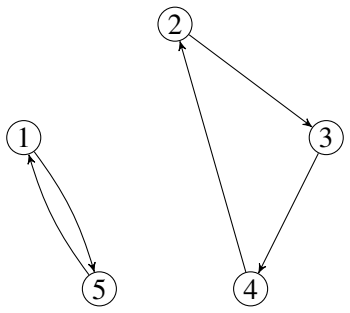


Figura 3.1 Exemplo de sub-rotas

Resumindo, o PCV pode ser escrito da forma seguinte,

$$\begin{aligned} \text{minimizar} \quad & \sum_{j=1}^{n}\sum_{i=1}^{n} d_{ij}x_{ij} && (3.4)\\ \text{sujeito a} \quad & (3.1)-(3.3) \\ & x_{ij} \in \{0,1\}, \quad \forall i,j \in V && (3.5) \end{aligned}$$

Esta formulação é um programa linear com n^2 variáveis binárias e um número exponencial de restrições, uma vez que (3.3) gera uma restrição para cada subconjunto de V.

Dependendo de características adicionais do problema, podem definir-se diferentes tipos de PCV. Por exemplo, quando $d_{ij} = d_{ji}, \forall i,j \in V$, o PCV diz-se simétrico. Caso contrário, o problema diz-se assimétrico. Além disso, um PCV diz-se métrico se verifica a desigualdade triangular,

$$d_{ij} + d_{jk} \geq d_{ik}, \quad \forall i,j,k \in V.$$

Por fim, o PCV euclidiano caracteriza-se pelo facto de os vértices a visitar corresponderem a pontos no plano $\mathbb{R}^2$ e d_{ij} representar a distância euclidiana entre o vértice i e o vértice j, isto é,

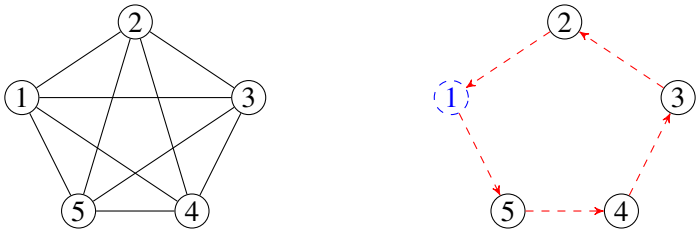
$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \quad \forall i,j \in V,$$

se os vértices i,j correspondem aos pontos $(x_i,y_i),(x_j,y_j) \in \mathbb{R}^2$, respetivamente. Naturalmente, qualquer PCV euclidiano é, também, um PCV métrico e simétrico.

Para exemplificar o PCV, consideremos um grafo completo com cinco vértices dispostos no plano, como indicado na Figura 3.2a, e como custos a matriz de distâncias euclidianas entre eles,

$$D = \begin{bmatrix} 0 & 2.5 & 4 & 3.6 & 2.2 \\ 2.5 & 0 & 2.5 & 3.6 & 3.6 \\ 4 & 2.5 & 0 & 2.2 & 3.6 \\ 3.6 & 3.6 & 2.2 & 0 & 2 \\ 2.2 & 3.6 & 3.6 & 2 & 0 \end{bmatrix}$$

Partindo do vértice 1 e viajando sempre pela aresta com a menor distância, obtém-se o resultado da Figura 3.2b. O circuito Hamiltoniano assim calculado é $\langle 1,5,4,3,2,1 \rangle$ e tem uma distância total mínima, concretamente de 11.4.



(a) Grafo completo K_5 (b) Solução do PCV

Figura 3.2 Exemplo de uma rota ótima em K_5

O modelo para o PCV apresentado acima não é único, pelo que em seguida descrevemos um modelo alternativo, igualmente linear, mas apenas com um número polinomial de restrições. Este modelo deve-se a Miller, Tucker e Zemlin [16], que introduziram um novo conjunto de variáveis de decisão do problema. Qualquer que seja o vértice $i \in V$, considere-se a variável $u_i \in \mathbb{N}$, que representa a ordem em que o vértice i é visitado na solução.

Suponhamos que uma aresta ij pertence à solução, para $i, j \in V$. Isto significa que os vértices i e j ocupam posições consecutivas na rota, portanto

$$u_i = u_j - 1.$$

Se, pelo contrário, ij não pertence à solução e supondo, sem perda de generalidade, que $u_i < u_j$, então

$$u_i \leq u_j - 1 \leq u_j + n - 1.$$

Pode, assim, definir-se

$$u_i - u_j + nx_{ij} \leq n - 1, \quad \forall i, j \in V. \tag{3.6}$$

Por outro lado, como a solução deve incluir todos os vértices, tem-se

$$1 \leq u_i \leq n, \quad \forall i \in V. \tag{3.7}$$

Desta forma, garante-se igualmente a inexistência de sub-rotas. O modelo assim obtido reúne as restrições (3.1) a (3.7),

$$\begin{aligned}
\text{minimizar} \quad & \sum_{j=1}^{n} \sum_{i=1}^{n} d_{ij} x_{ij} \\
\text{sujeito a} \quad & \sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \in V \\
& \sum_{i=1}^{n} x_{ij} = 1, \quad \forall j \in V \\
& u_i - u_j + nx_{ij} \leq n - 1, \quad \forall i, j \in V \\
& 1 \leq u_i \leq n, \quad \forall i \in V \\
& x_{ij} \in \{0, 1\}, \quad \forall i, j \in V \\
& u_i \in \mathbb{Z}, \quad \forall i \in V
\end{aligned} \tag{3.8}$$

Esta formulação continua a ser linear e contém mais n variáveis, inteiras do que a anterior, no total $n^2 + n$. No entanto, o seu número de restrições propriamente ditas é apenas de $\mathcal{O}\left(n^2\right)$.

3.3 Métodos heurísticos

São conhecidos na literatura vários métodos para resolver o PCV, entre os quais, heurísticas construtivas e de melhoria. Como referido no início do capítulo, estes métodos encontram uma solução admissível do problema, que não é necessariamente a ótima, mas fazem-nos num tempo computa-

cional polinomial. Nesta secção apresentam-se alguns destes métodos, tais como os algoritmos da árvore dupla, de Christofides e do vizinho mais próximo. Descrevem-se ainda heurísticas de melhoria, usadas para melhorar a solução obtida pelos métodos construtivos, tal como a k-opt. Começa-se por introduzir uma medida de desempenho de algoritmos de aproximação, em termos da qualidade das soluções que encontram.

Definição 3.3.1. *Sejam P um problema de minimização e $k \geq 1$. Seja ainda A um algoritmo para P, com tempo polinomial. Dada uma concretização I do problema P, denote-se por $OPT(I)$ o seu valor ótimo e por $A(I)$ o valor objetivo obtido através do algoritmo A. A é um algoritmo de aproximação com rácio de desempenho k, ou um algoritmo de k-aproximação, para P, se*

$$A(I) \leq k\, OPT(I)$$

para qualquer concretização I de P. Diz-se então que o algoritmo A tem uma garantia, ou um rácio, de desempenho k.

De seguida apresentam-se alguns métodos construtivos para resolver o PCV.

Algoritmo da Árvore Dupla O método da árvore dupla foi proposto por Rosenkrantz, Stearns e Lewis [21] e aplica-se ao PCV métrico. O algoritmo é constituído por vários passos. Inicialmente é calculada uma árvore geradora mínima do grafo, por exemplo aplicando os algoritmos de Kruskal ou de Prim, o que permite ligar todos os vértices do grafo.

O segundo passo consiste em duplicar as arestas da árvore geradora determinada. Isto garante que se obtém um novo grafo cujos vértices têm todos grau par, e, de acordo com o Teorema 2.2.1, a existência de um trilho fechado Euleriano.

Em seguida determina-se um trilho fechado Euleriano no novo grafo, o que pode ser realizado através do algoritmo de Fleury [9].

O trilho fechado Euleriano calculado inclui todos os vértices do grafo, mas alguns podem ser repetidos. Então, por último, é necessário eliminar as arestas redundantes, ou seja, aquelas que incidem em vértices com grau superior a dois. Isto é conseguido substituindo sequências de arestas por "atalhos"[1], mais concretamente, analisando sequencialmente os vértices do trilho fechado, se $j \in V$ for um desses vértices e $ij, jk \in E$ duas dessas arestas, substitui-se este par de arestas por $ik \in E$. A desigualdade triangular garante que $d_{ij} + d_{jk} \geq d_{ik}$, pelo que a distância da nova rota não excede a da anterior. O método da árvore dupla é resumido no Algoritmo 1.

Algorithm 1: Algoritmo da árvore dupla

$T \leftarrow$ árvore geradora mínima em G
$E \leftarrow E'$ tal que E' contém duas vezes as arestas de E
$C \leftarrow$ trilho fechado Euleriano em (V, E')
$C' \leftarrow$ circuito Hamiltoniano obtido a partir de C, passando uma única vez por cada vértice pela aresta mais barata

[1] Do inglês *shortcuts*.

O número de operações requerido por este processo depende do número de operações realizadas pelos quatro passos do método. São precisas $\mathcal{O}(n^2)$ operações para o cálculo da árvore geradora mínima se é utilizado o algoritmo de Prim, $\mathcal{O}(m)$ para duplicar as arestas, $\mathcal{O}(m)$ para formar um trilho fechado Euleriano e $\mathcal{O}(n^2)$ para incluir "atalhos", isto é, eliminar arestas rendundantes. Portanto, o tempo deste método tem ordem de complexidade de $\mathcal{O}(n^2)$. O rácio de desempenho do Algoritmo 1 é $k = 2$, como se mostra em seguida.

Teorema 3.3.1. *O Algoritmo 1 tem um rácio de desempenho $k = 2$.*

Demonstração. Seja I uma concretização do PCV métrico. Como T é uma árvore geradora mínima e retirando uma aresta a qualquer circuito Hamiltoniano se obtém uma árvore geradora, temos que

$$d(E(T)) \leq OPT(I).$$

Adicionalmente, sendo C e C' o trilho fechado Euleriano e circuito Hamiltoniano obtidos pelo Algoritmo 1, pela desigualdade triangular tem-se $d(E(C')) \leq d(E(C))$. Por outro lado, $d(E(C)) = 2\,d(E(T))$, donde

$$d\left(E\left(C'\right)\right) \leq 2\,OPT(I)$$

como pretendido. □

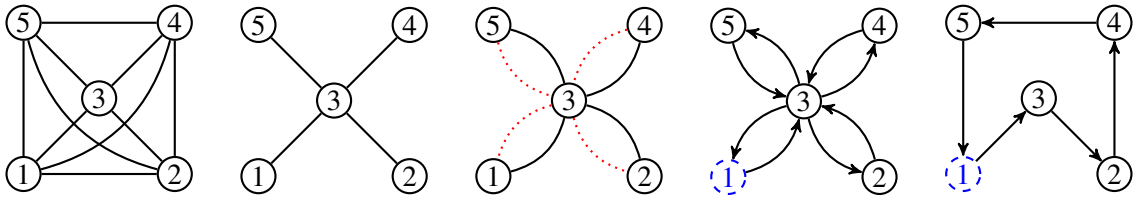
De forma a ilustrar o Algoritmo 1, consideremos um grafo completo tendo por conjunto de vértices cinco pontos de $\mathbb{R}^2$, dispostos como na Figura 3.3a, e as distâncias entre cada par de vértices dadas pelas distâncias euclidianas na matriz seguinte

$$D = \begin{bmatrix} 0 & 4 & 2.8 & 5.7 & 4 \\ 4 & 0 & 2.8 & 4 & 5.7 \\ 2.8 & 2.8 & 0 & 2.8 & 2.8 \\ 5.7 & 4 & 2.8 & 0 & 4 \\ 4 & 5.7 & 2.8 & 4 & 0 \end{bmatrix}$$

Inicialmente calcula-se a árvore geradora mínima representada na Figura 3.3b. No segundo passo do algoritmo duplicam-se as arestas da árvore encontrada, como ilustrado pelas arestas a vermelho (ponteado) na Figura 3.3c. De seguida é criado um trilho fechado Euleriano no novo grafo, por exemplo, partindo do vértice 1 (a azul e tracejado na Figura 3.3d) forma-se o trilho fechado $C = \langle 1,3,2,3,4,3,5,3,1\rangle$. Visto haver várias visitas ao vértice 3, eliminam-se essas repetições, formando assim a rota final $R_f = \langle 1,3,2,4,5,1\rangle$. Esta rota, representada na Figura 3.3e, tem uma distância total de 17.6.

Algoritmo de Christofides Tal como o algoritmo da árvore dupla, o algoritmo de Christofides [4] aplica-se ao PCV métrico e começa por calcular uma árvore geradora mínima do grafo inicial.

Em seguida, para transformar a árvore geradora mínima num grafo Euleriano, consideram-se apenas os vértices que têm grau ímpar nessa árvore e calcula-se um emparelhamento perfeito com custo mínimo destes vértices. Isto permite aumentar em uma unidade o grau dos vértices com grau ímpar e, portanto, garante que a união da árvore e do emparelhamento é um grafo Euleriano.



(a) Grafo K_5 (b) Árvore geradora mínima T (c) Árvore geradora duplicada $2T$ (d) Circuito Euleriano C (e) Rota final C'

Figura 3.3 Algoritmo da árvore dupla

Os passos seguintes são novamente semelhantes ao método da árvore dupla. Primeiro calcula-se um trilho fechado de Euler e, em seguida, eliminam-se arestas com extremos em vértices e que incidem mais do que duas arestas. O resultado é resumido no Algoritmo 2.

Algorithm 2: Algoritmo de Christofides

$T \leftarrow$ árvore geradora mínima em G
$M \leftarrow$ emparelhamento perfeito com custo mínimo no subgrafo G, induzido pelo conjunto de vértices com grau ímpar em T
$E' \leftarrow E(T) \cup E(M)$
$C \leftarrow$ trilho fechado Euleriano em (V, E')
$C' \leftarrow$ circuito Hamiltoniano obtido a partir de C, passando uma única vez por cada vértice

Em termos de tempo a ordem de complexidade deste método é análoga à do método da árvore dupla, exceto na duplicação de arestas da árvore geradora, que agora é substituída pelo cálculo de um emparelhamento perfeito com custo mínimo. Este emparelhamento pode ser determinado através do algoritmo de Edmonds, que tem tempo de $\mathcal{O}(mn)$. Conclui-se então que o Algoritmo 2 tem tempo de $\mathcal{O}(mn)$, portanto para grafos completos tem $\mathcal{O}(n^3)$. Relativamente ao desempenho, este método tem rácio $k = \frac{3}{2}$.

Teorema 3.3.2. *O algoritmo de Christofides tem um rácio de desempenho de* $\frac{3}{2}$.

Demonstração. Seja I uma concretização do PCV métrico. Tal como na demonstração do Teorema 3.3.1,

$$d(E(T)) \leq OPT(I).$$

Seja C^* uma rota ótima e C^{**} um circuito que passa nos vértices com grau ímpar de T, pela mesma ordem que em C^*. Pela desigualdade triangular, $d(C^{**}) \leq d(C^*)$. Além disso, o número de vértices com grau ímpar em T é par, donde C^{**} pode ser decomposto em dois emparelhamentos perfeitos, M_1 e M_2, e $d(C^{**}) = d(M_1) + d(M_2)$. Sendo M um emparelhamento máximo com custo mínimo, $d(M) \leq d(M_1)$ e $d(M) \leq d(M_2)$, donde

$$2d(M) \leq d(M_1) + d(M_2) = d(C^{**}) \leq d(C^*),$$

isto é,

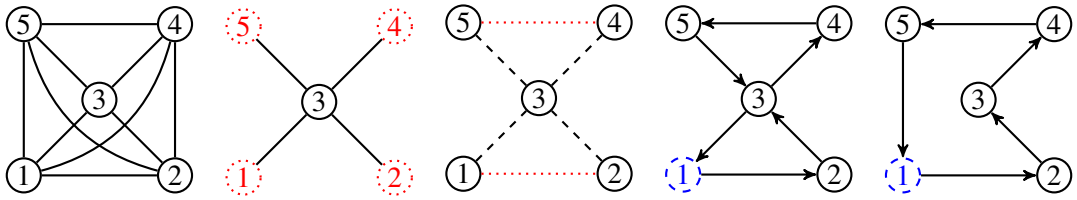
$$d(M) \leq \frac{1}{2} OPT(I).$$

Então,

$$d\left(E\left(C^{*}\right)\right) = d\left(E\left(T\right)\right) + d\left(M\right) \leq OPT\left(I\right) + \frac{1}{2}\,OPT\left(I\right) = \frac{3}{2}\,OPT\left(I\right),$$

como se pretendia mostrar. □

Para ilustrar o método de Christofides, consideremos novamente o exemplo anterior. Tal como referido, o passo inicial deste algoritmo é igual ao do método da árvore dupla, surgindo a diferença na fase de cálculo de um emparelhamento perfeito com custo mínimo relativamente ao conjunto dos vértices da árvore que têm grau ímpar. Tais vértices são mostrados a ponteado (vermelho) na Figura 3.4b e o emparelhamento procurado é constituído pelas arestas a ponteado (vermelho) na Figura 3.4c. De seguida, a árvore com custo mínimo e o emparelhamento são reunidos para determinar um trilho fechado Euleriano, tal como o que tem início no vértice 1, a tracejado (azul) na Figura 3.4d, percorrendo assim todas as arestas disponíveis, $C = \langle 1,2,3,4,5,3,1\rangle$. Neste circuito o vértice 3 é visitado mais do que uma vez, pelo que o último passo do método elimina estas repetições, obtendo-se a rota $R_f = \langle 1,2,3,4,5,1\rangle$. A rota R_f está representada na Figura 3.4e e tem uma distância total de 17.6.



(a) Grafo K_5 (b) Árvore geradora mínima T (c) Emparelhamento perfeito de custo mínimo M (d) Circuito Euleriano C (e) Rota final C'

Figura 3.4 Algoritmo de Christofides

Algoritmo do vizinho mais próximo Tal como os dois métodos anteriores, o algoritmo do vizinho mais próximo[2] [17] é um método construtivo, que permite obter uma solução do PCV. A chave para esta abordagem é visitar sempre o vértice mais próximo do mais recente, desde que não tenha sido visitado previamente, de forma a procurar uma rota de distância mínima. O método pode ser aplicado a qualquer tipo de PCV, no entanto não é possível garantir um majorante do quociente entre o valor objetivo de uma qualquer solução obtida com este método e o valor ótimo. O método pode ser traduzido nos passos no Algoritmo 3, onde o símbolo "$\circ$" denota a concatenação de dois passeios.

A complexidade do número de operações do algoritmo do vizinho mais próximo é de $\mathscr{O}(m)$, uma vez que a análise de cada vértice x implica a análise de todas as arestas que incidem em x. No pior dos casos o grafo é completo e a complexidade é de $\mathscr{O}(n^2)$.

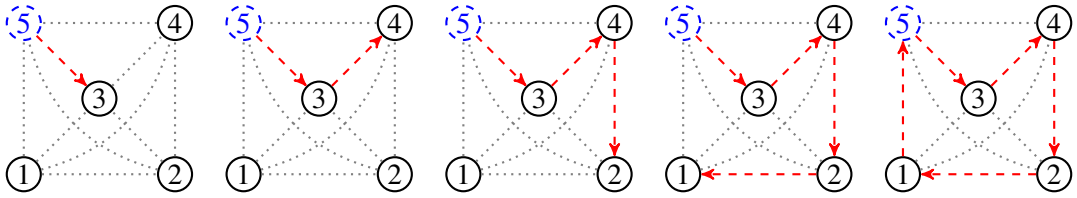
De forma a exemplificar este método, tomemos o exemplo usado no algoritmo da árvore dupla com cinco vértices e exatamente as mesmas distâncias. Como sugerido pela sequência de passos acima descrita, começa-se então pelo vértice 5. Daqui procura-se o vizinho mais próximo e que se liga ao vértice inicial. Seguindo este raciocínio, repetem-se as visitas sem formar circuitos, até

[2] Do inglês, *nearest neighbour*.

Algorithm 3: Algoritmo do vizinho mais próximo

$x \leftarrow$ vértice em V
$C \leftarrow \langle x \rangle$; $V \leftarrow V - \{x\}$
$w \leftarrow x$
while $V \neq \emptyset$ **do**
  $e \leftarrow \text{argmin}\{d_e : e \in E \wedge e \equiv xy\}$
  $C \leftarrow C \circ \langle y \rangle$; $V \leftarrow V - \{y\}$
  $x \leftarrow y$
$C \leftarrow C \circ \langle w \rangle$

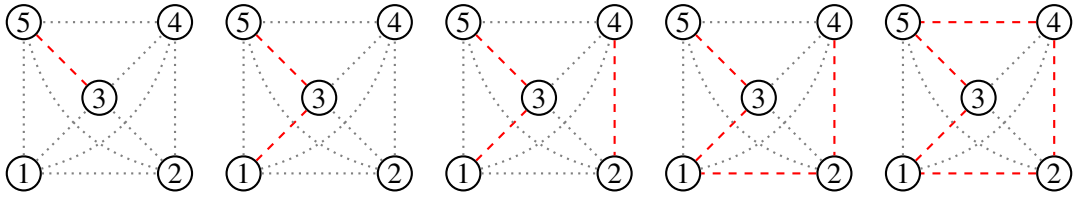
chegar ao ponto de partida. Obtém-se assim a rota $R_f = \langle 5,3,4,2,1,5 \rangle$, com uma distância total de 17.6. A Figura 3.5 representa as iterações que são feitas pelo algoritmo, onde o vértice inicial está representado a azul (tracejado) e as arestas sucessivamente selecionadas representadas a vermelho (tracejado).



(a) Iteração 1 (b) Iteração 2 (c) Iteração 3 (d) Iteração 4 (e) Iteração 5

Figura 3.5 Algoritmo do vizinho mais próximo

Algoritmo da aresta mais curta Outro método guloso para obter rotas admissíveis para o PCV baseia-se na escolha de arestas, por ordem crescente de custo, desde que não criem circuitos com menos de n arestas, não aumentem o grau de nenhum vértice para mais do que dois e não repita a mesma aresta duas vezes. Este método é conhecido por algoritmo da aresta mais curta [17]. Em termos de tempo a complexidade associada a este processo é $\mathscr{O}(n^2 \log n)$. No entanto, tal como no caso anterior, não é conhecido um rácio de desempenho constante para esta abordagem. As várias iterações do método são ilustradas na Figura 3.6. Neste exemplo a rota final não coincide com as encontradas pelas abordagens anteriores, apesar de ser a mesma instância.



(a) Iteração 1 (b) Iteração 2 (c) Iteração 3 (d) Iteração 4 (e) Iteração 5

Figura 3.6 Algoritmo da aresta mais curta

Heurísticas de melhoria As heurísticas de melhoria recebem como *input* uma solução admissível do PCV, obtida por exemplo através de um dos métodos anteriores, e procuram melhorá-la em termos de custo, embora continuem sem garantir a otimalidade da solução encontrada.

Um dos grupos de heurísticas de melhoria mais conhecido é o das heurísticas k-opt, que se baseiam em trocar k arestas da solução dada por outras k arestas, desde que isso melhore o valor da função objetivo [14]. O valor k pode variar e um circuito que não possa ser melhorado através da heurística k-opt, pode sê-lo para a heurística $(k+1)$-opt. Por exemplo, consideremos a rede completa formada pelos vértices na Figura 3.7 e a distância euclidiana entre eles. A solução na Figura 3.7a é ótima relativamente à heurísticas 2-opt, isto é, não pode ser melhorada pela troca de sequências com 2 arestas. No entanto, considerando trocas de 3 arestas pode obter-se o circuito na Figura 3.7b, onde as arestas indicadas a vermelho (tracejado) substituem as indicadas a cinzento (ponteado).

(a) Circuito Hamiltoniano 2-opt (b) Circuito Hamiltoniano 3-opt

Figura 3.7 Exemplo das heurísticas 2-opt e 3-opt

Chandra, Karloff e Tovey [3] introduziram resultados pouco animadores sobre o rácio de desempenho deste tipo de métodos, tendo mostrado que a rota produzida pela heurística k-opt pode ter uma distância $\frac{1}{4}n^{1/2k}$ vezes superior à distância ótima. Em termos de tempo computacional estas heurísticas têm um comportamento teórico exponencial, dado que exigem a comparação de combinações de k arestas. Contudo, na prática têm-se revelado computacionalmente eficientes para valores pequenos de k, como $k = 2,3$. Quando $k \geq 4$, as comparações entre dois conjuntos de k arestas começa a requerer um esforço computacional elevado, que o melhoramento em termos de valor da função objetivo não compensa.

Na Figura 3.8, está exemplicada uma modificação de uma 4-opt, também conhecido por movimento da ponte dupla, onde se retiram as arestas $\{\{2,7\},\{3,6\},\{1,4\},\{8,5\}\}$ (tracejado) e se inserem as arestas $\{\{2,3\},\{4,5\},\{6,7\},\{8,1\}\}$ (a vermelho). Uma generalização de 2-opt é a heurística de Lin e Kerninghan [15], onde é feita a divisão da solução em duas partes disjuntas, calculando a forma mais rentável para unir as duas soluções.

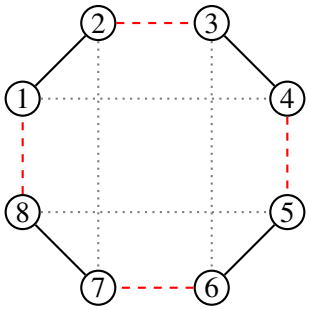


Figura 3.8 Exemplo de uma 4-Opt.

Capítulo 4

O Problema de Recolha de Produtos

4.1 Introdução

O PCV, abordado no capítulo anterior, pode ser estendido aumentando a frota de veículos disponíveis para mais do que um. O objetivo passa, assim, a determinar várias rotas, em vez de uma única, que obriguem a percorrer uma distância total mínima e garantam que todos os vértices são visitados por algum dos veículos. Problemas deste tipo dividem-se em duas classes: o problema do caixeiro viajante múltiplo (PCVM) e o problema do estabelecimento de rotas de veículos (PRV)[1]. Ambos podem incluir restrições de vários tipos além das de base, dependendo da aplicação a que se destinam, mas, de acordo com Bektas [2], a diferença é que no segundo se considera que os veículos utilizados têm capacidade limitada, enquanto que para o PCVM esta restrição não é relevante. Apesar disso a maioria dos trabalhos na literatura utilizam técnicas semelhantes para tratar os dois problemas.

Naturalmente, tratando-se de extensões do PCV, tanto o PCVM como o PRV são problemas $\mathscr{NP}$-difíceis [10]. Em termos práticos, isto traduz-se diretamente no tempo necessário para obter soluções ótimas de problemas com média, ou grande, dimensão. Ainda assim as duas classes de problemas tem inegável utilidade, razão pela qual têm sido profundamente estudadas há mais de 50 anos.

Apesar da dificuldade em obter uma solução ótima para PCVMs e para PRVs com grandes dimensões, têm sido desenvolvidos vários métodos para este efeito. Os métodos exatos conhecidos empregam técnicas variadas, como relaxação lagrangeana, decomposição de Dantzig-Wolfe, geração de colunas ou métodos *branch-and-bound*, entre outros. Paralelamente, têm sido propostos métodos heurísticos, que calculam soluções aproximadas daqueles problemas. Os trabalhos de Bektas [2] e de Toth e Vigo [24] apresentam revisões de literatura sobre o PCVM e o PRV.

O presente capítulo é dedicado a uma variante do PCVM e do PRV, a que chamaremos problema da recolha de produtos (PRP). Tal como o nome sugere, pretende-se efetuar a recolha de produtos em vários clientes, conhecidos *a priori*. O objetivo do PRP é comum ao PCVM e ao PRV. Adicionalmente, é conhecido um intervalo de tempo, ou seja, uma janela temporal, para cada cliente, durante o qual a recolha deve ter lugar. Por último, a duração de cada rota não deve exceder um determinado limite, imposto à partida. A carga transportada por cada veículo não está condicionada, pelo que este é um PCVM.

[1] No original *m-traveling salesmen problem* e *vehicle routing problem*, respetivamente.

A secção seguinte é dedicada à introdução de uma formulação para o PRP, juntamente com a apresentação de um exemplo. Em seguida apresenta-se uma revisão de alguns métodos heurísticos para o PCVM e o PRV, do tipo construtivo e do tipo de melhoria, que poderão ser adaptados à resolução do PRP.

4.2 Formulação matemática

Suponhamos que uma empresa pretende recolher produtos em n clientes de uma determinada localidade e que dispõe de n_v veículos para esse efeito. Os clientes estabelecem um período para efetuar o serviço. Pretende-se encontrar rotas que respeitem os horários definidos pelos clientes e que minimizem a distância total percorrida pelos n_v veículos.

Para formalizar o problema, consideremos um grafo $G = (V,A)$ em que $V = \{1,2,\ldots,n\}$ é o conjunto de vértices, que representam os clientes, $A \subseteq V \times V$ é o conjunto de arestas que definem as ligações entre esses vértices. Sejam ainda $N_v = \{1,2,\ldots,n_v\}$ o conjunto dos veículos da empresa e $V_0 = V \cup \{0\}$ o conjunto de todos os vértices, incluindo o depósito, ou seja, o ponto de partida/chegada, que se considera ser a empresa. Este ponto é representado pelo vértice adicional 0.

Como dados do problema, suponhamos conhecidos os valores d_{ij}, que representam a distância entre o par de clientes/empresa i e j, para $i, j \in V_0$.

Como variáveis de decisão, consideremos

$$x_{ijk} = \begin{cases} 1 & \text{se o veículo } k \text{ vai diretamente do cliente } i \text{ para o cliente } j \\ 0 & \text{caso contrário} \end{cases}$$

para $i, j \in V_0$ e $k \in N_v$, e

$$y_{ik} = \begin{cases} 1 & \text{se o veículo } k \text{ visita o cliente } i \\ 0 & \text{caso contrário} \end{cases}$$

para $i \in V_0$ e $k \in N_v$.

No PRP pretende-se minimizar a distância total percorrida por todos os veículos, pelo que a função objetivo é definida como

$$\sum_{k \in N_v} \sum_{i \in V_0} \sum_{\substack{j \in V_0 \\ i \neq j}} d_{ij} x_{ijk} \tag{4.1}$$

Como restrições é necessário impôr que todos os clientes sejam visitados por um único veículo

$$\sum_{k \in N_v} y_{ik} = 1, \quad \forall i \in V \tag{4.2}$$

e que todas as rotas comecem e terminem no depósito

$$\sum_{k \in N_v} y_{0k} = n_v. \tag{4.3}$$

Além disso, é preciso definir a rota para cada veículo k. Para tal basta relacionar as variáveis de decisão x_{ijk} e y_{ik}, como se segue

$$\sum_{i \in V} x_{ijk} = y_{jk}, \quad \forall j \in V_0, \quad \forall k \in N_v \tag{4.4}$$

$$\sum_{j \in V} x_{ijk} = y_{ik}, \quad \forall i \in V_0, \quad \forall k \in N_v. \tag{4.5}$$

Por fim, é necessário garantir que as soluções não contêm sub-rotas, isto é, que todas as rotas passam pelo depósito, o que pode ser conseguido através das restrições

$$\sum_{i \in S} \sum_{\substack{j \in S \\ i \neq j}} x_{ijk} \geq r(S), \quad \forall S \subseteq V_0, S \neq \emptyset \quad \forall k \in N_v. \tag{4.6}$$

onde $r(S)$ é o número mínimo de veículos necessários para servir o subconjunto de vértices S [13]. Então, uma primeira formulação do PRP é o seguinte programa linear inteiro (mais concretamente binário)

$$\begin{aligned} \text{minimizar} \quad & \sum_{k \in N_v} \sum_{i \in V_0} \sum_{\substack{j \in V_0 \\ i \neq j}} d_{ij} x_{ijk} \\ \text{sujeito a} \quad & (4.2) - (4.6) \\ & x_{ijk} \in \{0,1\}, \quad \forall i, j \in V_0, \quad \forall k \in N_v \qquad (4.7) \\ & y_{ik} \in \{0,1\}, \quad \forall i \in V_0, \quad \forall k \in N_v \qquad (4.8) \end{aligned}$$

É de notar que é possível separar o depósito em dois vértices, 0 e $n+1$, representando o ponto de partida e o ponto de chegada, respetivamente. Para tal, basta introduzir um novo conjunto de restrições análogas a (4.3) e ajustar adequadamente as restrições (4.4) e (4.5).

Cada cliente deve ser visitado durante o horário que especificou. Então, por forma a modelar janelas temporais e a incluí-las na formulação anterior, consideremos ainda conhecidos:

- t_{ij}, que denota o tempo de viagem do cliente i até ao cliente j,
- t_s, o tempo de serviço no cliente i, e
- $[a_i, b_i]$, que representa o intervalo de tempo que cada cliente tem disponível para o serviço a efetuar.

Como variáveis de decisão adicionais do problema consideramos $t_i^k \geq 0$, que representam o instante de chegada do veículo k ao cliente i. Para garantir que os tempos de chegada a cada cliente são compatíveis entre dois clientes consecutivos acrescentamos as restrições

$$x_{ijk}\left(t_i^k + t_s + t_{ij} - t_j^k\right) \leq 0 \quad \forall k \in N_v \quad \forall i, j \in V, \quad i \neq j \tag{4.9}$$

Além disso, para garantir as janelas temporais definidas pelos clientes, impomos

$$a_i \leq t_i^k \leq b_i, \quad \forall i \in V, \quad \forall k \in N_v. \tag{4.10}$$

As restrições (4.9) garantem a inexistência de sub-rotas e, portanto, dispensam a inclusão das condições (4.6) [7]. Então, o PCVM com janelas temporais pode ser formulado como

$$\begin{aligned} \text{minimizar} \quad & \sum_{k\in N_v}\sum_{i\in V_0}\sum_{\substack{j\in V_0\\ i\neq j}} d_{ij}x_{ijk} \\ \text{sujeito a} \quad & (4.2)-(4.5),(4.7)-(4.10) \end{aligned}$$

As restrições (4.9) não são lineares. No entanto, podem ser substituídas por um conjunto de restrições lineares, como se mostra em seguida. O maior valor que $t_i^k + t_s + t_{ij} - t_j^k$ assume quando t_i^k é máximo e t_j^k é mínimo, ou seja, para $t_i^k = b_i$ e $t_j^k = a_j$, e portanto, definindo R_{ij} como

$$R_{ij} = b_i + t_{ij} + t_s - a_j, \quad i,j \in V_0,$$

garantimos que

$$R_{ij} \geq t_i^k + t_s + t_{ij} - t_j^k.$$

Então, a restrição (4.9) pode ser substituída por

$$t_j^k \geq t_i^k + t_s + t_{ij} - \left(1 - x_{ijk}\right) R_{ij}, \quad i,j \in V_0. \tag{4.11}$$

De facto, se $x_{ijk} = 0$ esta condição transforma-se em

$$t_j^k \geq t_i^k + t_s + t_{ij} - R_{ij},$$

que é sempre satisfeita, atendendo à definição R_{ij}, tal como acontece em (4.9). Caso contrário, se $x_{ijk} = 1$, então a condição (4.11) implica

$$t_j^k \geq t_i^k + t_s + t_{ij},$$

que coincide com (4.9). Em conclusão, o PCVM com janelas temporais pode ser formulado como o seguinte programa linear inteiro misto

$$\begin{aligned} \text{minimizar} \quad & \sum_{k\in N_v}\sum_{i\in V_0}\sum_{\substack{j\in V_0\\ i\neq j}} d_{ij}x_{ijk} \\ \text{sujeito a} \quad & (4.2)-(4.5) \\ & (4.7)-(4.8),(4.10)-(4.11) \end{aligned}$$

Apresentamos agora um exemplo concreto da aplicação do PCVM com janelas temporais. Consideremos que a empresa tem três veículos e oito clientes, e que todas as recolhas são efetuadas em horário de expediente. Suponhamos que a distribuição dos clientes em termos geográficos corresponde à apresentada na Figura 4.1 e que os horários propostos pelos clientes para visita e recolha de produtos são os apresentados na Tabela 4.1. Suponhamos também que o tempo de serviço é de 15 minutos para todos os clientes.

Tabela 4.1 Horários dos clientes, $[a_i, b_i]$

i	1	2	3	4	5	6	7	8
a_i	8:00	12:00	14:00	10:30	16:30	12:30	15:30	15:00
b_i	9:00	13:00	14:30	11:30	17:30	16:00	16:00	15:30

As distâncias (em quilómetros) entre clientes, ou entre um cliente e a empresa, correspondem à distância euclidiana entre os pontos indicados na Figura 4.1 e são dadas pela matriz $D = [d_{ij}]_{i,j \in V_0}$, com elementos

$$D = \begin{bmatrix}
0.000 & 11.180 & 9.220 & 9.055 & 9.487 & 6.083 & 4.472 & 6.403 & 7.280 \\
11.180 & 0.000 & 3.162 & 6.083 & 8.062 & 7.211 & 9.220 & 11.662 & 14.422 \\
9.220 & 3.162 & 0.000 & 3.000 & 5.000 & 4.243 & 6.403 & 8.602 & 11.402 \\
9.055 & 6.083 & 3.000 & 0.000 & 2.000 & 3.000 & 5.099 & 6.403 & 9.220 \\
9.487 & 8.062 & 5.000 & 2.000 & 0.000 & 3.606 & 5.099 & 5.385 & 8.062 \\
6.083 & 7.211 & 4.243 & 3.000 & 3.606 & 0.000 & 2.236 & 4.472 & 7.211 \\
4.472 & 9.220 & 6.403 & 5.099 & 5.099 & 2.236 & 0.000 & 3.000 & 5.385 \\
6.403 & 11.662 & 8.602 & 6.403 & 5.385 & 4.472 & 3.000 & 0.000 & 2.828 \\
7.280 & 14.422 & 11.402 & 9.220 & 8.062 & 7.211 & 5.385 & 2.828 & 0.000
\end{bmatrix}$$

Note-se que a primeira linha e a primeira coluna da matriz contêm informação relativa ao depósito, enquanto que as restantes posições respeitam aos clientes da empresa. Admitamos que os veículos se deslocam a uma velocidade constante de 35 quilómetros/hora, pelo que os tempos (em horas) correspondentes à matriz D são dados por $T = [t_{ij}]_{i,j \in V_0}$, com $t_{ij} = d_{ij}/35$. A distribuição ótima dos clientes pelos veículos é representada na Figura 4.1. A Tabela 4.2 apresenta os tempos de chegada a cada vértice.

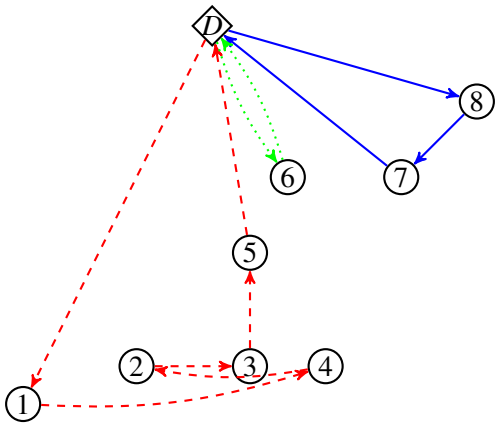


Figura 4.1 Solução do problema

Na solução ótima, o primeiro veículo, representado a azul (linha sólida), está associado à rota $v_1 = \langle D, 8, 7, D \rangle$. O segundo veículo, a vermelho (tracejado), efetuou a rota $v_2 = \langle D, 1, 4, 2, 3, 5, D \rangle$. O terceiro, a verde (ponteado), efetuou o percurso $v_3 = \langle D, 6, D \rangle$. Em termos da distância percorrida, o percurso total foi de 61.78 quilómetros. A duração total de cada rota, incluindo o tempo de espera

em cada vértice, é apresentada na Tabela 4.3. Na Tabela 4.2 são apresentados os horários de chegada a cada cliente.

Tabela 4.2 Horários de chegada, t_i^k

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$k=1$							15:30	15:00
$k=2$	8:00	12:00	14:15	10:30	16:30			
$k=3$						13:45		

Tabela 4.3 Duração total das rotas na Figura 4.1

Rota	Duração
$k=1$	1:08
$k=2$	9:15
$k=3$	0:30

Neste exemplo o tempo máximo de viagem propriamente dita e de serviço nos clientes de cada veículo ronda as 2 horas e 15 minutos, No entanto, para respeitar as janelas temporais, os tempos de espera em cada cliente fazem com que a duração da viagem mais longa ultrapasse as 9 horas, conforme indicado na Tabela 4.3. No que se segue pretende-se limitar o tempo total para efetuar cada rota. Para esse efeito, começa por se dividir o vértice que representa o depósito em dois, 0 e $n+1$, o primeiro representando o ponto de partida para todos os veículos e o último representando o seu ponto de chegada. Reajustemos então as restrições e a formulação do problema ao novo conjunto de vértices.

Consideremos que $V_{0,n+1} = V \cup \{0, n+1\}$ denota o conjunto de todos os clientes, reunido com os vértices que definem o depósito. Denotemos também por $V_0 = V \cup \{0\}$ o conjunto de todos os clientes e o ponto de partida, com o índice 0, e $V_{n+1} = V \cup \{n+1\}$ o conjunto dos clientes e o ponto de chegada, com o índice $n+1$. Seja também T_M o tempo limite máximo para cada rota.

Para considerar esta nova numeração dos vértices e modelar o limite do tempo de cada rota começa por se adaptar a função objetivo e as restrições definidas anteriormente. Relativamente à função objetivo tem-se

$$\sum_{k\in N_v}\sum_{i\in V}\sum_{\substack{j\in V\\ i\neq j}} d_{ij}x_{ijk} + \sum_{k\in N_v}\sum_{j\in V} d_{0j}x_{0jk} + \sum_{k\in N_v}\sum_{i\in V} d_{i,n+1}x_{i,n+1,k}$$

Para obrigar a que todos os veículos saiam do depósito, 0, e cheguem ao depósito, $n+1$, e que exatamente um deles passe em cada vértice, acrescenta-se

$$\sum_{k\in N_v} y_{ik} = 1, \quad \forall i \in V$$

$$\sum_{k\in N_v} y_{0k} = \sum_{k\in N_v} y_{n+1,k} = n_v$$

Para associar uma sequência de vértices a cada veículo, considera-se

$$\sum_{j\in V} x_{ijk} = y_{ik}, \quad \forall i \in V_0, \quad \forall k \in N_v$$
$$\sum_{i\in V} x_{ijk} = y_{jk}, \quad \forall j \in V_{n+1}, \quad \forall k \in N_v$$

enquanto que as janelas temporais são definidas por

$$a_i \leq t_i^k \leq b_i, \quad \forall i \in V, \quad \forall k \in N_v$$
$$t_j^k \geq t_i^k + t_s + t_{ij} - (1 - x_{ijk})R_{ij}, \quad i \in V_0, \quad j \in V_{n+1}$$

Com estas alterações é possível calcular os tempos de chegada dados por t_j^k, para cada $j \in V_{n+1}$, e, portanto, as restrições

$$t_{n+1}^k - t_{0i} - t_i^k \leq T_M, \quad i \in V, \quad k \in N_v$$

implicam que o tempo de rota do veículo k não ultrapasse o limite T_M, previamente estabelecido. Resumindo, o PRP pode ser escrito na forma,

$$\begin{aligned}
\text{minimizar} \quad & \sum_{k\in N_v}\sum_{i\in V}\sum_{\substack{j\in V\\ i\neq j}} d_{ij}x_{ijk} + \sum_{k\in N_v}\sum_{j\in V} d_{0j}x_{0jk} + \sum_{k\in N_v}\sum_{i\in V} d_{i,n+1}x_{i,n+1,k} \\
\text{sujeito a} \quad & \sum_{k\in N_v} y_{ik} = 1, \quad \forall i \in V \\
& \sum_{k\in N_v} y_{0k} = \sum_{k\in N_v} y_{n+1,k} = n_v \\
& \sum_{j\in V} x_{ijk} = y_{ik}, \quad \forall i \in V_0, \quad \forall k \in N_v \\
& \sum_{i\in V} x_{ijk} = y_{jk}, \quad \forall j \in V_{n+1}, \quad \forall k \in N_v \\
& a_i \leq t_i^k \leq b_i, \quad \forall i \in V, \quad \forall k \in N_v \\
& t_j^k \geq t_i^k + t_s + t_{ij} - (1 - x_{ijk})R_{ij}, \quad i \in V_0, \quad j \in V_{n+1} \\
& t_{n+1}^k - t_{0i} - t_i^k \leq T_M, \quad i \in V, \quad k \in N_v \\
& x_{ijk} \in \{0,1\}, \quad \forall i \in V_0, \quad j \in V_{n+1}, \quad k \in N_v \\
& y_{ik} \in \{0,1\}, \quad \forall i \in V_0, \quad k \in N_v
\end{aligned} \tag{4.12}$$

Para ilustrar esta modificação na prática, utilizemos novamente o exemplo anterior, agora condicionando o tempo todas as rotas a um máximo, T_M, de 2 horas. Após a sua resolução verificou-se que este problema dá inadmissível para apenas $n_v = 3$ veículos e, portanto, considerou-se $n_v = 4$. Na Figura 4.2 é apresentada a solução ótima.

Na solução ótima, o primeiro veículo, a azul (linha sólida), visita os clientes $v_1 = \langle 0,6,3,9\rangle$; o segundo veículo, a vermelho (tracejado), visita os clientes $v_2 = \langle 0,4,2,9\rangle$; o terceiro veículo, a verde (ponteado), $v_3 = \langle 0,8,7,5,9\rangle$ e o quarto veículo, a preto (traço e ponto), $v_4 = \langle 0,1,9\rangle$. Na Tabela 4.4 são apresentados os tempos de chegada a cada vértice na solução representada na Figura 4.2, enquanto que a Tabela 4.5 mostra a duração da rota realizada por cada veículo. Daqui se conclui que todas terminam passadas menos de 2 horas, como pretendido. Em termos de distância percorrida, o percurso total percorrido pelos veículos são de 85.356 quilómetros.

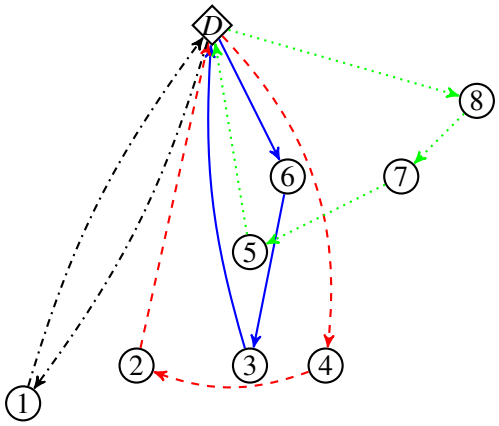


Figura 4.2 Solução do problema

Tabela 4.4 Horários de chegada, t_i^k

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$k=1$			14:24			14:00		
$k=2$		12:00		11:02				
$k=3$					16:30		15:43	15:23
$k=4$	9:00							

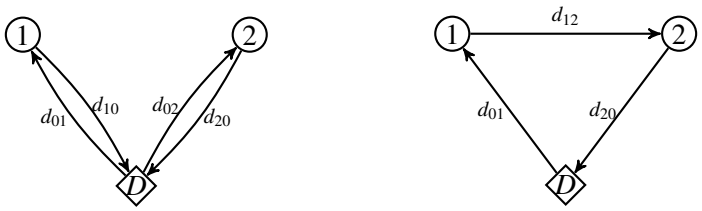
Tabela 4.5 Duração total das rotas na Figura 4.2

Rota	Duração
$k=1$	1:02
$k=2$	1:11
$k=3$	1:20
$k=4$	0:53

4.3 Métodos heurísticos construtivos

As heurísticas construtivas têm por objetivo determinar soluções admissíveis. No caso de problemas que envolvem várias rotas isto pode ser conseguido de forma sequencial ou paralela. A construção sequencial forma uma rota de cada vez, procurando não afetar veículos adicionais, a menos que haja solicitações que não sejam compatíveis com o veículo atual. A construção paralela cria várias rotas em simultâneo, partindo de um número fixo de veículos a usar. Posteriormente é possível adicionar mais veículos, se tal for necessário, sem violar as restrições do problema. Em seguida, apresentamos os métodos de Clarke e Wright, de inserção, do vizinho mais próximo em tempo orientado e varrimento.

Heurística de poupança de Clarke e Wright A ideia do método de poupança, proposto por Clarke e Wright [5], é começar por atribuir um veículo a cada cliente. Assim, a rota inicial de cada veículo consiste numa viagem do depósito para o cliente e em seguida no sentido contrário, o que permite obter uma solução admissível do problema. Na segunda fase procura-se melhorar a solução anterior, sem comprometer a sua admissibilidade.



(a) Rotas individuais (b) Fusão das rotas

Figura 4.3 Heurísticas de poupança

A Figura 4.3 mostra um exemplo, para um problema com duas rotas. São dadas duas rotas separadas (como na Figura 4.3a), uma que serve o cliente 1 e outra que serve o cliente 2. A distância total das rotas separadas é dada por

$$D_{rs} = d_{10} + d_{01} + d_{20} + d_{02}.$$

No entanto, as duas podem ser combinadas de forma a servir os dois clientes como representado na Figura 4.3b. Neste caso a distância total das rotas unidas é dada por

$$D_{ru} = d_{01} + d_{12} + d_{20}.$$

Então, ao fundir estas rotas, a alteração na distância percorrida é dada por

$$P_{12} = D_{rs} - D_{ru} = d_{10} + d_{02} - d_{12} \tag{4.13}$$

No método de Clarke e Wright a alteração das rotas é efetuada, desde que signifique uma poupança relativamente à distância, isto é, se $P_{12} > 0$. Solomon [23] propôs alterações a este método para ser aplicado ao PCVM, adicionando a orientação temporal do percurso. Isto é, para além de ter em conta o fator da poupança é necessário que os clientes sejam atendidos por ordem temporal crescente. Por exemplo, no caso da Figura 4.3a se o cliente 1 tiver um limite temporal inferior ao do cliente 2 e $P_{12} > 0$ então o resultado obtido seria como o da Figura 4.3b. Caso contrário, o cliente 2 teria prioridade no atendimento.

Heurística do vizinho mais próximo orientada pelo tempo Este método é um processo de construção sequencial do tipo guloso, com inspiração semelhante à da heurística do vizinho mais próximo descrita no Capítulo 3 para o PCV. Tal como esse método, esta heurística começa por inicializar a rota atual tendo o depósito como único vértice. Em seguida, o cliente mais próximo do depósito é inserido na rota como sendo o próximo cliente a visitar. Posteriormente, o processo repete-se para o cliente mais próximo do último visitado e é inserido na rota a seguir a esse, caso mantenha a admissibilidade na solução, isto é, se esta inserção não violar nenhuma restrição. Se nenhum dos restantes clientes ainda sem rota for admissível, é atribuída uma nova rota e o processo é repetido. Solomon [23] alargou este método para o caso em que existem janelas temporais, definindo a "proximidade" entre vértices como uma combinação que relaciona três fatores, a distância geográfica, d_{ij}, a distância em termos de tempo, T_{ij}, e também a "urgência" em visitar o próximo cliente, u_{ij}. Então a proximidade entre um

cliente i e um cliente j é dada por

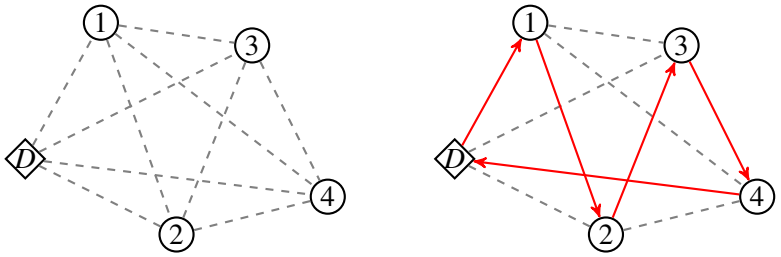
$$\begin{cases} \Gamma_{ij} = \delta_1 d_{ij} + \delta_2 T_{ij} + \delta_3 u_{ij} \\ T_{ij} = t_j^k - (t_i^k + t_s) \\ u_{ij} = b_j - (t_i^k + t_s + t_{ij}) \\ t_j^k = \max\{a_j, t_i^k + t_s + t_{ij}\} \end{cases} \tag{4.14}$$

onde $\delta_1, \delta_2, \delta_3 \in [0,1]$, com $\delta_1 + \delta_2 + \delta_3 = 1$, são pesos associados a cada um dos fatores e t_i^k é o tempo de chegada ao cliente i na rota k.

Para exemplicar este método, suponhamos dado um conjunto de três clientes, com intervalos temporais para visita $[a_i, b_i]$ e um veículo disponível para fazer o serviço pretendido. Na Figura 4.4a representa-se uma possível distribuição geográfica dos clientes e uma rota final determinada pelo algoritmo. Esta rota é obtida após o cálculo de Γ_{ij}, definidos na equação (4.14), que refletem a "proximidade" entre vértices. Supondo que os pesos $\delta_1,\ \delta_2,\ \delta_3$ são escolhidos de modo que

$$\Gamma_{01} < \Gamma_{12} < \Gamma_{23} < \Gamma_{34} < \Gamma_{40},$$

então a rota formada seria como a indicada na Figura 4.4b.



(a) Distribuição geográfica dos clientes (b) Rota final

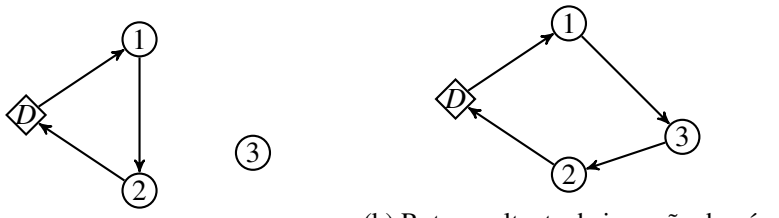
Figura 4.4 Heurística do vizinho mais próximo orientada pelo tempo

Heurísticas de inserção Este método, desenvolvido por Solomon [23], é baseado na expansão de uma dada rota pela inserção de um novo cliente k, que ainda não lhe pertence, em cada iteração.

Mais especificamente, esta heurística começa por inicializar uma rota em que é visitado um único cliente, dito semente. O cliente semente é onde começa a construção da rota e é em torno dele que esta é construída. Os clientes adicionais são selecionados, para além da rota atual, com base na distância dos vértices ainda sem rota aos vértices já incluídos na rota e no tempo de serviço mínimo. Quando não houver mais clientes sem rota que possam ser inseridos na rota atual, o processo é repetido para uma nova rota. O cliente semente pode ser selecionado como o cliente que tenha que ser servido mais cedo.

Solomon propôs três variantes deste método ($I1$, $I2$ e $I3$), baseadas em critérios diferentes, para selecionar um cliente ainda não incluído na rota. Para $I1$, o objetivo é encontrar o melhor local de inserção através da combinação entre a distância e o tempo necessário para visitar o próximo cliente. O método de inserção $I2$, o cliente a ser inserido está relacionado com a minimização da distância e

tempo total percorrido naquela rota. Por fim, no método $I3$ um cliente é inserido entre dois clientes adjacentes de acordo com a urgência em visitar o cliente seguinte. O método é ilustrado na Figura 4.5, onde se considera formada a rota $\langle 0,1,2,0\rangle$, ilustrada na Figura 4.5a, e na qual se insere o vértice 3. A rota resultante é apresentada na Figura 4.5b.



(a) Rota intermédia

(b) Rota resultante da inserção do vértice 3

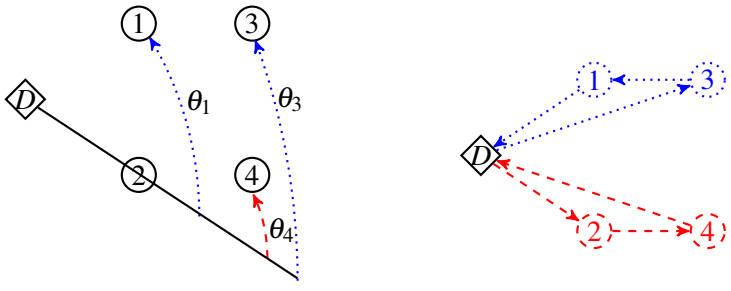
Figura 4.5 Heurísticas de inserção

Heurística de varrimento Este método foi introduzido por Gillett e Miller [11] e aplica-se a instâncias planares de um PRV. O método é constituído pelas duas fases seguintes:

1. formam-se *clusters*, isto é, grupos, de clientes, com base na sua proximidade geográfica e tendo em conta as restrições do problema;
2. determina-se a melhor rota em cada *cluster*, através da resolução de um PCV.

Algumas implementações deste método incluem ainda uma fase de pós-otimização, durante a qual vértices de *clusters* adjacentes podem ser trocados, caso tal corresponda a uma poupança na distância.

Uma implementação simples deste algoritmo consiste em representar cada vértice i pelas suas coordenadas polares relativamente ao depósito, (θ_i, ρ_i), onde θ_i denota o ângulo formado com uma semi-reta de referência, com origem no depósito, e ρ_i é a distância no plano entre o vértice i e o depósito. A semi-reta de referência é obtida atribuindo $\theta_k = 0$ a um vértice arbitrário k, sendo assim possível calcular os restantes ângulos. Para formar os *clusters* de clientes, os ângulos θ_i são ordenados por ordem crescente e todos os clientes são "varridos" usando essa ordem.



(a) *Clustering*

(b) Resolução do PCV

Figura 4.6 Heurística de varrimento

Na Figura 4.6 é exemplificada a heurística de varrimento aplicada a um caso com quatro vértices e um depósito. Tomando o depósito e o cliente 2 como referência, é possível calcular os ângulos de

cada cliente relativamente à semi-reta de referência criada, são formados dois *clusters*: um constituído pelos clientes 2 e 4, e outro pelos clientes 1 e 3, como representado na Figura 4.6a. Estes dois grupos estão representados a vermelho (tracejado) e azul (ponteado), respetivamente, na Figura 4.6b, onde se mostram também as rotas obtidas ao resolver o PCV para cada *cluster*.

4.4 Métodos heurísticos de melhoria

Os métodos heurísticos de melhoria são aplicados após ser conhecida uma solução do problema, gerada por algum dos métodos descritos acima. O objetivo é melhorar essa solução, trocando subcaminhos dentro da mesma rota ou em mais do que uma, ou trocando clientes na mesma rota ou em rotas diferentes. Trocas realizadas entre caminhos ou clientes na mesma rota dizem-se intra-rota. Caso essas trocas afetem rotas diferentes, chamam-se inter-rota.

Or-Opt (intra-rota) Or [18], propôs este método de forma a melhorar uma solução proveniente de uma heurística construtiva. Tal baseia-se na realocação de um subcaminho, constituído por uma sequência de um, dois ou três clientes, para uma posição diferente na rota preservando a sua orientação. Esta troca é feita sempre que a distância obtida com a troca seja inferior à distância anterior. Este método tem uma complexidade de $\mathscr{O}\left(n^2\right)$. Esta situação é ilustrada na Figura 4.7.

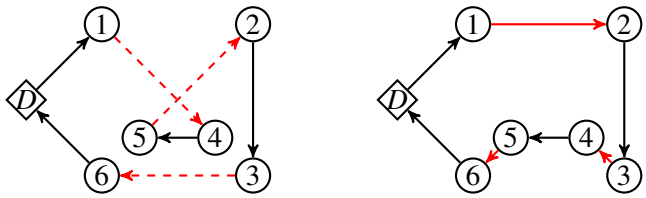


Figura 4.7 Or-opt

Reposicionamento de clientes (intra e inter-rota) Esta heurística foi introduzida por Savelsbergh em 1992 [22], juntamente com os operadores troca e trocas cruzadas resumidos adiante. O método procede à realocação de clientes numa solução, se esta implicar uma poupança positiva, ou seja, se houver redução no comprimento da rota. Sejam $I, O \subseteq V_{0,n+1} \times V_{0,n+1}$ os conjuntos das arestas a inserir na rota e das arestas a eliminar da rota, respetivamente, e

$$d_I = \sum_{(i,j)\in I} d_{ij} \quad \text{e} \quad d_O = \sum_{(i,j)\in O} d_{ij}.$$

Então, a poupança ao substituir O por I é dada por $d_O - d_I$ e a alteração das rotas é efetuada se $d_O - d_I > 0$. Este método possui uma complexidade de $\mathscr{O}(n)$. Este método pode igualmente ser aplicado em subcaminhos em rotas diferentes.

As Figuras 4.8 e 4.9 mostram exemplos do método de reposicionamento de clientes, nos casos intra e inter-rota, respetivamente. Na Figura 4.8, $O = \{\{1,3\},\{3,2\},\{4,5\}\}$ e $I = \{\{1,2\},\{4,3\},\{3,5\}\}$. Após calcular d_I e d_O as arestas de O são substituídas pelas de I, alterando assim a posição do cliente 3 na rota. O mesmo pode ser feito para rotas diferentes, como se mostra na Figura 4.9. Neste caso a alteração é semelhante à anterior, mas afeta duas rotas distintas.

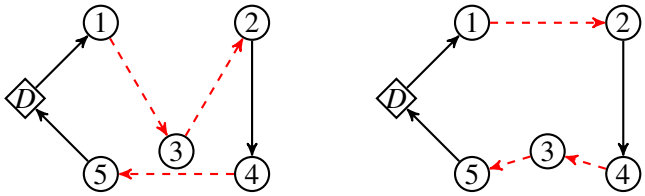


Figura 4.8 Reposicionamento de clientes (Intra-rota)

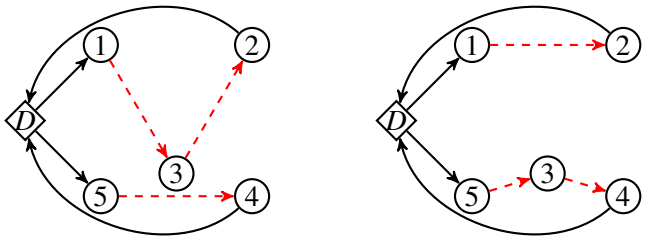


Figura 4.9 Reposicionamento de clientes (Inter-rota)

Troca (intra- e inter-rota) Este método troca a posição de dois clientes numa rota, para serem servidos por uma ordem diferente, de forma a que haja uma poupança positiva. A poupança é dada por $d_O - d_I$, da mesma forma que na operação definida anteriormente. Este operador pode ser também considerado como a aplicação sucessiva de dois métodos de reposicionamento de clientes. Além disso, tal como anteriormente, é extensível a rotas diferentes.

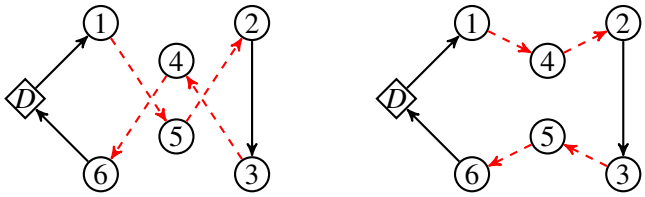


Figura 4.10 Troca (Intra-rota)

A Figura 4.10 é um exemplo da aplicação do método da troca, no caso intra-rota, onde é feita a troca das arestas $O = \{\{1,5\},\{5,2\},\{3,4\},\{4,6\}\}$ por $I = \{\{1,4\},\{4,2\},\{3,5\},\{5,6\}\}$, de forma a que os clientes 4 e 5 sejam atendidos por uma ordem diferente, permitindo assim melhorar o valor da função objetivo. Esta ideia pode ser aplicada em rotas diferentes (caso inter-rota), trocando os clientes de rota de forma a gerar uma melhoria. Este caso pode ser observado na Figura 4.11.

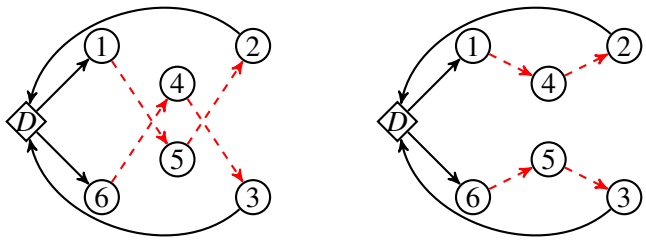


Figura 4.11 Troca (Inter-rota)

2-Opt* (inter-rota) Este método, proposto por Potvin e Rousseau [19], é semelhante à heurística Or-Opt em termos de implementação, mas as suas soluções resultam da modificação em duas arestas de rotas diferentes. O procedimento é ilustrado na Figura 4.12.

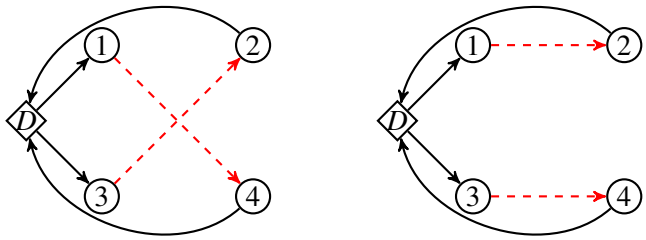


Figura 4.12 Heurística 2-opt*

Trocas cruzadas (inter-rota) O método das trocas cruzadas consiste em selecionar dois subcaminhos, ou seja, duas sequências de clientes, e trocar as suas posições na rota. O tamanho da troca cruzada é tipicamente reduzido, considerando-se apenas sequências com um número limitado de clientes. Este método tem uma complexidade de $\mathcal{O}(n)$ e é exemplificado na Figura 4.13.

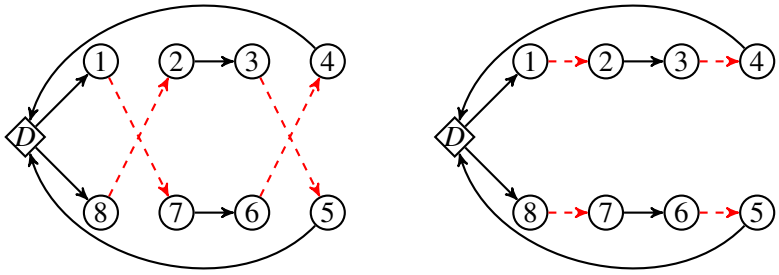


Figura 4.13 Trocas cruzadas

Capítulo 5

Testes Computacionais

5.1 Introdução

Este capítulo é dedicado ao estudo computacional do PRP, apresentado no Capítulo 4. Primeiramente analisa-se brevemente a dificuldade do problema, utilizando *software* comercial especializado para resolver um conjunto de concretizações do problema geradas aleatoriamente. Em seguida descrevem-se métodos heurísticos baseados nos discutidos no Capítulo 4, que são posteriormente testados para o mesmo conjunto de problemas e para um grupo de problemas de maiores dimensões.

5.2 Resolução exata do PRP

Como referido anteriormente, o PRP é um problema difícil em termos computacionais. Nesta secção pretende-se avaliar empiricamente a dificuldade do problema e, em particular, averiguar para que dimensões do problema é possível encontrar uma solução ótima em tempo "razoável". Para este efeito utilizou-se a formulação (4.12) e o pacote comercial para problemas de programação matemática CPLEX [6]. Os problemas foram resolvidos utilizando o CPLEX 12.6, numa máquina Dual Core AMD Opteron Processor 270 com 8 GB de RAM.

Cada concretização do PRP é caracterizada por dois parâmetros, o número de clientes, n, e o número de veículos, n_v. Nos testes realizados consideraram-se os valores resumidos na Tabela 5.1.[1]

Tabela 5.1 Conjunto de dados

n	n_v
15	6, 8, 10
19	8, 10, 13
25	11, 13, 18

Os n vértices a visitar foram gerados aleatoriamente, sem repetição, como pontos distribuídos num quadrado com 20 quilómetros de lado. A distância entre vértices foi a distância euclideana. Supôs-se que o tempo de deslocação entre dois vértices $i, j \in V$ depende linearmente da distância entre eles, de

[1] Os resultados discriminados podem ser consultados em http://www.mat.uc.pt/ mat0912/Resultados/.

acordo com

$$t_{ij} = \frac{d_{ij}}{v_m},$$

onde $v_m = 35$ quilómetros/hora é um valor constante e representa a velocidade média de qualquer veículo.

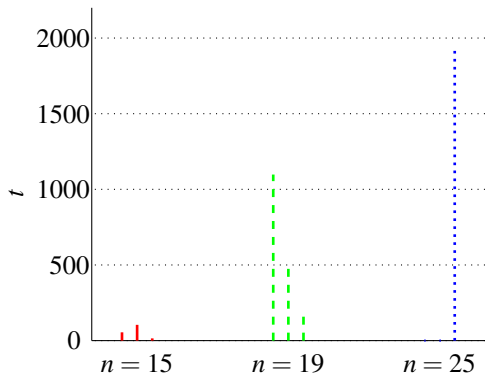
Os limites inferior e superior das janelas temporais também foram calculados aleatoriamente, no intervalo das 8 às 18 horas. Para cada par (n, n_v) geraram-se concretizações do PRP para 15 sementes distintas. Os 15 problemas foram resolvidos pelo CPLEX, com tempo limite máximo de 1 hora (3600 segundos) para cada um.

Alguns dos problemas gerados revelaram-se inadmissíveis e não são considerados no que se segue. Os problemas admissíveis dividem-se em dois grupos: os resolvidos em menos de 1 hora e aqueles para os quais não foi possível encontrar uma solução ótima em menos de 1 hora.

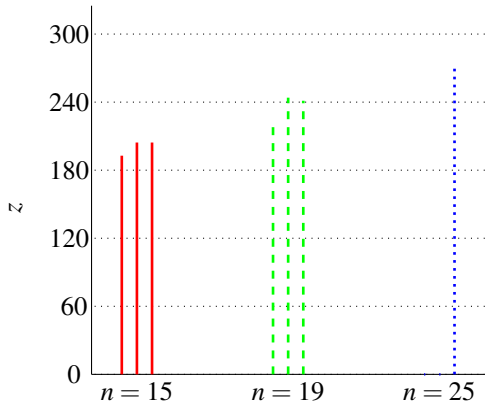
Tabela 5.2 Resultados do CPLEX para problemas resolvidos em menos de 1 hora

	$n = 15$			$n = 19$			$n = 25$
	$n_v = 6$	$n_v = 8$	$n_v = 10$	$n_v = 8$	$n_v = 10$	$n_v = 13$	$n_v = 18$
$\bar{t}$	54.808	104.269	14.749	1097.723	519.759	167.126	1934.730
s_t	78.377	279.508	16.066	1331.462	381.647	137.438	0.000
$\bar{z}$	192.784	204.391	204.391	218.077	243.927	241.219	269.736
s_z	6.458	21.349	21.349	24.947	39.459	38.354	0.000
# prob.	5	15	15	8	12	13	1

A Tabela 5.2 refere-se aos problemas resolvidos até à otimalidade em menos de 1 hora. Na tabela resumem-se as médias e os desvios padrão dos tempos de execução do CPLEX ($\bar{t}$ e s_t, respetivamente) e dos valores objetivo ótimos ($\bar{z}$ e s_z, respetivamente). Na última linha apresenta-se ainda o número de problemas considerado para cálculo dos valores médios e do desvio padrão, uma vez que este pode ter sido inferior aos 15 problemas gerados para cada par (n, n_v) fixo. É de notar que os valores do desvio padrão dos tempos requeridos pelo CPLEX são elevados. Isto reflete o facto de alguns dos problemas se terem mostrado consideravelmente mais difíceis de resolver do que outros.



(a) Tempo de execução (segundos)



(b) Valor objetivo da solução ótima

Figura 5.1 Resultados do CPLEX para problemas resolvidos em menos de 1 hora

Os gráficos na Figura 5.1 apresentam os valores médios obtidos relativamente ao tempo – Figura 5.1a – e à distância total das rotas ótimas – Figura 5.1b. Os valores estão agrupados por número de vértices do problema e, para cada n fixo apresentam-se os resultados por ordem crescente do número de veículos. Tal como indicado na Tabela 5.2, apenas para um dos problemas considerados com $n = 25$ foi encontrada solução ótima, quando $n_v = 18$, motivo pelo qual se mostra apenas uma barra nesse caso.

Na Figura 5.1a observamos que os tempos de execução para problemas com $n = 15$ são relativamente baixos e estáveis, comparando com os tempos quando $n = 19$. Com o aumento para 19 vértices observa-se um aumento da média dos tempos de execução, assim como do respetivo desvio padrão. A situação é análoga quando $n = 25$, embora neste caso a amostra seja constituída por um único problema. Os restantes não foram resolvidos em menos de 1 hora. Através da Figura 5.1b verificamos que os valores médios da função objetivo aumentam ligeiramente com o número de vértices a visitar e o número de veículos disponíveis. No entanto, não existe uma diferença acentuada para as várias dimensões dos problemas.

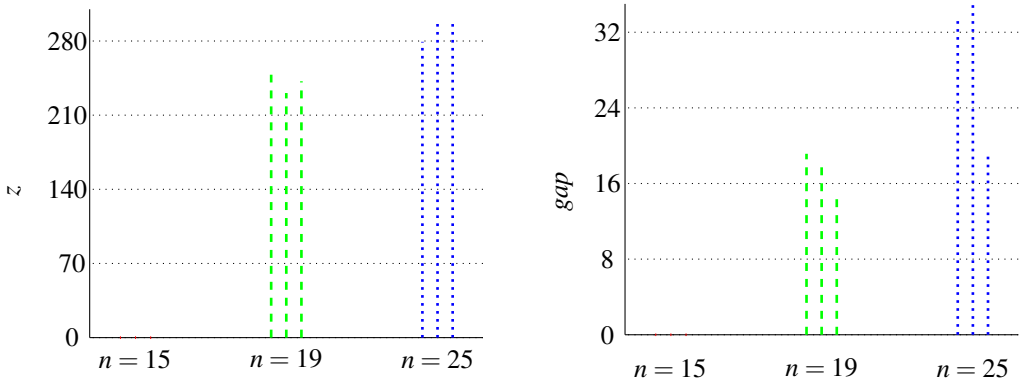
Tabela 5.3 Resultados do CPLEX para problemas não resolvidos até 1 hora

	$n = 19$			$n = 25$		
	$n_v = 8$	$n_v = 10$	$n_v = 13$	$n_v = 11$	$n_v = 13$	$n_v = 18$
$\bar{z}$	248.194	230.903	241.994	279.251	296.087	296.020
s_z	11.063	15.196	18.503	20.676	37.498	38.399
$\bar{x}_{gap}$	19.143	18.513	14.750	33.291	35.306	19.383
s_{gap}	3.770	6.392	5.176	7.207	10.569	7.606
# prob.	3	3	2	12	15	14

Os restantes problemas considerados não puderam ser resolvidos em menos de 1 hora. A Tabela 5.3 resume os resultados obtidos para estes problemas, excluindo-se os tempos de execução, dado serem todos iguais a 3600 segundos. Os valores apresentados na tabela são as médias e os desvios padrão do valor objetivo da última solução admissível encontrada e do *gap* correspondente ($\bar{x}_{gap}$ e s_{gap}). Tal como anteriormente, a última linha da tabela contém o número de elementos da amostra. O *gap* é um valor devolvido pelo CPLEX, definido em percentagem, que mede a proximidade da solução admissível encontrada relativamente à solução ótima (ou ao melhor limite inferior conhecido).

Os gráficos na Figura 5.2 apresentam os valores médios reportados na Tabela 5.3. A Figura 5.2a mostra a variação da distância total das rotas encontradas, enquanto que a Figura 5.2b mostra os valores médios do *gap* no momento em que foi interrompida a execução do CPLEX. Analogamente ao que aconteceu na figura anterior, todos os problemas com $n = 15$ foram resolvidos até ao fim, e portanto o gráfico não mostra barras para esse caso.

Relativamente à Figura 5.2a, nota-se que os valores da função objetivo observados são superiores aos obtidos no caso abordado anteriormente, o que pode dever-se ao facto de nenhum dos problemas ter sido resolvido até encontrar a solução ótima. Pelo mesmo motivo os *gap*'s na Figura 5.2b variam entre 14.750% e 35.306%, valores razoavelmente elevados e que indicam que a melhor solução encontrada está ainda longe da solução ótima. Estes valores aumentam ligeiramente com o número de vértices e o número de veículos, o que sugere que nesse caso o *software* necessita de mais tempo computacional para encontrar uma solução ótima, ou pelo menos uma solução relativamente "boa".



(a) Valor objetivo da melhor solução encontrada

(b) *gap* (%) para a melhor solução encontrada

Figura 5.2 Resultados do CPLEX para problemas não resolvidos até 1 hora

5.3 Resolução aproximada do PRP

Nesta secção pretende-se estudar alguns métodos para resolução aproximada do PRP, adaptando as heurísticas de Clarke e Wright, do vizinho mais próximo orientada pelo tempo e Or-Opt, as duas primeiras do tipo construtivo e a última de melhoria. Para tal começamos por descrever como são implementadas, de forma a satisfazer as restrições do problema. Posteriormente apresentamos os resultados obtidos para o conjunto de testes computacionais já descrito e em seguida para um conjunto de concretizações do PRP de maior dimensão.

Heurística de Clarke e Wright Conforme apresentado na Secção 4.3, esta heurística começa por criar rotas iniciais que visitam um único cliente. Após este passo, faz-se a união das rotas com dois clientes, caso haja uma poupança na distância percorrida. Para facilitar a implementação da fusão de rotas as poupanças são calculadas previamente e organizadas por ordem decrescente. Além disso, apenas se fundem rotas considerando os clientes nas suas extremidades, isto é, o seguinte e o anterior ao depósito. No entanto, no PRP a fusão de rotas não depende apenas da distância, uma vez que é necessário garantir a admissibilidade da solução. Assim, deve atender-se também aos tempos de chegada a cada cliente, impedindo que sejam unidas rotas com janelas temporais incompatíveis. Adicionalmente, uma fusão é implementada desde que a duração da nova rota não ultrapasse T_M horas, caso contrário o processo recomeça. O pseudo-código desta heurística é resumido no Algoritmo 4.

Heurística do vizinho mais próximo orientada pelo o tempo Este método é uma extensão da heurística do vizinho mais próximo, descrita no Capítulo 3. Nesta abordagem os vértices a visitar são escolhidos de acordo com os valores de Γ_{ij} definidos por (4.14), que são uma soma ponderada pelos parâmetros $\delta_1, \delta_2, \delta_3 \in [0,1]$, tais que $\delta_1 + \delta_2 + \delta_3 = 1$, da distância entre os clientes i e j, da diferença em termos de tempo entre ambos, e da urgência em visitar o vértice j como o próximo cliente, respetivamente. Os vértices são selecionados de acordo com os valores de Γ_{ij}. Também neste caso, é necessário ter em conta as restrições temporais do PRP, isto é, só é possível incluir um determinado cliente numa rota se o tempo de chegada não excede o fim do seu período de atendimento e se a rota não ultrapassa o tempo máximo permitido. Caso contrário, passa a analisar-se o cliente

Algorithm 4: Heurística de Clarke e Wright

for $i \in V$ **do**
  $\text{Rota}_i \leftarrow \langle 0, i, n+1 \rangle$
  $t_0^i \leftarrow a_i - t_{0i}$
  $t_i^i \leftarrow t_0^i + t_{0i} + t_s$
  $t_{n+1}^i \leftarrow t_i^i + t_{i,n+1}$
for $i, j \in V$ **do** $P_{ij} \leftarrow d_{0i} + d_{j,n+1} - d_{ij}$
Ordenar P por ordem decrescente
for $i, j \in V$ *tais que* $P_{ij} > 0$ *e* i *e* j *não foram atualizados* **do**
  $ki \leftarrow$ rota em que se insere o cliente i
  $kj \leftarrow$ rota em que se insere o cliente j
  if i *e* j *são o segundo ou o penúltimo clientes de* Rota_{ki} *e* Rota_{kj} **then**
    $t_j^{ki} \leftarrow t_i^{ki} + t_{ij} + t_s + \max\{0, a_j - t_j^{ki}\}$
    if $t_{n+1}^{ki} - t_0^{ki} \leq T_M$ *e* $a_i \leq t_i^{ki} \leq b_i$ **then** Funde Rota_{ki} e Rota_{kj} e atualiza t_i^k

seguinte. Quando não é possível incluir mais clientes numa determinada rota, o processo repete-se. O método resultante é descrito pelo Algoritmo 5.

Algorithm 5: Heurística do vizinho mais próximo orientada pelo tempo

for $k \in N_v$ **do**
  $\text{Rota}_k \leftarrow \langle 0, n+1 \rangle$
  $i \leftarrow 0$
  for $j \in V_{0,n+1}$ *tal que* j *ainda não foi incluído em nenhuma rota* **do**
    $j \leftarrow \text{argmin}\{\Gamma_{ij} : i \in V_{0,n+1}\}$
    $t_j^k \leftarrow t_i^k + t_s + t_{ij} + \max\{0, a_j - t_j^k\}$
    $t_{n+1}^k \leftarrow t_j^k + t_{j,n+1}$
    if $t_{n+1}^k - t_0^k \leq T_M$ *e* $a_j \leq t_j^k \leq b_j$ **then**
      Insere j em Rota_k e atualiza os tempos de chegada
    $i \leftarrow j$

Heurística de melhoria Or-Opt Tal como apresentado na Secção 4.4, este método é útil para melhorar uma solução admissível obtida por um método construtivo. Procura-se diminuir a distância total percorrida e atender cada cliente o mais cedo possível, reduzindo assim o tempo de cada rota. Esta heurística tenta alterar a posição dos clientes nas rotas. Esta alteração será feita apenas no caso da distância percorrida antes da mudança ser superior à distancia obtida após a alteração do cliente. Além disso, é preciso calcular os tempos de chegada a cada cliente e verificar a admissibilidade das janelas temporais durante essas trocas. Estas melhorias podem ser feitas várias vezes (sem voltar à rota inicial), de forma a obter a melhor distância possível. Novamente, a duração da rota não deve ultrapassar as T_M horas, portanto uma alteração na posição de clientes que exceda o tempo limite de circulação não será efetuada. O Algoritmo 6 resume esta descrição.

Algorithm 6: Heurística Or-Opt

for $k \in N_v$ **do**
  $dimRota \leftarrow$ dimensão de Rota$_k$
  if $dimRota \geq 4$ **then**
    $dist_{ant} \leftarrow$ calcula a distância de Rota$_k$
    $TT_{rotaAnt} \leftarrow t^k_{n+1} - t^k_0$
    for $i = 1, \ldots, dimRota - 1$ **do**
      for $j = i+1, \ldots, dimRota - 1$ **do**
        Faz uma troca auxiliar entre os clientes i e j na Rota$_k$
        $dist_{nova} \leftarrow$ calcula a nova distância da rota k
        $t^k_j \leftarrow t^k_i + t_s + t_{ij} + \max\{0, a_j - t^k_j\}$
        $TT_{rota} \leftarrow t^k_{n+1} - t^k_0$
        if $dist_{nova} < dist_{ant}$ *ou* $TT_{rota} < TT_{rotaAnt}$ *e* $a_i \leq t^k_i \leq b_i$ *e* $TT_{rota} \leq T_M$ **then**
          Atualiza as rotas e os tempos de chegada
          $dist_{ant} \leftarrow dist_{nova}$

Análise dos resultados Apresentamos agora os resultados obtidos pelos métodos aproximados em testes computacionais. Os métodos foram implementados em linguagem C++ e executados na máquina já referida.[2] No que se segue as heurísticas de Clarke e Wright, do vizinho mais próximo orientada pelo tempo e Or-Opt designam-se por `CW`, `VPOT` e `OrOpt`, respetivamente. Consideram-se três versões do método `VPOT`, designadas `VPOT`$_i$, $i = 1,2,3$, com ponderações $\delta^1 = (0.0, 0.0, 1.0)$, $\delta^2 = (0.5, 0.0, 0.5)$ e $\delta^3 = (0.2, 0.2, 0.6)$, respetivamente. Nestes testes foram usados os mesmos problemas descritos no início do capítulo, sendo possível comparar as soluções aproximadas com as soluções exatas ou com a melhor solução devolvida pelo CPLEX, para problemas não resolvidos até ao final.[3]

A Tabela 5.4 mostra médias das percentagens da distância relativa (Δ_z) entre a solução exata (z^*) e a solução obtida por uma das heurísticas (z_{aprox}),

$$\Delta_z = \frac{z_{aprox} - z^*}{z^*} \times 100(\%) \tag{5.1}$$

Esta tabela contém também os resultados para Δ_z para as combinações dos métodos construtivos com a heurística `OrOpt`. Daqui em diante os tempos de execução, t, são apresentados em milissegundos. Os valores a negrito são os melhores obtidos para cada problema, em termos de tempo e em termos de qualidade da solução.

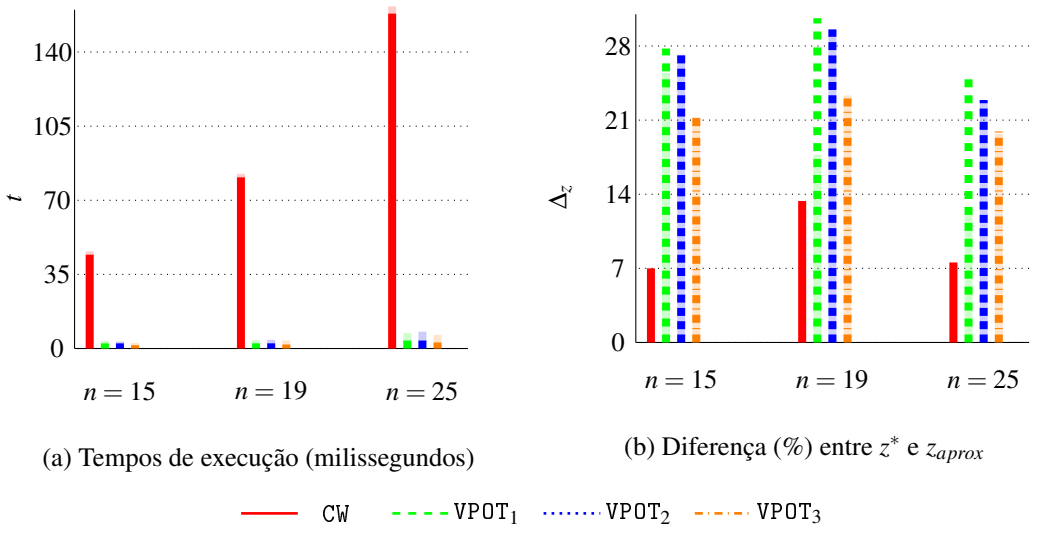
A Figura 5.3, mostra os valores anteriores, distinguindo o método construtivo e o método de melhoria através de diferentes tonalidades em cada barra. Os valores das heurísticas construtivas aparecem a opaco e os valores da heurística `OrOpt` aparecem translúcidos.

Com base na Figura 5.3b, concluímos que as soluções obtidas pela heurística `CW` são mais próximas das soluções exatas do que as encontradas pelas heurísticas `VPOT`. Observa-se também que a versão da heurística `VPOT` com os melhores resultados é `VPOT`$_3$, que dá prioridade à urgência em visitar o

[2] Os códigos podem ser consultados em http://www.mat.uc.pt/ mat0912/Codigos/.
[3] Os resultados discriminados podem ser consultados em http://www.mat.uc.pt/ mat0912/Resultados/Aproximados/.

Tabela 5.4 Resultados dos métodos construtivos para problemas pequenos

		$n = 15$		$n = 19$		$n = 25$	
		S/ `OrOpt`	C/ `OrOpt`	S/ `OrOpt`	C/ `OrOpt`	S/ `OrOpt`	C/ `OrOpt`
`CW`	$\bar{\Delta}_z$	6.995	**6.995**	13.365	**13.109**	7.555	**7.518**
	s_{Δ_z}	7.459	7.459	23.183	23.173	4.771	4.764
	$\bar{t}$	44.301	45.816	80.526	82.489	158.153	161.582
	s_t	13.799	13.805	21.135	20.938	27.564	27.652
$\texttt{VPOT}_1$	$\bar{\Delta}_z$	27.766	25.448	30.630	17.740	25.481	12.075
	s_{Δ_z}	15.954	9.793	19.641	26.479	12.671	13.125
	$\bar{t}$	2.433	3.581	2.452	3.973	4.875	7.299
	s_t	0.163	0.421	0.767	1.133	0.649	1.235
$\texttt{VPOT}_2$	$\bar{\Delta}_z$	27.174	27.174	29.568	29.557	22.899	22.754
	s_{Δ_z}	16.046	16.046	19.041	19.000	10.649	10.392
	$\bar{t}$	2.435	3.285	2.449	4.175	4.898	7.949
	s_t	0.159	0.314	0.764	1.217	0.639	1.656
$\texttt{VPOT}_3$	$\bar{\Delta}_z$	21.313	21.283	23.392	23.330	19.937	19.728
	s_{Δ_z}	13.619	13.664	18.563	18.352	10.880	10.733
	$\bar{t}$	1.535	**2.606**	1.894	**3.791**	2.851	**6.456**
	s_t	0.258	0.525	0.434	0.957	0.739	1.586



(a) Tempos de execução (milissegundos)

(b) Diferença (%) entre z^* e z_{aprox}

Figura 5.3 Resultados dos métodos construtivos para problemas pequenos

cliente j antes do fim da janela temporal, após visitar o cliente i, mas não negligenciando os restantes critérios. Os tempos de execução, ilustrados na Figura 5.3a, são todos muito próximos de 0 segundos, exceto para o método `CW`, que apresentou tempos mais elevados. Em geral, o método mais rápido é a heurística `VPOT`, em contrapartida a distância destas soluções relativamente às ótimas varia entre 20% e 30%. Analisando a Tabela 5.4 pode-se observar que as soluções obtidas com a heurística `OrOpt` foram melhoradas, no entanto com melhorias relativamente reduzidas. É de notar que a heurística mostra melhorias em quase todas as soluções, exceto ao aplicar $\texttt{VPOT}_2$ e `CW` a problemas com $n = 15$. É de notar que para $\texttt{VPOT}_1$ se obtiveram melhorias onde foram mais significativas para problemas

com $n = 19$ e $n = 25$. Em geral, as melhorias nas soluções obtidas foram bastante reduzidas, o que é expetável uma vez que as rotas do PRP têm poucos clientes e, portanto, muitas alterações são inadmissíveis em termos de janelas temporais ou de tempo total de rota. Relativamente aos tempos de execução do algoritmo, estes são próximos de 0 segundos. No entanto, a Figura 5.3a mostra um crescimento gradual do tempo, à medida que o número de clientes aumenta. No entanto, a heurística `CW` com a heurística `OrOpt` é a que apresenta os tempos de execução mais elevados.

Para avaliar o desempenho dos métodos implementados em problemas com maiores dimensões, testaram-se os mesmos considerando $n = 50, 70, 90, 130$. Também neste caso se geraram 15 instâncias para cada número de clientes.

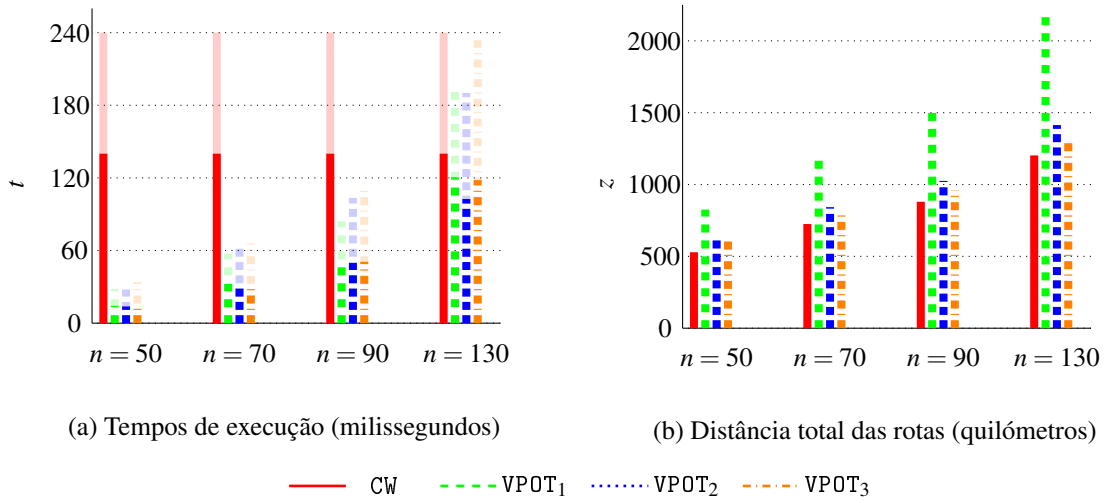
Tabela 5.5 Resultados dos métodos construtivos e de melhoria para problemas grandes

		$n = 50$		$n = 70$		$n = 90$		$n = 130$	
		S/ `OrOpt`	C/ `OrOpt`	S/ `OrOpt`	C/ `OrOpt`	S/ `OrOpt`	C/ `OrOpt`	S/ `OrOpt`	C/ `OrOpt`
`CW`	$\bar{z}$	528.468	**528.326**	725.015	**724.863**	879.763	**879.358**	1202.061	**1201.791**
	s_z	106.563	106.669	120.398	120.301	157.352	157.016	271.772	272.025
	$\bar{t}$	1274.297	1295.483	4083.937	4119.452	10055.28	10122.88	39020.11	39137.75
	s_t	33.919	33.513	40.797	42.373	100.134	103.020	354.909	350.852
	$\bar{n}_v$	**19**	**19**	**25**	**25**	**31**	**31**	**42**	**42**
$\texttt{VPOT}_1$	$\bar{z}$	825.399	814.521	1188.933	1164.582	1531.291	1510.201	2218.389	2186.227
	s_z	125.230	106.983	149.905	124.314	160.284	143.116	293.518	254.275
	$\bar{t}$	14.349	28.148	32.347	**57.314**	**48.436**	**84.205**	120.55	193.456
	s_t	3.652	7.231	4.185	9.304	10.604	18.089	22.12	30.544
	$\bar{n}_v$	24	24	33	33	42	42	60	60
$\texttt{VPOT}_2$	$\bar{z}$	629.749	628.796	842.185	841.184	1025.229	1024.717	1413.812	1413.264
	s_z	127.365	126.143	146.671	146.006	133.883	133.619	263.539	263.437
	$\bar{t}$	**13.811**	**27.826**	**30.292**	61.665	49.686	103.489	**102.588**	**190.225**
	s_t	3.948	7.959	6.138	11.989	7.562	18.272	13.577	16.476
	$\bar{n}_v$	24	24	30	30	37	37	51	51
$\texttt{VPOT}_3$	$\bar{z}$	612.131	610.491	791.898	790.557	960.469	959.714	1306.404	1305.317
	s_z	101.397	99.883	115.862	115.932	113.393	113.537	217.394	216.79
	$\bar{t}$	14.433	34.011	30.828	65.867	50.724	109.775	118.179	233.687
	s_t	1.855	5.479	1.449	9.635	2.106	13.900	18.748	27.028
	$\bar{n}_v$	20	20	26	26	**31**	**31**	**42**	**42**

A Tabela 5.5 mostra que em todos os casos a aplicação do método `OrOpt` mantém o número de veículos utilizados, mas melhora a distância total por eles percorrida. Além disso, as distâncias da solução final obtida por $\texttt{VPOT}_1$ são, em média, superiores às obtidos pelos restantes métodos. Por outro lado o método `CW` foi o que revelou melhores resultados neste aspeto, enquanto que $\texttt{VPOT}_3$ foi o melhor entre os métodos `VPOT`. Pode verificar-se também que os tempos de execução do método `CW` foram muito superiores aos obtidos pelas heurísticas `VPOT`, variando aproximadamente entre os 21 e os 650 segundos. Observa-se ainda que as heurísticas `CW` e $\texttt{VPOT}_3$ são as que apresentam os menores números médios de veículos necessários para resolver os problemas.

A Figura 5.4 mostra os valores anteriores, distinguindo o método construtivo e o método de melhoria através de diferentes tonalidades em cada barra. Os valores das heurísticas construtivas aparecem a opaco e os valores da heurística `OrOpt` aparecem translúcidos. Nota-se que a distância total das soluções não altera de forma significativa quando aplicado o método `OrOpt`, contudo os

tempos de execução aumentam. Além disso, em termos da qualidade das soluções a heurística `CW` é a mais adequada, em contrapartida este método exige mais tempo de resolução do que a heurística `VPOT`. O método mais rápido foi a heurística $\texttt{VPOT}_1$, mas apresentou os piores resultados em distância. A heurística $\texttt{VPOT}_2$ foi a mais equilibrada em termos de tempo de resolução e de distância. A versão $\texttt{VPOT}_3$ foi mais lenta, no entanto apresentou melhores soluções do que as outras duas.



(a) Tempos de execução (milissegundos)

(b) Distância total das rotas (quilómetros)

Figura 5.4 Resultados dos métodos construtivos e de melhoria para problemas grandes

Em conclusão, os métodos implementados permitem resolver o PRP, conseguindo obter soluções relativamente próximas das exatas em pouco tempo. Este facto pode ser observado na análise das comparações das soluções exatas com as aproximadas. No entanto, o método `CW` apresenta melhores soluções que a heurística `VPOT`, mas em termos de tempos de execução são relativamente superiores. A heurística `OrOpt` permite uma melhoria da solução final, embora, em geral, muito reduzidas. Este efeito parece dever-se ao número reduzido de clientes que cada rota inclui, o que dificulta a existência de trocas compatíveis em termos de janelas temporais e também a limitação temporal para cada rota. Ainda assim, em geral, o esforço computacional exigido pelos métodos heurísticos é reduzido.

Capítulo 6

Conclusão

Neste trabalho abordou-se uma versão do problema do caixeiro viajante múltiplo, designada por problema de recolha de produtos, que tem por objetivo calcular rota para uma frota de veículos, de modo a servir um dado conjunto de clientes, satisfazendo as janelas temporais em cada cliente e não ultrapassando um tempo total de viagem previamente estabelecido.

Numa primeira fase introduziram-se alguns conceitos preliminares sobre teoria de grafos e apresentou-se o problema do caixeiro viajante, que pode ser visto como um caso particular do PCVM em que um dado conjunto de vértices deve ser ligado por uma única rota. Estudaram-se algumas propriedades do PCV, referindo-se a sua dificuldade computacional, assim como métodos de aproximação e métodos heurísticos para este problema. Em seguida formulou-se o PRP como um programa linear misto e ilustrou-se o problema. Devido à proximidade do PRP e do PCVM ou do problema de rotas de veículos, descreveram-se métodos heurísticos, construtivos e de melhoria, conhecidos na literatura. Tal como para o PCV, a abordagem heurística destes problemas é justificada pela sua acrescida dificuldade. Em geral estes métodos permitem obter soluções próximas da ótima em tempos computacionais razoáveis, e procura-se que impliquem um esforço de implementação menor do que os métodos exatos.

O Capítulo 5 é dedicado a um estudo computacional do PRP baseado em dois conjuntos de testes, com diferentes dimensões e simulando um serviço a clientes gerados uniformemente num quadrado com 20 quilómetros de lado. Para este estudo adaptaram-se e implementaram-se algumas das heurísticas descritas para o PCVM e o PRV. Resolveu-se a formulação apresentada antes através de métodos exatos, com o auxílio do *software* CPLEX, e utilizaram-se os métodos heurísticos.

O programa CPLEX foi eficaz para resolver problemas com dimensões até 15 clientes, em tempos inferiores a 2 minutos. No entanto, não encontrou solução ótima para vários problemas com 19 e 25 clientes no tempo limite de 1 hora, o que confirma a dificuldade do PRP.

As heurísticas construtivas implementadas, `CW` e `VPOT`, foram capazes de encontrar soluções admissíveis para os problemas com um esforço computacional reduzido, embora ainda sensível ao número de clientes. Concretamente, problemas com 25 clientes foram resolvidos em menos de 3 segundos com uma diferença média de cerca de 8% relativamente ao valor objetivo ótimo pelo método `CW`, e em menos de 3 milissegundos com uma diferença de cerca de 20% relativamente ao valor objetivo ótimo pelo método `VPOT`. Num segundo conjunto de testes os mesmos métodos encontraram rotas admissíveis para problemas com 130 clientes em cerca de 11 minutos e de 2 segundos, respetivamente.

É de salientar que a heurística `CW` foi sempre mais lenta do que `VPOT`, tendo no entanto conduzido a soluções melhores.

A heurística de melhoria implementada, `OrOpt`, permitiu melhorar ligeiramente as soluções encontradas pelos métodos `CW` e `VPOT` com um tempo de execução adicional muito reduzido. O melhoramento das soluções observado foi mais acentuado para a versão $\texttt{VPOT}_1$.

Dos testes realizados parece poder concluir-se que em termos de rapidez de execução as heurísticas mais indicadas são $\texttt{VPOT}_2$ e $\texttt{VPOT}_3$. Se se pretende priveligiar a qualidade da solução final, pensando na distância percorrida por todos os veículos e no número de veículos da frotas, então o métodos com melhores resultados são `CW` e $\texttt{VPOT}_1$, qualquer um deles seguido da aplicação do método `OrOpt`.

Bibliografia

[1] D. Applegate, R. Bixby, V. Chvatal, and W. Cook. *The traveling salesman problem: a computational study*. Princeton university press, 2011.

[2] T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34:209–219, 2006.

[3] B. Chandra, H. Karloff, and C. Tovey. New results on the old *k*-opt algorithm for the traveling salesman problem. *SIAM Journal on Computing*, 28:1998–2029, 1999.

[4] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Technical Report 388, GSIA, Carnegie-Mellon University, Pittsburgh, 1976.

[5] G. Clarke and J. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12:568–581, 1964.

[6] IBM ILOG CPLEX. V12. 1: User's manual for CPLEX. *International Business Machines Corporation*, 46:157, 2009.

[7] J. Desrosiers, F. Soumis, and M. Desrochers. Routing with time windows by column generation. *Networks*, 14:545–565, 1984.

[8] J. Edmonds and R. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19:248–264, 1972.

[9] M Fleury. Deux problemes de geometrie de situation. *Journal de mathematiques elementaires*, 2:257–261, 1883.

[10] M. Garey and D. Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

[11] B. Gillett and L. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22:340–349, 1974.

[12] J. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7:48–50, 1956.

[13] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European journal of operational research*, 59:345–358, 1992.

[14] E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys. *The Traveling Salesman Problem*. John Wiley & Sons, New York, 1985.

[15] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21:498–516, 1973.

[16] C. Miller, A. Tucker, and R. Zemlin. Integer programming formulation of traveling salesman problem. *Journal of the ACM*, 7:326–329, 1960.

[17] C. Nilsson. Heuristics for the traveling salesman problem. *Linkoping University*, pages 1–6, 2003.

[18] I. Or. *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. PhD thesis, Northwestern University Evanston, IL, 1976.

[19] J.-Y. Potvin and J.-M. Rousseau. An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, 46:1433–1446, 1995.

[20] R. Prim. Shortest connection networks and some generalizations. *Bell Labs Technical Journal*, 36:1389–1401, 1957.

[21] D. Rosenkrantz, R. Stearns, and P. Lewis. An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing*, 6:563–581, 1977.

[22] M. Savelsbergh. The vehicle routing problem with time windows: Minimizing route duration. *ORSA journal on computing*, 4:146–154, 1992.

[23] M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35:254–265, 1987.

[24] P. Toth and D. Vigo. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, 2002.