UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE ESTATÍSTICA E INVESTIGAÇÃO OPERACIONAL

GRASP: uma aplicação ao problema de recolha e distribuição de produtos agrícolas em circuitos de proximidade

Tânia Sofia dos Santos Zhi Wen

Mestrado em Matemática Aplicada à Economia e Gestão

Trabalho de Projeto orientado por:
Professora Doutora Maria da Conceição Fonseca

2019

Agradecimentos

Quero agradecer à Professora Doutora Maria da Conceição Fonseca pela disponibilidade e contribuições que tornaram possível a realização deste trabalho de projeto.

Aos meus amigos, os Incógnitos, pelo incentivo e acompanhamento no desenvolvimento deste trabalho de projeto.

A uma amiga que me acompanha desde o secundário, Joana Vieira, pelo o apoio e aconselhamento no desenvolvimento deste trabalho de projeto.

Um agradecimento à minha família, em especial à minha mãe e irmã, pelo apoio incondicional bem como encorajamento nos tempos mais difíceis.

Obrigada por tudo.

Resumo

A maioria dos produtos agrícolas que são consumidos passam por centros de logística de grande distribuição. Um dos grandes problemas destes centros é o elevado número de quilómetros que os produtos percorrem e o elevado intervalo de tempo entre colheita e consumo. Vários fatores, por parte do consumidor, tais como, a necessidade de consumir alimentos frescos, de conhecer as suas origens e a forma como são produzidos e de ajudar os agricultores e o ambiente, conduziu a um crescimento do comércio local. Esta mudança de atitude levou à implementação de circuitos curtos e de proximidade. Nos circuitos curtos existem apenas um ou nenhum intermediário. Nos circuitos de proximidade define-se uma distância máxima entre o local de produção e venda. Pretende-se obter circuitos que sejam uma combinação entre ambos, ou seja, rotas de recolha e distribuição de produtos agrícolas, tal que, o número de intermediários e a distância entre produção e venda seja mínimo.

Este trabalho de projeto tem como base o desenvolvimento de rotas, para vários veículos, que incluem pontos de recolha e distribuição, que respeitam os princípios dos circuitos curtos e de proximidade. Os pontos de recolha são os agricultores e os mercados onde os agricultores fazem a venda direta dos seus produtos, enquanto que a distribuição é efetuada nos clientes. Os clientes podem ser: vendas *online*, restaurantes, escolas, mercearias, lojas de organização de produtores, cabazes, entre outros. Estas rotas contribuem para o desenvolvimento de sistemas locais e regionais de produção de alimentos. O desenvolvimento de um sistema local é um processo difícil, sendo importante a participação dos agricultores e dos consumidores.

O método utilizado neste trabalho de projeto, de forma a obter soluções admissíveis para este problema, é a heurística *GRASP* (*Greedy Randomized Adaptive Search Procedure*). A heurística *GRASP* será implementada em linguagem de programação MATLAB e testada com exemplos gerados aleatoriamente. A geração dos dados, considera diferentes valores para vários parâmetros, de maneira a obter diferentes cenários com o objetivo de simular a realidade.

Palavras Chave: *GRASP*, rotas de recolha e distribuição de produtos, circuitos curtos, circuitos de proximidade, programação linear inteira mista.

Abstract

Most of the agricultural products that are consumed go through large distribution logistics centers. One of the major problems of these centers is the large number of kilometers the products travel and the long time between harvest and consumption. Several factors, from the consumer's point of view, such as the need to consume fresh products, to know their origins and how they are produced and to help farmers and the environment, have led to a growth of the local trade. This change in attitude led to the implementation of short and proximity circuits. In short circuits there is only one or no intermediate. Proximity circuits define a maximum distance between the place of production and sale. It is intended to obtain circuits which are a combination of both, that is, routes with pickup and delivery of agricultural products, such that the number of intermediaries and the distance between production and sale is minimal.

This project work is based on the development of routes, for various vehicles, that include pickup and delivery, which respect the principles of short and proximity circuits. The pickup is made in farmers and markets where farmers sell their products directly, while delivery is made directly to customers. Customers can be: online sales, restaurants, schools, grocery stores, producer organization stores, baskets, among others. These routes contribute to the development of local and regional food production systems. Developing a local system is a difficult process, in which the participation of farmers and consumers is of the utmost importance.

The method used in this project work in order to obtain feasible solutions for this problem is the *GRASP* (*Greedy Randomized Adaptive Search Procedure*) heuristic. The *GRASP* heuristic will be implemented in MATLAB programming language and tested with randomly generated examples. The generation of examples considers different values for various parameters, in order to obtain different scenarios with the objective of simulating reality.

Keywords: *GRASP*, vehicle routing problem with pickup and delivery, short circuits, proximity circuits, mixed integer linear programming.

Índice

ÍNDICE

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Atualmente a maioria dos produtos agrícolas que são consumidos passam por centros de logística de grande distribuição. Em Portugal, um exemplo é o MARL (Mercado Abastecedor da Região de Lisboa) [1]. O MARL é um centro de abastecimento de produtos agroalimentares, no qual diversos segmentos de retalhistas podem encontrar uma grande diversidade de produtos alimentares produzidos desde pequenos produtores até grandes empresas de outros pontos do país ou de Espanha.

Um dos grandes problemas dos centros de logística de grande distribuição é o elevado número de quilómetros que os produtos percorrem e o elevado intervalo de tempo entre colheita e consumo. Por exemplo, na Europa, os frutos e legumes percorrem, em média, 1500 km [2].

Vários fatores, por parte do consumidor, conduziram a um aumento da procura de produtos agrícolas locais. Como por exemplo, a necessidade de consumir alimentos frescos, de conhecer a origem dos alimentos e a forma como são produzidos e de ajudar os agricultores e o ambiente. Esta mudança de atitude conduziu a um crescimento do comércio local, que levou à implementação de circuitos curtos e de proximidade por parte dos produtores e empresas.

Define-se circuitos curtos como uma forma de comercializar produtos agrícolas, podendo ser através de vendas diretas ou indiretas. Vendas diretas implicam a inexistência de intermediários entre o agricultor e o consumidor (exemplos: mercados rurais, feiras). Vendas indiretas significa a existência de apenas um intermediário entre o agricultor e o consumidor (exemplos: vendas *online*, restaurantes, escolas, mercearias, lojas de organização de produtores, cabazes)

Nos circuitos de proximidade define-se uma distância máxima entre o local de produção e o local de venda. Define-se esta distância por *food miles*.

Pretende-se obter circuitos que sejam uma combinação entre circuitos curtos e de proximidade, ou seja, pretende-se determinar rotas de recolha e distribuição de produtos agrícolas tal que o número de intermediários e de *food miles* seja mínimo.

O desenvolvimento de sistemas locais e regionais de produção de alimentos surgiu no Japão, alargando-se para os Estados Unidos e para a Europa [3]. O aparecimento de sistemas deste género, por todo o mundo, ocorreu devido à criação de estruturas de suporte e fundações de apoio. Um exemplo é a fundação RUAF (*Resource Centre on Urban Agriculture and Food Security*). A RUAF contribui para a redução da pobreza urbana e o melhoramento da segurança alimentar e da gestão ambiental urbana [2]. Outro apoio, para o desenvolvimento de sistemas deste género, é a nível político. Por exemplo, na França e na Grã-Bretanha, o governo nacional promove o uso de sistemas locais e regionais de produção de alimentos [2]. Contudo, em Portugal, ainda existe pouca preocupação em relação a este tema quer a nível político quer a nível da população [3]. A ADREPES (Associação para o desenvolvimento regional da península de Setúbal) em conjunto com uma associação semelhante do *Pays du Mans*, em França,

tentou em 2013 desenvolver um sistema regional de produção de alimentos [4]. Com efeito, existe uma grande hesitação na implementação de circuitos curtos e de proximidade por parte dos produtores. Deste modo, as grandes superfícies, continuam a ser a escolha principal para a compra de produtos alimentares.

Como foi indicado no parágrafo anterior, a existência de circuitos curtos e de proximidade não são novidade, principalmente na periferia das cidades. Contudo, o uso de métodos de distribuição deste género decresceu ao longo do século 20. Um exemplo é a rede de distribuição de Paris, onde há dois séculos atrás a distância percorrida por produtos agrícolas era, em média, de aproximadamente 150 km. Atualmente, a distância percorrida é cerca de 660 km, ou seja, a distância quadruplicou em apenas 200 anos [2]. Razões para este aumento foram o surgimento do transporte ferroviário, o forte crescimento do transporte rodoviário, a internacionalização dos mercados e o desenvolvimento de estruturas de fornecimento industrial. A distribuição tornou-se assim globalizada e industrializada, e a ligação entre produtores e consumidores tornou-se cada vez mais rara [2].

A implementação de circuitos curtos e de proximidade proporciona, para consumidores, negócios locais e agricultores, vários benefícios económicos, sociais e ambientais. A compra de alimentos produzidos localmente permite aos agricultores, que têm dificuldades em escoar a produção ou que são obrigados a vender a preços bastante baixos, acesso a novos mercados. Um exemplo é a utilização de produtos biológicos, provenientes de agricultores locais, na confeção de refeições em alguns refeitórios nas escolas na zona dos Olivais, em Lisboa [1]. Outro exemplo são os cabazes de legumes e frutas distribuídos em Lisboa, onde os produtos utilizados são originários de agricultores dos arredores da cidade [1]. Outro benefício é a resistência na flutuação dos preços, que possibilita o crescimento da economia local e a criação de mais oportunidades para negócios locais e de uma maior oferta de emprego. Socialmente, proporciona a criação de uma relação de confiança entre produtor e consumidor formando, deste modo, comunidades mais fortes, saudáveis e auto-suficientes. Por exemplo, em certas escolas existem quintas agrícolas, que permitem às crianças um contacto direto com a produção agrícola [5]. A implementação destes circuitos implica uma redução na distância percorrida pelos alimentos, o que provoca um decréscimo na emissão de CO_2. Contudo alguns estudos referem que o modo como são produzidos os alimentos (biológico ou não) também afeta a forma como a pegada ecológica varia [6].

O desenvolvimento de um sistema alimentar local é uma processo difícil, sendo fundamental a cooperação dos produtores e dos consumidores. A existência de organizações regionais, que envolvam os agricultores e os consumidores, é um ponto importante para o êxito no desenvolvimento de sistemas alimentares locais. Além disso, é necessário que haja uma forte perceção do futuro, de forma a haver condições adequadas para a implementação de modelos de produção e comercialização local.

Este trabalho apresenta a criação de rotas que permitem a recolha de produtos alimentares nos agricultores e nos mercados onde os agricultores vendem diretamente e posterior distribuição aos clientes. Estas rotas contribuem para o desenvolvimento de sistemas locais e regionais de produção de alimentos. O problema em estudo, consiste na determinação de rotas, para vários veículos, em que existem pontos de recolha de produtos alimentares (*pickup*) e pontos onde esses produtos são entregues (*delivery*). Os pontos de recolha são os agricultores e os mercados onde os agricultores vendem os seus produtos diretamente, e os pontos onde os produtos são entregues são os clientes. Este problema designa-se de VRP (*Vehicle Routing Problem*) com *pickup* e *delivery*, ou seja, designa-se de VRPPD (*Vehicle Routing Problem with Pickup and Delivery*). No caso do desenvolvimento de sistemas locais de produção de alimentos, as rotas devem respeitar os princípios já indicados, dos circuitos curtos e de proximidade. Ou seja, é necessário que o número de intermediários e de *food miles* seja mínimo. O método utilizado neste trabalho para obter soluções admissíveis para este problema é a heurística GRASP (*Greedy Randomized Adaptive Search Procedure*). A heurística *GRASP* será implementada em linguagem de programação

MATLAB e testada com exemplos gerados aleatoriamente. Na geração aleatória dos dados, consideram-se diferentes valores para vários parâmetros, de forma a obter diferentes cenários com o objetivo de simular a realidade.

As rotas obtidas através da heurística *GRASP*, desenvolvida neste trabalho de projeto, e a geração aleatória dos dados respeitam os princípios dos circuito curtos e de proximidade. Ou seja, o número de intermediários e de *food miles* deve ser mínimo. Nos circuitos curtos o número de intermediários deve ser um ou zero, deste modo, as rotas construídas consistem em rotas com pontos de recolha e distribuição. A recolha é feita diretamente nos agricultores ou nos mercados onde os agricultores fazem a venda direta dos produtos, e a distribuição é realizada nos clientes. Os clientes podem ser o próprio consumidor, e neste caso não existem intermediários, ou podem ser por exemplo, restaurantes, escolas, mercearias, lojas de organização de produtos e cabazes, e neste caso existe apenas um intermediário. Além disso, as rotas não incluem armazenamento, garantindo assim que a distribuição é realizada imediatamente após a recolha. Nos circuitos de proximidade a distância entre o local de produção e venda deve ser mínima. Deste modo, a heurística *GRASP* aplica-se a um problema que tem como função objetivo minimizar o custo, que neste caso se considera proporcional à distância total percorrida. Neste projeto considera-se que a constante de proporcionalidade é um, pelo que, o custo é igual à distância percorrida. Além disso, na geração dos dados são consideradas áreas pequenas, ou seja, as distâncias entre agricultores/mercados e clientes são pequenas.

Este trabalho de projeto está divido em cinco capítulos, incluindo este. No capítulo 2 descreve-se o modelo em programação linear inteira mista para o problema de determinação de rotas, para vários veículos, com pontos de recolha e distribuição de produtos em circuitos curtos e de proximidade.

No capítulo 3 efetua-se uma breve descrição da heurística *GRASP*. O *GRASP* é uma meta-heurística iterativa, em que cada iteração consiste em duas fases: construção e pesquisa local. O *GRASP* apenas depende de dois parâmetros: número de iterações e o parâmetro α. Também se descreve as duas versões da heurística *GRASP* desenvolvidas para o problema em estudo.

No capítulo 4, descreve-se os dados gerados aleatoriamente, considerando vários valores para diversos parâmetros, de forma a obter diversos cenários com o objetivo de simular a realidade. Apresenta-se o valor usado nos testes computacionais para os parâmetros necessários para a implementação da heurística. Comparam-se as duas versões do *GRASP*, desenvolvidas neste projeto. Os resultados computacionais, obtidos usando a heurística *GRASP*, são apresentados de forma resumida. Uma análise dos resultados computacionais é efetuada. No capítulo 5 apresentam-se as conclusões finais obtidas.

Capítulo 2

Apresentação do problema e do modelo em programação linear inteira mista

Neste secção apresenta-se o problema em estudo, ou seja, o problema de determinação de rotas com recolha e distribuição de produtos alimentares. O modelo em programação linear inteira mista é apresentado para o problema acima mencionado.

O problema em estudo tem como objetivo determinar rotas com recolha dos produtos alimentares nos mercados e imediata distribuição nos clientes. Para simplificar, ao longo deste capítulo 2 e dos capítulos 3, 4 e 5, vai-se referir apenas mercados, em vez de agricultores e mercados onde os agricultores vendem os seus produtos diretamente. Os clientes podem ser, por exemplo: vendas *online*, restaurantes, escolas, mercearias, lojas de organização de produtores, cabazes, entre outros. O VRPPD é da família do problema de determinação de rotas para veículos, VRP. Resolver o VRPPD consiste em determinar rotas tais que de acordo com [9] devem verificar:

- cada rota visita o depósito, de onde partem e chegam os veículos;
- cada cliente é visitado por exatamente um veículo, isto é, pertence a uma só rota;
- a carga do veículo ao longo da rota é não negativa e não pode exceder a capacidade do veículo;
- cada cliente pertence à mesma rota que o(s) mercado(s) que o serve(m) e aparece na rota depois deste(s);
- o(s) mercado(s) que serve(m) o cliente pertence(m) à mesma rota que este e antes deste na rota.

O modelo em programação linear inteira apresentado para o problema de determinação de rotas de recolha e distribuição de produtos agrícolas, em que não é obrigatório visitar todos os mercados, baseia-se no proposto por Salvelsberg e Sol, em [8], para o problema de determinação de rotas de recolha e distribuição de produtos com janelas temporais e no apresentado por Oliveira, em [4], para um problema de planeamento da distribuição de produtos agrícolas num canal curto de distribuição.

Algumas hipóteses:

- Os veículos estão disponíveis num depósito e iniciam e terminam a rota nesse depósito;
- Em cada rota um mercado é visitado no máximo uma vez;

2. APRESENTAÇÃO DO PROBLEMA E DO MODELO EM PROGRAMAÇÃO LINEAR INTEIRA MISTA

- O tempo total da rota está de acordo com as restrições relativas ao horário dos motoristas dos veículos. Assim, cada veículo depois de visitar os mercados e os correspondestes clientes volta ao depósito;
- A procura dos clientes deve ser satisfeita;
- A disponibilidade de cada produto em cada mercado não pode ser excedida;
- A frota de veículos é homogénea, isto é, todos os veículos têm as mesmas características, pelo que existe uma capacidade máxima igual para todos os veículos.

Notação usada

- P - conjunto de produtos
- Cl - conjunto de clientes
- M - conjunto de mercados
- q_{pk} - procura do produto $p \in P$ relativa ao cliente $k \in Cl$
- a_{pj} - quantidade de produto $p \in P$ disponível no mercado $j \in M$
- VH - conjunto de veículos
- $CapVeq$ - capacidade comum a todos os veículos $v \in VH$.

Considere-se o grafo $G = (X, A)$ em que $X = Cl \cup M \cup O$. O representa o depósito. Consideram-se as réplicas O^+ e O^- do depósito que representam o depósito respetivamente à partida e chegada dos veículos. $A = \{(i, j) : i, j \in X\}$.

Como o número de veículos é estabelecido a priori, podem não ser todos usados. Assim, considera-se um arco entre a réplica do depósito que o representa à partida, O^+, para a réplica do depósito que o representa à chegada O^-. Assim, $A \leftarrow A \cup \{(O^+, O^-)\}$. Alguns dos arcos de G podem ser eliminados do conjunto A.

- Os veículos saem do depósito e os primeiros locais a visitar são mercados. Assim, em A só se consideram os arcos $\{(O^+, j) : j \in M\}$;
- De modo idêntico a chegada ao depósito é feita sempre a partir de clientes. Assim, só se consideram em A os arcos $\{(k, O^-) : k \in Cl\}$;
- Não existem *loops*. Os arcos (i, j) com $i = j$ não pertencem a A;
- d_{ij} representa a distância entre i e j, $(i, j) \in A$.

Variáveis de decisão

- $z_k^v = \begin{cases} 1 & \text{se o cliente } k \in Cl \text{ é visitado pelo veículo } v \in VH \\ 0 & \text{caso contrário} \end{cases}$
- $g_{jk}^v = \begin{cases} 1 & \text{se o mercado } j \in M \text{ e o cliente } k \in Cl \text{ pertencem à rota do veículo } v \in VH \\ 0 & \text{caso contrário} \end{cases}$

- $x_{ls}^v = \begin{cases} 1 & \text{se o veículo } v \in VH \text{ atravessa o arco } (l,s) \in A \\ 0 & \text{caso contrário} \end{cases}$
- f_{pjk}^v - quantidade de produto $p \in P$ comprado no mercado $j \in M$ para o cliente $k \in Cl$ transportado pelo veículo $v \in VVH$
- y_l^v - carga do veículo $v \in VH$ quando chega a $l \in X \cup O$

Como o número de veículos é estabelecido a priori, tem-se $x_{O^+O^-}^v \neq 0$ se o veículo $v \in VH$ não é usado. Sendo uma frota homogénea em que a capacidade de cada veículo é a mesma e igual a $Capveq$, tem-se que o número mínimo de veículos é $\mid VH \mid$ que verifica [9]

$$\frac{\sum_{p\in P}\sum_{k\in Cl} q_{pk}}{CapVeq} \leqslant \mid VH \mid$$

O modelo em Programação linear inteira mista é o seguinte:

$$\min \sum_{v\in VH}\sum_{(l,s)\in A} d_{ls}x_{ls}^v \tag{2.1}$$

$s.t$

$$\sum_{v\in VH} z_k^v = 1 \qquad \forall k \in Cl \tag{2.2}$$

$$g_{jk}^v \leq z_k^v \qquad \forall k \in Cl, \forall j \in M, v \in VH \tag{2.3}$$

$$\sum_{s:(j,s)\in A} x_{js}^v = g_{jk}^v \qquad \forall j \in M, \forall k \in Cl, v \in VH \tag{2.4}$$

$$\sum_{s:(k,s)\in A} x_{ks}^v = z_k^v \qquad \forall k \in Cl, v \in VH \tag{2.5}$$

$$\sum_{s:(l,s)\in A} x_{ls}^v - \sum_{s:(s,l)\in A} x_{sl}^v = 0 \qquad \forall l \in X, v \in VH \tag{2.6}$$

$$\sum_{s\in M\cup O^-} x_{O^+s}^v = 1 \qquad \forall v \in VH \tag{2.7}$$

$$\sum_{s\in Cl\cup O^+} x_{sO^-}^v = 1 \qquad \forall v \in VH \tag{2.8}$$

$$f_{pjk}^v \leq \bar{U}(g_{jk}^v + 1 - z_k^v) \qquad \forall p \in P, j \in M, k \in Cl, v \in VH \tag{2.9}$$

$$\sum_{v\in VH}\sum_{k\in Cl} f^v_{pjk} \leq a_{pj} \qquad \forall p \in P, j \in M \qquad (2.10)$$

$$\sum_{v\in VH}\sum_{j\in M} f^v_{pjk} = q_{pk} \qquad \forall p \in P, k \in Cl \qquad (2.11)$$

$$y^v_{O^+} = 0 \qquad \forall v \in VH \qquad (2.12)$$

$$y^v_{O^-} = 0 \qquad \forall v \in VH \qquad (2.13)$$

$$y^v_j + \sum_{p\in P} f^v_{pjk} \leq y^v_s + CapVeq(1-x^v_{js}) \qquad \forall v \in VH, j \in M \cup O^+, k \in Cl, s : (j,s) \in A \qquad (2.14)$$

$$y^v_k - \sum_{j\in M}\sum_{p\in P} f^v_{pjk} \leq y^v_s + CapVeq(1-x^v_{ks}) \qquad \forall v \in VH, k \in Cl \cup O^-, s : (k,s) \in A \qquad (2.15)$$

$$\sum_{v\in VH} CapVeqz^v_k \geq y^v_k \qquad \forall k \in Cl, \forall v \in VH \qquad (2.16)$$

$$\sum_{v\in VH} CapVeqg^v_{jk} \geq y^v_j \qquad \forall j \in M, \forall k \in Cl, \forall v \in VH \qquad (2.17)$$

Eliminação de subcircuitos (2.18)

$$z^v_k \in \{0,1\} \qquad \forall k \in Cl, v \in VH \qquad (2.19)$$

$$g^v_{jk} \in \{0,1\} \qquad \forall j \in M, \forall k \in Cl, v \in VH \qquad (2.20)$$

$$x^v_{ls} \in \{0,1\} \qquad \forall (l,s) \in A, v \in VH \qquad (2.21)$$

$$f^v_{pjk} \geq 0 \qquad \forall p \in P, j \in M, k \in Cl \qquad (2.22)$$

$$y^v_l \geq 0 \qquad \forall l \in X \qquad (2.23)$$

onde $\bar{U}$ é um número suficientemente grande de modo a não impor um limite às variáveis.

A função objetivo (2.1) é a minimização da distância total das rotas.
As restrições (2.2) garantem que cada cliente é visitado por um único veículo. As restrições (2.3) garantem que se um veículo não visita um cliente então não pode visitar os mercados que lhe estão associados. As restrições (2.4) e (2.5) garantem que se um mercado ou cliente, respetivamente, é visitado por um veículo este sai do mercado ou cliente, respetivamente, apenas uma vez. As restrições (2.6) são restrições de conservação do fluxo, isto é, se um veículo chega a um local também tem que sair dele. As restrições (2.7) e (2.8) garantem que cada veículo inicia e termina a viagem no depósito. Depois de sair do depósito cada veículo visita um mercado e chega ao depósito depois de visitar um cliente ou vai

diretamente do depósito origem para o depósito destino se não for usado. As restrições (2.9) garantem que se um veículo visita um cliente e não visita determinado mercado então este mercado não fornece qualquer produto para esse cliente. As restrições (2.10) garantem que a disponibilidade de produtos nos mercados não é excedida e as restrições (2.11) garantem que a procura de cada cliente é satisfeita. As restrições (2.12) e (2.13) garantem que o veículo parte e respetivamente chega ao depósito vazio. As restrições (2.14) e (2.15) dizem respeito à carga do veículo quando chega a cada local. As restrições (2.16) e (2.17) garantem que a capacidade de cada veículo não é excedida. As restrições (2.19) até (2.23) definem o domínio das variáveis.

As restrições (2.18) dizem respeito à eliminação de subcircuitos. Existem muitas versões polinomiais para este tipo de restrições. Oncan, Altimel e Laporte [10] apresentam várias versões para estas restrições no caso do problema do caixeiro viajante e fazem uma análise das formulações resultantes. Neste artigo referem as restrições apresentadas por Sarin *et al* [11] que são usadas por Rais *et al* [12] para o problema de determinação de rotas de recolha e distribuição com pontos de transfega. Estas são as desigualdades usadas neste modelo para garantir a eliminação de subcircuitos.
Definem-se as variáveis

$$w_{ij}^{v} = \begin{cases} 1 & \text{se o vértice } i \text{ precede o vértice } j \text{ na rota do veículo } v \\ 0 & \text{caso contrário} \end{cases}$$

$$i \in X \setminus \{O\}, j \in X \setminus \{O\}, v \in VH$$

Tem-se

$$x_{ij}^{v} \leq w_{ij}^{v} \qquad \forall i, j \in X \setminus \{O\}, v \in VH \tag{2.24}$$

$$w_{ij}^{v} + w_{ji}^{v} = 1 \qquad \forall i, j \in X \setminus \{O\}, v \in VH \tag{2.25}$$

$$w_{ij}^{v} + w_{jl}^{v} + w_{li}^{v} \leq 2 \qquad \forall i, j, l \in X \setminus \{O\}, v \in VH \tag{2.26}$$

Capítulo 3

Greedy Randomized Adaptive Search Procedures (GRASP)

3.1 Introdução

Problemas de otimização que contêm um número grande, mas finito de soluções admissíveis, ocorrem frequentemente na indústria, na ciência e ao nível das decisões governamentais. Soluções admissíveis são as soluções que verificam as condições exigidas num problema. Nesses problemas, é teoricamente possível enumerar todas as soluções admissíveis, sendo a melhor denominada solução ótima. Contudo, na prática, tal processo é inviável, uma vez que, o número de soluções admissíveis geralmente cresce, exponencialmente, com a dimensão do problema. Deste modo, houve a necessidade de desenvolver métodos que não exigiam a determinação de todas as soluções admissíveis. Isso conduziu ao desenvolvimento de várias heurísticas, mais especificamente, várias meta-heurísticas.

Meta-heurísticas são procedimentos que coordenam heurísticas simples e regras, para encontrar soluções de boa qualidade para problemas de otimização combinatória, computacionalmente difíceis. Exemplos de meta-heurísticas incluem *simulated annealing*, *tabu search*, *genetic algorithms* e *GRASP (Greedy Randomized Adaptive Search Procedures*).

No âmbito deste trabalho de projeto, utiliza-se a meta-heurística *GRASP* para determinar rotas de recolha e distribuição de produtos agrícolas. Os produtos alimentares são recolhidos nos mercados e são posteriormente distribuídos pelos vários clientes (exemplos: vendas *online*, restaurantes, escolas, mercearias, lojas de organização de produtores, cabazes). A meta-heurística *GRASP* aplica-se, neste trabalho, ao VRPPD.

O *GRASP* é uma meta-heurística iterativa, em que cada iteração consiste em duas fases: construção e pesquisa local. Na fase de construção uma solução é construída, adicionando-se um elemento de cada vez à solução. Cada elemento é selecionado aleatoriamente de uma lista de candidatos determinada por uma função *greedy*, adaptativa. Deste modo, a cada iteração do *GRASP* uma solução diferente é determinada. Caso a solução encontrada não seja admissível, procedimentos devem ser aplicados de forma a obter uma solução admissível. Contudo, caso tal não seja possível, deve-se descartar a solução e iniciar de novo o processo de construção. No presente projeto vai-se sempre construir soluções admissíveis. Dada uma vizinhança simples, a solução construída na fase de construção, não é garantidamente, localmente ótima. Logo é quase sempre benéfico aplicar uma pesquisa local, de maneira a tentar melhorar cada solução obtida na fase de construção.

A pesquisa local funciona de maneira iterativa, substituindo sucessivamente a solução atual por uma

solução melhor na vizinhança da solução atual. Esta fase termina, quando não é possível encontrar uma solução melhor na vizinhança. Soluções vizinhas são as soluções que são possíveis de obter aplicando modificações elementares à solução inicial. A qualidade da solução obtida, através da pesquisa local, depende da estrutura da vizinhança, da eficiência das técnicas de pesquisa das vizinhanças e da solução inicial,

Uma característica que torna o *GRASP* uma heurística apelativa, é a facilidade com que é implementada. Na implementação do *GRASP*, apenas dois parâmetros necessitam de ser definidos: o número de iterações e o parâmetro *alpha*, que determina a qualidade dos elementos na lista de candidatos. Deste modo, o desenvolvimento da heurística pode concentrar-se na implementação de estruturas de dados eficientes, de forma a assegurar iterações rápidas. Outra característica apelativa do *GRASP*, é o facto de poder ser implementado, trivialmente, em paralelo. Cada processador pode ser inicializado com uma cópia do procedimento, com uns dados da instância e com uma sequência numérica aleatória independente. As iterações do GRASP são então executadas em paralelo, com apenas uma variável global necessária para armazenar a melhor solução encontrada, em todos os processadores.

Uma desvantagem do uso do *GRASP* básico, é o facto de ser um procedimento sem memória. Isto significa que cada iteração da heurística é independente da outra, ou seja, a heurística não usa a informação recolhida da iteração anterior. Tal acontece, pois o *GRASP* básico descarta informação sobre alguma solução que não melhore a solução atual. Vários melhoramentos do *GRASP* básico foram desenvolvidos, tais como *path relinking*, *long-term memory*, *reactive GRASP*, entre outros [13].

Mauricio G. C. Resende em [14], indica vários tipos de problemas onde a heurística *GRASP* pode ser aplicada. Alguns dos problemas indicados em [14] incluem: problemas de escalonamento (escalonamento de voos e de condutores de autocarros), problemas de determinação de rotas (rotas de aviões e de veículos com janelas temporais), problemas em teoria de grafos (problemas de cobertura de conjuntos e determinação de conjuntos independentes máximos) e problemas de produção (seleção dos equipamentos num sistema produtivo).

Thomas A. Feo e Mauricio G. C. Resende em [15], indicam várias aplicações do *GRASP*. De seguida, apresentam-se alguns exemplos de aplicações do *GRASP* mencionadas em [15]. Laguna e González-Velarde aplicaram a heurística ao sequenciamento de máquinas paralelas no ambiente de produção *just-in-time*. Feo, Venkatraman e Bard desenvolveram o *GRASP* para problemas de sequenciamento de uma única máquina, com tempo de fluxo e penalidades por antecipação. Feo e Smith aplicaram o *GRASP* na coloração de grafos esparsos. Resende e Feo descreveram várias implementações do *GRASP* para o problema da satisfatibilidade.

3.2 Descrição da heurística GRASP desenvolvida para o problema em estudo

Nesta secção descreve-se a heurística *GRASP* desenvolvida para o problema apresentado neste projeto. A heurística desenvolvida determina rotas que permitem a recolha e posterior distribuição de produtos alimentares. O pontos de recolha são os mercados e os pontos de distribuição são os clientes (exemplos: vendas *online*, restaurantes, escolas, mercearias, lojas de organização de produtores, cabazes).

A heurística *GRASP*, cuja descrição genérica se apresenta de seguida, tem como objetivo determinar rotas, que incluem pontos de distribuição (clientes) e recolha (mercados), de forma a minimizar o custo total. O custo é proporcional à distância entre os intervenientes (depósito, mercados e clientes). Neste projeto considera-se que a constante de proporcionalidade é um, pelo que, o custo é igual à distância

percorrida. A heurística *GRASP* apresenta-se no Algoritmo 1.

Algorithm 1 GRASP

```
1:  procedure GRASP(ITERACOES.MAX)
2:      Set f* ← +∞;
3:      for k =1,...,Iteracoes.max do
4:          S ← GreedyRandomizedConstruction(α);
5:          S ← LocalSearchClientes(S);
6:          (R,T) ← AddPickUpNodes(S);
7:          T ← LocalSearchMercados(S,R,T);
8:          T ← LocalSearchClientesMercados(T);
9:          if f(T) < f* then
10:         T* ← T;
11:         f* ← f(T);
12:     end;
13:     end;
14:     return T*;
    end.
```

A heurística *GRASP* desenvolvida neste trabalho, como indicado na secção 3.1, considera sempre a obtenção de soluções admissíveis em cada iteração. Em cada iteração da heurística determina-se, através do algoritmo *GreedyRandomizedConstruction*, uma solução S que inclui apenas clientes. De seguida, vai-se tentar melhorar a solução S recorrendo ao algoritmo *LocalSearchClientes*. Depois determina-se os pontos de recolha para a solução S, já melhorada, com a implementação do algoritmo *AddPickUpNodes*. Este algoritmo permite obter as soluções R (inclui apenas mercados) e T (inclui clientes e mercados), a partir da solução S que inclui apenas clientes. Posteriormente vai-se tentar melhorar a solução R (rotas só com mercados) com recurso ao algoritmo *LocalSearchMercados*. De seguida, vai-se tentar melhorar a solução T através da implementação do algoritmo *LocalSearchClientesMercados*. Seja T^* a melhor solução encontrada até ao momento, define-se $f(T)$ e f^* como os custos totais associados às soluções T e T^*, respetivamente. Em cada iteração, compara-se o custo total associado a ambas as soluções. Caso a solução obtida T seja melhor que T^*, ou seja, caso T tenha um custo total associado menor que T^*, atualiza-se T^* com o valor de T. Caso contrário, não se altera T^*. Este processo é repetido um número de iterações pré-estabelecido (*Iteracoes.max*).

Os algoritmos *GreedyRandomizedConstruction* e *AddPickUpNodes* constituem a fase de construção, enquanto que os algoritmos *LocalSearchClientes*, *LocalSearchMercados* e *LocalSearchClientesMercados* representam a fase de pesquisa local.

3.2.1 Fase de construção

Nesta secção descreve-se a fase de construção, de uma solução admissível, que inclui os algoritmos *GreedyRandomizedConstruction* (ver secção 3.2.1.1) e *AddPickUpNodes* (ver secção 3.2.1.2).

3.2.1.1 *GreedyRandomizedConstruction*

O algoritmo *GreedyRandomizedConstruction (GRC)* apresenta-se no Algoritmo 2. Este algoritmo permite obter as rotas só com clientes, para o problema apresentado.

No algoritmo *GRC*, de maneira a obter os pontos de distribuição, considera-se

$$E = \{\text{todos os clientes}\},$$

Algorithm 2 GreedyRandomizedConstruction (GRC)

procedure GREEDYRANDOMIZEDCONSTRUCTION(α)
  $S \leftarrow \emptyset$;
  Inicializar o conjunto de candidatos: $C \leftarrow E$;
  Determinar o custo incremental $c(e)$ para todo $e \in C$;
  $i \leftarrow 1$;
  $Nrotas \leftarrow 1$;
  $S_i \leftarrow \emptyset$;
  $SumProcura(S_i) \leftarrow 0$;
  while $C \neq \emptyset$ **do**
    $c^{min} \leftarrow \min\{c(e)|e \in C\}$;
    $c^{max} \leftarrow \max\{c(e)|e \in C\}$;
    Construir a lista de candidatos restrita:
    $RCL \leftarrow \{e \in C|c(e) \leq c^{min} + \alpha(c^{max} - c^{min})\}$;
    Escolher s aleatoriamente da RCL;
    if $SumProcura(S_i) + newProc.s \leq CapVeq$ **then**
      Incorporar s na solução: $S_i \leftarrow S_i \cup \{s\}$;
      $SumProcura(S_i) \leftarrow SumProcura(S_i) + new.Proc.s$;
      Atualizar o conjunto de candidatos: $C \leftarrow C \setminus \{s\}$;
      Recalcular o custo incremental $c(e)$ para todo $e \in C$;
    else
      $i \leftarrow i + 1$;
      $Nrotas \leftarrow Nrotas + 1$;
      $S_i \leftarrow \emptyset$;
      $SumProcura(S_i) \leftarrow 0$;
      Recalcular o custo incremental $c(e)$ para todo $e \in C$;
    end;
  end;
  return $S = \bigcup_{i \in \{1,\ldots,Nrotas\}} S_i$;
end.

S_i : rota constituída só por clientes, $\forall i \in \{1,...,Nrotas\}$

onde *Nrotas* é o número de rotas criadas. Define-se também

$$c(e), e \in E$$

o custo incremental associado à inserção de um cliente na rota S_i. Neste caso, os valores dos custos incrementais calculam-se considerando as distâncias percorridas entre clientes e entre clientes e o depósito. Ter em conta que na primeira iteração os custos a considerar devem ser a soma das distâncias de ir do depósito ao cliente e vice-versa. Posteriormente, os custos a analisar são da inserção de um cliente após o anteriormente adicionado. Isto é, devem-se avaliar a soma das distâncias de ir do cliente anteriormente adicionado ao cliente candidato a inserção e deste ao depósito.

Em cada iteração, c^{min} e c^{max} são respetivamente, o valor mínimo e o máximo associado a *c(e), e* $\in$ *E.* A *Lista de Candidatos Restrita (RCL)* é constituída por elementos $e \in E$ tal que satisfazem a seguinte desigualdade

$$c(e) \leq c^{min} + \alpha(c^{max} - c^{min})$$

onde o valor de α pode variar no intervalo $[0,1]$. O caso de $\alpha = 0$ corresponde a um algoritmo puramente *greedy*, enquanto que $\alpha = 1$ equivale a um algoritmo completamente aleatório. Mauricio G. C. Resende e Celso C. Ribeiro em [16] consideram que o valor para α, que em geral, determina melhores soluções, é o valor para α de 0.2.

Define-se a *SumProcura*(S_i), como a procura total dos clientes presentes na rota S_i, *newProc.s* a procura do cliente *s* e *CapVeq* a capacidade total do veículo. A cada iteração, escolhe-se aleatoriamente um elemento *s* $\in$ *RCL.* Contudo antes de adicionar o elemento *s* na rota S_i, é necessário que se verifique a seguinte desigualdade

$$SumProcura(S_i)+newProc.s \leq CapVeq.$$

Caso se verifique tal desigualdade, adiciona-se *s* à rota S_i. Além disso, atualiza-se a *SumProcura*(S_i), o conjunto E e o valor de $c(e), e \in E$. Caso contrário, inicia-se uma nova rota e atualiza-se o valor $c(e), e \in E$. Este processo é repetido enquanto existirem clientes para adicionar, ou seja, enquanto $E \neq \emptyset$. A solução final, *S*, é o resultado da união de todos os S_i, i $\in$ {1,...,Nrotas}.

3.2.1.2 *AddPickUpNodes*

A cada S_i, rota só com clientes, $i \in \{1,...,Nrotas\}$, vai-se adicionar os pontos de recolha. Para tal implementa-se o algoritmo *AddPickUpNodes*. Vai-se considerar para este algoritmo duas opções de escolha dos mercados. A primeira opção do algoritmo, denominada *AddPickUpNodes1 (APUN1)* (ver secção 3.2.1.2.1), permite determinar quais os mercados a visitar de maneira a satisfazer a procura dos clientes presentes em $S_i, i \in \{1,...,Nrotas\}$ de forma totalmente aleatória. A segunda opção, denominada *AddPickUpNodes2 (APUN2)* (ver secção 3.2.1.2.2), adiciona na rota os pontos de recolha com base no custo incremental. Para simplificar, ao longo desta secção 3.2, vai-se referir apenas *AddPickUpNodes (APUN)* sem fazer distinção entre as duas versões.

3.2.1.2.1 *AddPickUpNodes1*

O algoritmo *APUN1* apresenta-se no Algoritmo 3. Este algoritmo tem como objetivo determinar quais os pontos de recolha a incluir em cada rota só com clientes, sendo a escolha feita de forma totalmente aleatória.

Algorithm 3 AddPickUpNodes1 (APUN1)

procedure ADDPICKUPNODES(S)
  $R \leftarrow \emptyset$;
  $T \leftarrow \emptyset$;
  Inicializar o conjunto de produtos: $P \leftarrow A$;
  Inicializar o conjunto de candidatos: $M \leftarrow B$;
  Determinar a oferta de cada mercado i referente a cada produto j : $OM_{ij}, i \in M$ e $j \in P$;
  $NS \leftarrow Q$;
  for $k \in NS$ **do**
    $R_k \leftarrow \emptyset$;
    $T_k \leftarrow \emptyset$;
    Determinar os produtos e a respetiva procura presente na rota S_k: PS_k;
    while $PS_k \neq \emptyset$ **do**
      Escolher p aleatoriamente de PS_k;
      Construir a lista de mercados candidatos do produto p : $MP \leftarrow \{m \in M | m \text{ oferece o produto } p\}$;
      Escolher m aleatoriamente de MP;
      Incorporar m na rota: $R_k \leftarrow R_k \cup \{m\}$;
      if m satisfaz na totalidade a procura do produto p **then**
        Atualizar o conjunto PS_k: $PS_k \leftarrow PS_k \setminus \{produto_p, procura_p\}$;
      else
        Atualizar a procura do produto p no conjunto PS_k;
      end;
      if m oferece outros produtos em PS_k **then**
        Satisfazer, tendo em conta a disponibilidade do mercado m, a procura de tais produtos;
      end;
      Atualizar o conjunto OM_{ij};
  end;
  $T_k = R_k \cup S_k$;
  end;
  return $R = \bigcup_{w \in \{1,\ldots,Nrotas\}} R_w$;
  return $T = \bigcup_{w \in \{1,\ldots,Nrotas\}} T_w$;
end.

No algoritmo *APUN1*, define-se

A={todos os produtos},

B={todos os mercados}

e

$$OM_{ij} = \{\text{oferta de cada mercado } i \text{ para cada produto } j\},\ \forall i \in B \text{ e } \forall j \in A$$

No algoritmo *GRC*, obtém-se uma solução

$$S = \bigcup_{w \in \{1,...,Nrotas\}} S_w$$

onde S_w é o conjunto de clientes a visitar e $w \in \{1,...,Nrotas\}$, em que $Nrotas$ é o número de rotas criadas em S. Define-se

$$Q = \{\text{números de 1 a } Nrotas \text{ ordenados aleatoriamente}\}.$$

Desta forma, as rotas são escolhidas aleatoriamente para inserção dos pontos de recolha. Para cada $k \in Q$, isto é, para cada S_k, define-se

$$R_k\text{: rota constituída só por mercados}$$

$$T_k\text{ : rota de recolha e distribuição,}$$

isto é,

$$T_k = R_k \cup S_k$$

e

$$PS_k = \{\text{todos os produtos em } S_k \text{ cuja procura é diferente de zero}\}.$$

Para cada k, o conjunto PS_k vai ser da seguinte forma

$$PS_k = \{(produto_1, procura_1), (produto_2, procura_2), ...\}$$

Escolhe-se aleatoriamente um produto $p \in PS_k$. Para o produto p constrói-se a *Lista de Mercados Candidatos do produto p (MP)*. O vetor MP é constituído por elementos $m \in B$, tal que o mercado m oferece o produto p. Em seguida, escolhe-se aleatoriamente um elemento $m \in MP$ e adiciona-se após o último mercado inserido na rota R_k. A cada iteração é necessário verificar se o mercado m satisfaz na totalidade a procura do produto p. Caso tal se verifique, retira-se o produto p do conjunto PS_k. Caso contrário, atualiza-se a procura do produto p no conjunto PS_k. Caso o mercado m ofereça outros produtos presentes no conjunto PS_k, então a procura de tais produtos deve ser satisfeita, tendo em conta a disponibilidade do mercado. Após cada iteração é necessário atualizar a oferta do mercado m, ou seja, atualizar o conjunto $OM_{ij}, i \in B$ e $j \in A$.

Para cada rota k, o processo é repetido até que a procura de cada rota seja satisfeita, ou seja, enquanto $PS_k \neq \emptyset$. Após tal processo terminar ($PS_k = \emptyset$) obtém-se, para cada $k \in Q$, T_k. T_k é o resultado da união de R_k e S_k. Na rota T_k, os elementos de R_k estão todos antes dos elementos de S_k. Depois da construção de todas as T_k rotas, obtém-se a solução final T, que resulta da união de todos os T_w, $w \in \{1,...,Nrotas\}$. Também se obtém a solução $R = \bigcup_{w \in \{1,...,Nrotas\}} R_w$, tal que, as rotas em R contêm apenas os mercados.

3.2.1.2.2 *AddPickUpNodes2*

O algoritmo *APUN2* apresenta-se no Algoritmo 4. Este algoritmo tem como objetivo determinar os pontos de recolha com base no custo incremental.

O algoritmo *APUN2* apresentado é semelhante ao já descrito na secção 3.2.1.2.1, contudo difere, daquele em alguns aspetos. Define-se como

$$c(m), m \in B$$

o custo incremental associado à inserção de um mercado na rota R_k. Neste caso, os valores dos custos incrementais calculam-se considerando as distâncias percorridas entre clientes e mercados, entre mercados e o depósito e entre mercados. Ter em conta, que na primeira iteração os custos a considerar devem ser a soma das distâncias de ir do depósito ao mercado e deste ao primeiro cliente presente em S_k. Posteriormente, os custos a analisar são da inserção de um mercado após o anteriormente adicionado, isto é, somam-se as distâncias de ir do mercado anteriormente adicionado ao mercado candidato a inserção e deste ao primeiro cliente em S_k.

Como no *APUN1*, começa-se por escolher aleatoriamente um produto $p \in PS_k$ e constrói-se a *Lista de Mercados Candidatos do produto p (MP)*. Em cada iteração do algoritmo, c^{min} e c^{max} são respetivamente, o valor mínimo e o máximo associado a *c(m)*, $m \in B$. Depois, constrói-se a lista de candidatos restrita *RCL*. Esta é constituída por elementos $r \in MP$ tal que satisfazem a seguinte desigualdade

$$c(r) \leq c^{min} + \alpha(c^{max} - c^{min})$$

onde o valor de α varia no intervalo [0,1]. O valor para α usado no $APUN2$ vai ser igual ao usado no algoritmo *GRC*.

Em seguida, escolhe-se aleatoriamente um elemento $r \in RCL$, e adiciona-se após o último mercado inserido na rota R_k. O processo seguinte é igual ao do $APUN1$, contudo é necessário atualizar o valor de $c(m), m \in B$.

Tal como no *APUN1*, para cada rota k, o processo é repetido até que a procura de cada rota seja satisfeita, ou seja, enquanto $PS_k \neq \emptyset$. Após tal processo terminar ($PS_k = \emptyset$) obtém-se, para cada $k \in Q$, T_k. T_k é o resultado da união de R_k e S_k. Depois da construção de todas as T_k rotas, obtém-se a solução T, que resulta da união de todos os $T_w, w \in \{1, ..., Nrotas\}$. Também se obtém a solução $R = \bigcup_{w \in \{1,...,Nrotas\}} R_w$, tal que, as rotas em R contêm apenas mercados.

Algorithm 4 AddPickUpNodes2 (APUN2)

procedure ADDPICKUPNODES(S)
  $R \leftarrow \emptyset$;
  $T \leftarrow \emptyset$;
  Inicializar o conjunto de produtos: $P \leftarrow A$;
  Inicializar o conjunto de candidatos: $M \leftarrow B$;
  Determinar a oferta de cada mercado i referente a cada produto j : $OM_{ij}, i \in M$ e $j \in P$;
  Determinar o custo incremental $c(m)$ para todo $m \in M$;
  $NS \leftarrow Q$;
  for $k \in NS$ **do**
    $R_k \leftarrow \emptyset$;
    $T_k \leftarrow \emptyset$;
    Determinar os produtos e a respetiva procura presente na rota S_k: PS_k;
    while $PS_k \neq \emptyset$ **do**
      Escolher p *aleatoriamente de* PS_k;
      Construir a lista de mercados candidatos do produto p : $MP \leftarrow \{m \in M | m \text{ oferece o produto } p\}$;
      $c^{min} \leftarrow \min\{c(m) | m \in MP\}$;
      $c^{max} \leftarrow \max\{c(m) | m \in MP\}$;
      Construir a lista de candidatos restrita: $RCL \leftarrow \{r \in MP | c(r) \leq c^{min} + \alpha(c^{max} - c^{min})\}$;
      Escolher r aleatoriamente da RCL;
      Incorporar r na rota: $R_k \leftarrow R_k \cup \{r\}$;
      if r satisfaz na totalidade a procura do produto p **then**
        Atualizar o conjunto PS_k: $PS_k \leftarrow PS_k \setminus \{produto_p, procura_p\}$;
      else
        Atualizar a procura do produto p no conjunto PS_k;
      end;
      if r oferece outros produtos em PS_k **then**
        Satisfazer, tendo em conta a disponibilidade do mercado r, a procura de tais produtos;
      end;
      Atualizar o conjunto OM_{ij};
      Recalcular o custo incremental $c(m)$ para todo $m \in M$;
  end;
  $T_k = R_k \cup S_k$;
  end;
  return $R = \bigcup_{w \in \{1,\ldots,Nrotas\}} R_w$;
  return $T = \bigcup_{w \in \{1,\ldots,Nrotas\}} T_w$;
end.

3.2.2 Fase de pesquisa local

Nesta secção descreve-se a fase de pesquisa local, que inclui os algoritmos *LocalSearchClientes* (ver secção 3.2.2.1), *LocalSearchMercados* (ver secção 3.2.2.2) e *LocalSearchClientesMercados* (ver secção 3.2.2.3).

3.2.2.1 LocalSearchClientes

O algoritmo *LocalSearchClientes (LSC)* apresenta-se no Algoritmo 5. Este algoritmo tenta melhorar a solução S obtida através do algoritmo *GRC*, tal que, $S = \bigcup_{i=\{1,...,Nrotas\}} S_i$. Este melhoramento vai tentar reinserir os clientes com recurso à heurística de inserção de menor custo. O algoritmo *LSC* é implementado após o algoritmo *GRC*, desta forma, o melhoramento local nas rotas só com clientes, é efetuada antes de adicionar os mercados.

Algorithm 5 LocalSearchClientes (LSC)

1: **procedure** LOCALSEARCHCLIENTES(S)
2: **for** `i = 1,...,Nrotas` **do**
3: `Inicializar o conjunto dos clientes da rota` S_i: $CC_i \leftarrow H_i$;
4: $RT_i \leftarrow \emptyset$;
5: `Determinar o custo incremental` $c(e)$ `para todo` $e \in CC_i$;
6: **while** $CC_i \neq \emptyset$ **do**
7: $c^{min} \leftarrow min\{c(e) \mid e \in CC_i\}$;
8: `Incorporar o cliente` e `na rota` RT_i : $RT_i \leftarrow RT_i \cup \{e\}$;
9: `Atualizar o conjunto` CC_i : $CC_i \leftarrow CC_i \setminus \{e\}$;
10: `Recalcular o custo incremental` $c(e)$ `para todo` $e \in CC_i$;
11: **end;**
12: **if** $f(RT_i) < f(S_i)$ **then**
13: $S_i \leftarrow RT_i$;
14: $f(S_i) \leftarrow f(RT_i)$;
15: **end;**
16: **end;**
17: **return** $S = \bigcup_{i=\{1,...,Nrotas\}} S_i$;
end.

Dada a solução $S = \bigcup_{i=\{1,...,Nrotas\}} S_i$, obtida através do algoritmo *GRC*, no algoritmo *LSC* define-se, para cada $i \in \{1,...,Nrotas\}$

$$H_i = \{\text{clientes presentes na rota } S_i\},$$

$$RT_i : \text{rota temporária.}$$

A criação da rota temporária RT_i, é feita, de forma a manter inalterada a rota S_i. Define-se também

$$c(e), e \in H_i$$

o custo incremental associado à inserção de um cliente na rota RT_i. Os valores dos custos incrementais calculam-se considerando as distâncias entre clientes e entre clientes e depósito. Na primeira iteração, os custos a considerar são a soma das distâncias de ir do depósito ao cliente e vice-versa. Constrói-se assim uma rota só com um cliente. Nas iterações seguintes, os clientes são inseridos na rota RT_i, usando a heurística de inserção de menor custo.

Exemplo 3.2.1. *Considere-se a rota parcial {D C2 C1 D}. Supondo que se vai inserir o cliente C3. Os valores dos custos incrementais são os custos de inserir o cliente C3 entre o depósito (D) e o cliente C2, entre o cliente C2 e o cliente C1 e entre o cliente C1 e o depósito (D).*

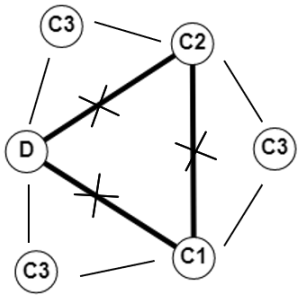

Figura 3.1: Inserção do cliente $C3$ na rota parcial *{D C2 C1 D}*

Seja $d_{i,j}$ a distância entre i e j, $i, j \in \{D, C2, C1, C3\}$, os valores dos custos, associados à inserção do cliente C3 na rota parcial, são os seguintes.

Inserção de C3 entre D e C2: $a = d_{D,C3} + d_{C3,C2} - d_{D,C2}$
Inserção de C3 entre C2 e C1: $b = d_{C2,C3} + d_{C3,C1} - d_{C2,C1}$
Inserção de C3 entre C1 e D: $c = d_{C1,C3} + d_{C3,D} - d_{C1,D}$
Deste modo o custo incremental associado à inserção do cliente C3 na rota parcial é o seguinte.

$$c(3) = min\{a, b, c\}.$$

A cada iteração c^{min} é o valor mínimo associado a $c(e)$, $e \in H_i$. Adiciona-se o cliente e na rota RT_i, na posição associada ao valor c^{min}. Além disso, atualiza-se o conjunto H_i e recalcula-se o valor de $c(e)$, $e \in H_i$. Este processo é repetido enquanto existirem clientes para inserir.

Seja $f(RT_i)$ e $f(S_i)$, os custos associados às rotas RT_i e S_i, respetivamente. Para cada $i \in \{1, ..Nrotas\}$, compara-se os valores de $f(RT_i)$ e $f(S_i)$. Caso a rota RT_i seja melhor que a rota S_i, ou seja, se o custo associado à rota RT_i for menor que o custo associado à rota S_i, atribui-se RT_i a S_i. Caso contrário, não se altera S_i. A solução final S, resulta da união de todos os $S_i, i \in \{1, ..., Nrotas\}$.

3.2.2.2 LocalSearchMercados

O algoritmo *LocalSearchMercados (LSM)* apresenta-se no Algoritmo 6. Este algoritmo tenta melhorar a solução R, obtida através do algoritmo *APUN*, tal que, $R = \bigcup_{i=\{1,...,Nrotas\}} R_i$. Este melhoramento, semelhante ao apresentado na secção 3.2.2.1, vai tentar reinserir os mercados com recurso à heurística de inserção de menor custo.

Dada as soluções S, R, T, obtidas através dos algoritmos *GRC*, *LSC* e *APUN*, no algoritmo *LSM*, define-se para cada $i \in \{1, ..Nrotas\}$

J_i = {mercados presentes na rota T_i },

RTT_i : rota temporária.

A criação da rota temporária RTT_i, é feita, de forma a manter inalterada a rota R_i. Define-se também

PCC_i = primeiro cliente na rota T_i

Algorithm 6 LocalSearchMercados (LSM)

1: **procedure** LOCALSEARCHMERCADOS(S,R,T)
2: **for** `i = 1,...,Nrotas` **do**
3: `Inicializar o conjunto dos mercados da rota` T_i: $MM_i \leftarrow J_i$;
4: $RTT_i \leftarrow \emptyset$;
5: `Determinar o primeiro cliente na rota` T_i: PCC_i;
6: `Determinar o custo incremental` $c(m)$ `para todo` $m \in MM_I$;
7: **while** $MM_i \neq \emptyset$ **do**
8: $c^{min} \leftarrow min\{c(m) \mid m \in MM_i\}$;
9: `Incorporar o mercado` m `na rota` RTT_i : $RTT_i \leftarrow RTT_i \cup \{m\}$;
10: `Atualizar o conjunto` MM_i : $MM_i \leftarrow MM_i \setminus \{m\}$;
11: `Recalcular o custo incremental` $c(m)$ `para todo` $m \in MM_i$;
12: **end;**
13: **if** $f(RTT_i) < f(R_i)$ **then**
14: $R_i \leftarrow RTT_i$;
15: $f(R_i) \leftarrow f(RTT_i)$;
16: $T_i = R_i \cup S_i$;
17: `Atualizar o custo da rota` T_i;
18: **end;**
19: **end;**
20: **return** $T = \bigcup_{w=\{1,...,Nrotas\}} T_w$;
end.

e

$$c(m), m \in J_i$$

o custo incremental associado à inserção do mercado m na rota temporária RTT_i. Neste caso, os valores dos custos incrementais são as distâncias entre mercados, entre mercados e depósito e entre mercados e o primeiro cliente presente na rota T_i, ou seja, o cliente PCC_i. Na primeira iteração, os custos a analisar são a soma das distâncias de ir do depósito ao mercado e deste ao cliente PCC_i. Constrói-se assim uma rota apenas com um mercado. Tal como no algoritmo *LSC*, a inserção de novos mercados, na rota RTT_i, é feita usando a heurística de inserção de menor custo.

Seja c^{min}, o valor mínimo associado a $c(m)$, $m \in J_i$. Insere-se o mercado m na rota RTT_i, na posição associada ao valor c^{min}. Além disso, atualiza-se o conjunto J_i e recalcula-se o valor de $c(m)$, $m \in J_i$. Este processo é repetido enquanto existirem mercados a adicionar, ou seja, enquanto $J_i \neq \emptyset$.

Seja $f(RTT_i)$ e $f(R_i)$, os custos associados às rotas RTT_i e R_i, respetivamente. Caso a rota RTT_i seja melhor que a rota R_i, atribui-se RTT_i a R_i. Além disso, atualiza-se a rota T_i e o respetivo custo. Este processo é repetido para todo o $i \in \{1,...,Nrotas\}$. No fim é obtido a solução T, que resulta da união de $T_w, w \in \{1,...,Nrotas\}$.

3.2.2.3 LocalSearchClientesMercados

O algoritmo *LocalSearchClientesMercados (LSCM)* apresenta-se no Algoritmo 7. Este algoritmo permite melhorar a solução T, obtida através do algoritmo *LSM*, tal que, $T = \bigcup_{i=\{1,...,Nrotas\}} T_i$. Este melhoramento vai tentar inserir mercados entre clientes.

Dada a solução $T = \bigcup_{i\in\{1,...,Nrotas\}} T_i$, no algoritmo *LSCM*, define-se para cada $i \in \{1,...,Nrotas\}$

$$L_i = \{\text{mercados presentes na rota } T_i\ \},$$

Algorithm 7 LocalSearchClientesMercados(LSCM)

```
1: procedure LocalSearchClientesMercados(T)
2:     for i = 1,...,Nrotas do
3:         Inicializar o conjunto de mercados da rota T_i: MMM_i ← L_i;
4:         Inicializar o conjunto de clientes da rota T_i: CCC_i ← V_i;
5:         RRT_i ← T_i;
6:         for j ∈ MMM_i do
7:             Determinar para o mercado  j  a ordem, na rota, do primeiro cliente
   servido pelo mercado j: PC_j;
8:             if PC_j ≠ 1 then
9:                 Determinar custo incremental c(r)_j para todo r ∈ {2,...,PC_j};
10:                c^min ← min{c(r)_j | r ∈ {2,...,PC_j}};
11:                Atualizar a rota RRT_i;
12:                if f(RRT_i) < f(T_i) then
13:                    T_i ← RRT_i;
14:                    f(T_i) ← f(RRT_i);
15:                else
16:                    RRT_i ← T_i;
17:                    f(RRT_i) ← f(T_i);
18:                end;
19:            end;
20:        end;
21:    end;
22:    return T= ⋃_{i={1,...,Nrotas}} T_i;
   end.
```

$$V_i = \{\text{clientes presentes na rota } T_i\ \}$$

e

$$RRT_i = T_i$$

Para cada mercado $j \in L_i$, associa-se o número PC_j, que é a ordem na rota T_i do primeiro cliente servido pelo mercado j.

Exemplo 3.2.2. *Considere-se a rota {D M2 M1 M3 C1 C3 C2 D}. Supondo que o mercado M2 tem como primeiro cliente, o cliente C2, tem-se que,* $PC_2 = 3$. *Vai-se tentar colocar o mercado M2 antes dos clientes C3 e C2.*

O processo descrito de seguida, apenas é efetuado caso a ordem do primeiro cliente servido pelo mercado j seja diferente de 1 ($PC_j \neq 1$), ou seja, caso o mercado j não sirva o primeiro cliente presente na rota em questão. Define-se, para cada mercado $j \in L_i$

$$c(r)_j, r \in \{2,...,PC_j\}$$

o custo incremental associado à inserção do mercado j antes do cliente $V(r)_i, r \in \{2,...,PC_j\}$. Neste caso, os valores dos custos incrementais calculam-se considerando as distâncias entre mercados e depósito, entre mercados, entre clientes e mercados e entre clientes. Os valores dos custo incrementais vão ser iguais à soma das distâncias de retirar o mercado j, da sua posição inicial, da distância entre os clientes de ordem r e $r-1$, o simétrico da distância de unir a rota, na posição onde foi retirado o mercado j, e o simétrico das distâncias de adicionar o mercado j antes do cliente $V(r)_i, r \in \{2,...,PC_j\}$

Exemplo 3.2.3. *Recorrendo ao exemplo 3.2.2. Supondo que vai-se tentar inserir o mercado M2 antes do cliente C3. Seja $j = 2$, $r = 2$ e $d_{i,k}$ a distância entre i e k, $i,k \in \{D, M1, M2, C1, C3\}$. O valor do custo*

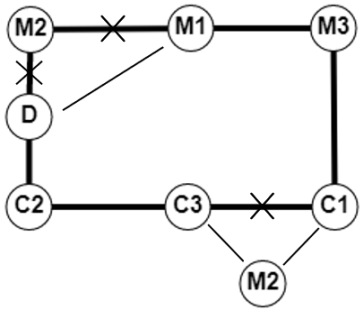

Figura 3.2: Inserção do mercado $M2$ antes do cliente $C3$

incremental, associado à inserção do mercado M2 antes do cliente C3, é o seguinte

$$c(2)_2 = (d_{D,M2} + d_{M2,M1}) + d_{C1,C3} - d_{D,M1} - (d_{C1,M2} + d_{M2,C3}).$$

Caso o valor do custo incremental seja negativo, significa que não existe poupança. Nestes casos os respetivos valores devem ser eliminados. Deste modo, os valores dos custo incrementais são sempre positivos.

No caso de existirem mercados antes do cliente $V(r)_i$, devem-se averiguar os custos de inserir o mercado j, antes do cliente $V(r)_i$ e dos mercados presentes, usando a heurística de inserção de menor custo. De seguida é necessário determinar para os custos associados à inserção do mercado j antes do cliente $V(r)_i$ e dos mercados presentes, o custo mínimo. O custo mínimo vai ser o custo incremental associado à inserção do mercado j antes do cliente $V(r)_i$.

Exemplo 3.2.4. *Recorrendo ao exemplo 3.2.3. Supondo que o mercado M2 é inserido antes do cliente C3 e que o mercado M3, tem como primeiro cliente, o cliente C3, ou seja, $PC_3 = 2$. Vai-se tentar colocar o mercado M3 antes do cliente C3. Neste caso, antes do cliente C3 está o mercado M2. Sendo assim o*

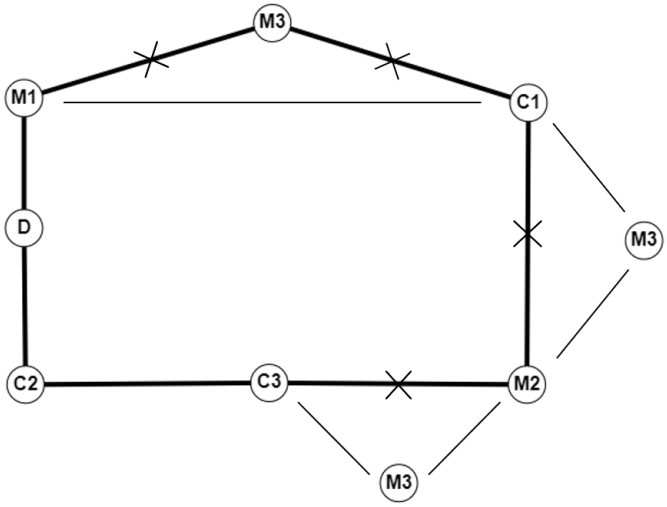

Figura 3.3: Inserção do mercado $M3$ antes do cliente $C3$

valor do custo incremental associado à inserção do mercado M3 antes do cliente C3 vai ser o seguinte.

$$c(2)_3 = d_{M1,M3} + d_{M3,C1} + min\{d_{C1,M2} - d_{C1,M3} - d_{M3,M2}, d_{M2,C3} - d_{M2,M3} - d_{M3,C3}\} - d_{M1,C1}.$$

Seja c^{min}, o valor mínimo associado a $c(r)_j, r \in \{2,...,PS_j\}$, retira-se na rota RRT_i o mercado j da sua posição inicial e recoloca-se o mercado j na posição associada ao valor c^{min}.

Seja os custos associados às rotas RRT_i e T_i, $f(RRT_i)$ e $f(T_i)$, respetivamente. Para cada mercado $j \in L_i$, compara-se os valores de $f(RRT_i)$ e $f(T_i)$. Caso o custo associado à rota RRT_i seja inferior ao custo associado à rota T_i, atribui-se RRT_i a T_i. Caso contrário, atribui-se a rota T_i à rota RRT_i. Este processo é repetido para todas as rotas, ou seja, para todo o $i \in \{1,...,Nrotas\}$.

Capítulo 4

Apresentação e análise dos resultados

Neste capítulo apresentam-se, considerando dados gerados aleatoriamente, os resultados computacionais obtidos utilizando a heurística *GRASP* desenvolvida neste trabalho de projeto para o problema de determinação de rotas de recolha e distribuição de produtos agrícolas. A heurística *GRASP* foi implementada em linguagem de programação MATLAB. Faz-se também uma análise dos resultados computacionais obtidos. Os resultados computacionais apresentados, foram obtidos usando um computador portátil com processador Intel(R) Core(TM) i3-5005U CPU @ 2.00GHZ 2.00 GHz e com 4.00 GB RAM instalada, e um sistema operativo Windows 10 Home.

4.1 Descrição dos dados

Nesta secção descreve-se cada conjunto de dados gerados aleatoriamente respeitando o princípio dos circuitos de proximidade. Nos circuitos de proximidade o número de *food miles* deve ser mínimo, deste modo, o parâmetro área deve ser pequeno, ou seja, a área onde se encontram o depósito, os clientes e os mercados deve ser pequena. O princípio dos circuitos curtos, ou seja, que o número de intermediários deve ser um ou zero, como foi explicado no capítulo 1, está incorporado no tipo de rotas construídas.

Na geração aleatória dos dados, consideram-se diferentes valores para vários parâmetros, de modo a obter diferentes cenários com o objetivo de simular a realidade. Um dos parâmetros importantes é a probabilidade de um cliente/mercado ter procura/oferta de um produto. Os dois casos considerados apresentam-se na tabela 4.1.

Tabela 4.1: Probabilidade de um cliente/mercado ter procura/oferta de um produto

	$Caso_1$	$Caso_2$
Clientes	70%	40%
Mercados	70%	70%

O parâmetro área indica a área onde os mercados, clientes e depósito se encontram. Consideram-se duas áreas: 1000 (50x20) e 5400 (90x60). Além disso, consideram-se diferentes valores para o número de mercados e clientes. A tabela 4.2 indica os vários pares de números de clientes e mercados considerados.

A geração das matrizes das distâncias entre depósito, clientes e mercados, é realizada da forma que se descreve seguidamente. Primeiro, geram-se, nas áreas consideradas, as coordenadas do depósito, dos clientes e dos mercados. De seguida, constrói-se uma matriz de distâncias euclidianas com densidade 80%, ou seja, a probabilidade de haver um caminho direto entre quaisquer dois elementos, depósito,

Tabela 4.2: Números de clientes e mercados

	$Caso_1$	$Caso_2$	$Caso_3$
Clientes	5	10	15
Mercados	10	20	18

cliente e mercado, é de 80%. A matriz das distâncias, entre quaisquer par de elementos, é a matriz com os comprimentos dos caminhos mais curtos, obtida usando o algoritmo de *Floyd* [7]. Todas as distâncias verificam, deste modo, a desigualdade triangular.

Os parâmetros número de produtos e intervalo da quantidade de procura/oferta dos clientes/mercados para um produto vão ser iguais para todos os conjuntos de dados. A tabela 4.3 apresenta os valores para os parâmetros acima mencionados.

Tabela 4.3: Parâmetros: nº de produtos e int. de procura/oferta dos clientes/mercados

	Valor
Número de produtos	4
Procura	[8,40]
Oferta	[2,60]

Os resultados computacionais obtidos dizem respeito às várias combinações dos valores dos parâmetros referidos. Na tabela 4.4 apresentam-se os quatro tipos de dados gerados.

Tabela 4.4: Tipos de dados

	Prob prod.	Área
$Tipo_1$	(70,70)	1000
$Tipo_2$	(70,70)	5400
$Tipo_3$	(40,70)	1000
$Tipo_4$	(40,70)	5400

Quanto à capacidade do veículo, consideram-se duas capacidades: 150 e 250. Estas capacidades foram escolhidas tendo em conta os dados gerados. A capacidade 150 corresponde, aproximadamente, ao dobro da maior procura por parte dos clientes, do conjunto de dados do $Tipo_1$ e $Tipo_2$. Considerou-se também a capacidade de 250, para permitir a criação de rotas com um maior número de clientes.

Para cada tipo de dados, consideram-se os três pares de números de clientes e mercados e as duas capacidades do veículo. As figuras 4.1 e 4.2 exemplificam a disposição espacial do depósito, dos clientes e dos mercados, para exemplos gerados do $Tipo_1$ e $Tipo_2$, respetivamente. Em ambos os casos considera-se o par de número de clientes e mercados igual a (5,10).

4.2 Parâmetros utilizados na heurística *GRASP*: α *e Iteracoes.max*

Na heurística *GRASP* apenas dois parâmetros são definidos: o número de *Iteracoes.max* realizadas e o parâmetro α, que determina a qualidade dos elementos da lista de candidatos restrita. Nesta secção apresentam-se os valores para α (ver secção 4.2.1) e *Iteracoes.max* (ver secção 4.2.2) utilizados na heurística *GRASP* desenvolvida para o problema em estudo.

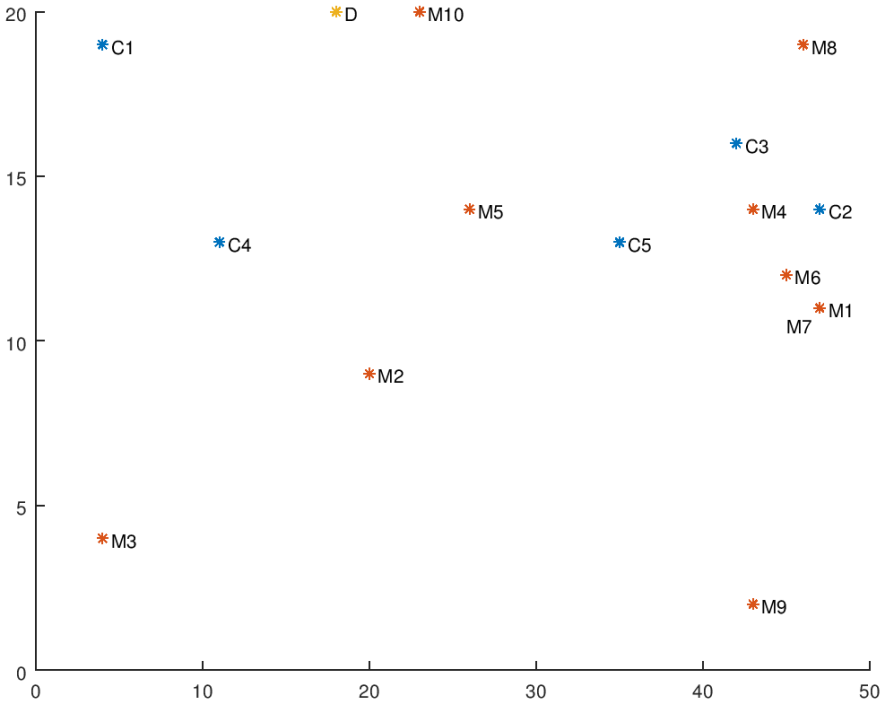

Figura 4.1: Exemplo - Tipo$_1$ (M: Mercados; C: clientes; D: depósito)

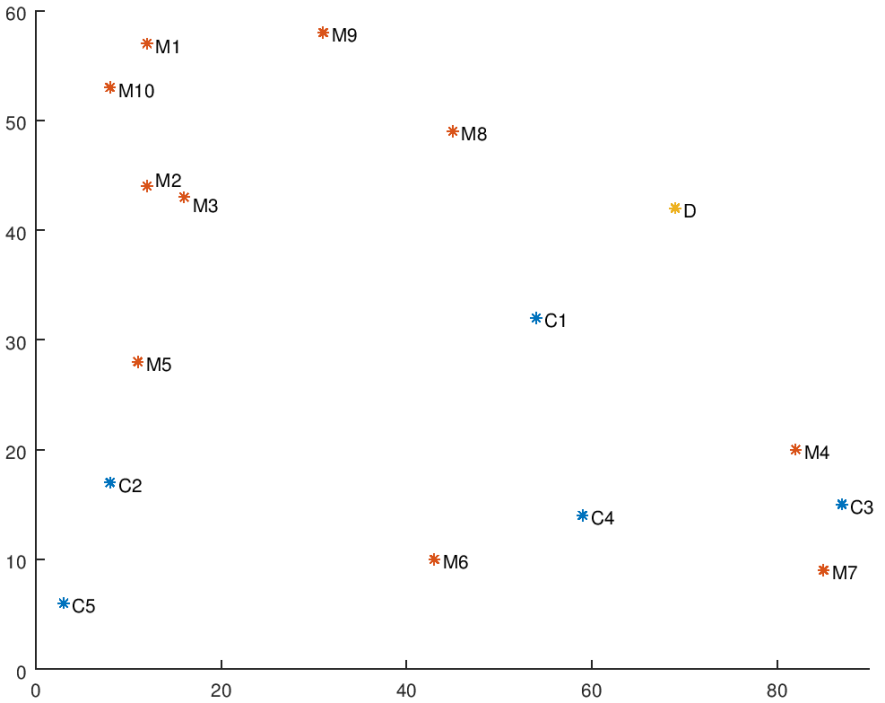

Figura 4.2: Exemplo - Tipo$_2$ (M: Mercados; C: clientes; D: depósito)

4.2.1 Parâmetro α

O parâmetro α varia no intervalo [0,1], correspondendo $\alpha = 0$ a um algoritmo puramente *greedy* e $\alpha = 1$ a um algoritmo completamente aleatório.

Dado um conjunto de dados, gerados aleatoriamente, considerando diferentes valores para os parâmetros indicados na secção 4.1, com o objetivo de obter um conjunto representativo dos dados, correu-se a heurística *GRASP*, para os valores de α iguais a: 0.1, 0.2, 0.5 e 0.7. Os valores testados para α, foram escolhidos com base em experiências realizadas anteriormente. O número de *Iteracoes.max* é escolhido consoante o número de clientes e mercados e com base em experiências realizadas anteriormente. A tabela 4.5 apresenta o número de *Iteracoes.max* realizadas, para cada número de clientes e mercados, no teste do parâmetro α. Na tabela 4.6 apresenta-se, para cada valor de α, a percentagem de casos em

Tabela 4.5: Valor de *Iteracoes.max* para o teste do parâmetro α

(clientes,mercados)	*Iteracoes.max*
(5,10)	250
(10,20)	500
(15,18)	600

que se obteve os melhores valores, ou seja, os menores custos. Os resultados completos, obtidos para o teste realizado para determinar o melhor valor para α, apresentam-se no anexo A na tabela A.1.

Tabela 4.6: Resultados do teste realizado para o valor de α

alpha	Percentagem (%)
0.1	10.42
0.2	54.17
0.5	20.83
0.7	20.83

Na tabela 4.6, observa-se que em 54.17% dos casos o valor de α igual a 0.2 apresenta melhores soluções, ou seja, soluções cujo custo associado é menor. Deste modo, define-se o valor para α de 0.2, conforme também é indicado por Mauricio G. C. Resende e Celso C. Ribeiro em [16].

4.2.2 Parâmetro *Iteracoes.max*

O número de iterações realizadas na heurística *GRASP*, ou seja, o valor para *Iteracoes.max*, depende dos vários parâmetros definidos na secção 4.1. Neste projeto, define-se para cada par de clientes e mercados o número de *Iteracoes.max*, tendo em conta também os outros parâmetros.

Dado um conjunto de dados, gerados aleatoriamente, com certos valores para os parâmetros indicados na tabela 4.4, correu-se a heurística *GRASP2* para os três pares de números de clientes e mercados, ou seja, para (5,10), (10,20) e (15,18), e para as duas capacidades, 150 e 250. A justificação do uso da heurística *GRASP2*, em vez da heurística *GRASP1*, é feita na secção 4.3.

O teste realizado para determinar o número de *Iteracoesm.max*, para cada par de número de clientes e mercados, foi o seguinte: começa-se, para um conjunto de dados, por correr a heurística *GRASP2* com um número inicial de iterações e vai-se incrementando o número de iterações, em cada execução da heurística, em 20 iterações. O critério de paragem é o seguinte: dado um *x* número de *Iteracoes.max*,

caso se observe, num intervalo de $[x, x+100]$ iterações, que a melhor solução obtida nesse intervalo tem um custo associado superior ou apenas uma diferença de 5 unidades, em relação à melhor solução obtida através de x iterações, então fica estabelecido o número para $Iteracoes.max$ de x, para aquele conjunto de dados. O valor de $Iteracoes.max$, obtido para cada par de número de clientes e mercados, corresponde ao máximo dos valores de $Iteracoes.max$ obtidos para cada conjunto de dados. Inicia-se o teste para o par (5,10) com o valor para $Iteracoes.max$ de 10. Enquanto que, inicia-se o teste para os pares (10,20) e (15,18) com o valor para $Iteracoes.max$ de 186, aproximadamente a média das $Iteracoes.max$ do par (5,10). Na tabela 4.7, estão presentes os valores para $Iteracoes.max$ consoante o par de números de clientes e mercados. Os resultados completos, obtidos para o teste realizado para determinar o valor para $Iteracoesmax$, apresentam-se no anexo A nas tabelas A.2, A.3 e A.4.

Tabela 4.7: Resultados do teste realizado para o número de $Iteracoes.max$

(clientes,mercados)	$Iteracoes.max$
(5,10)	390
(10,20)	526
(15,18)	666

4.3 Comparação entre *GRASP1 e GRASP2*

Na meta-heurística *GRASP*, depois de obter rotas só com clientes, vai-se adicionar os pontos de recolha considerando dois algoritmos: *AddPickUpNodes1* e *AddPickUpNodes2*. Designa-se por *GRASP1* a heurística contendo o algoritmo *AddPickUpNodes1* e por *GRASP2* a heurística contendo o algoritmo *AddPickUpNodes2*. Para vários conjuntos de dados, gerados aleatoriamente, e considerando vários valores para os parâmetros definido na secção 4.1, vai-se verificar qual das duas versões do *GRASP* determina as melhores soluções, ou seja, qual das versões determina soluções cujo custo associado é menor. O valor de α usado foi de 0.2, igual para o algoritmo *GRC* e $APUN2$, e o número de $Iteracoes.max$, como indicado para o teste do parâmetro α, foi escolhido consoante o número de clientes e mercados e a partir de experiências realizadas anteriormente. Na tabela 4.8 apresentam-se os resultados obtidos. Os resultados completos, obtidos para o teste realizado para comparar as duas versões da heurística, apresentam-se no anexo A na tabela A.5.

Tabela 4.8: Resultado da comparação entre *GRASP1* e *GRASP2*

	Percentagem (%)
$GRASP1$	17.39
$GRASP2$	82.61

Observa-se através da tabela 4.8, que em 82.61% dos casos a heurística *GRASP2*, ou seja, a heurística em que os mercados são escolhidos com recurso ao custo incremental, obtém resultados melhores que a heurística *GRASP1*, onde os mercados são escolhidos de forma completamente aleatória. Deste modo, os resultados computacionais dizem respeito à heurística *GRASP2*.

4.4 Resultados Computacionais

Nesta secção apresenta-se um resumo dos resultados computacionais obtidos usando a heurística *GRASP2*, desenvolvida neste trabalho de projeto para o problema de determinação de rotas de recolha e distribuição de produtos agrícolas. Também se apresenta, para um exemplo, as rotas obtidas usando a meta-heurística *GRASP2*, considerando os valores de α e $Iteracoes.max$ já definidos.

4.4.1 Exemplo

Os conjuntos de dados informam sobre a procura/oferta de um cliente/mercado para um produto, as distâncias entre depósito e clientes, entre clientes, entre depósito e mercados, entre mercados e entre mercados e clientes. De seguida, apresenta-se os dados gerados para um exemplo do Tipo_1, com cinco clientes e dez mercados, quatro tipos de produto e capacidade do veículo igual a 250 (ver anexo A, tabela A.7, exemplo 2).

$$\text{Procura} = \begin{matrix} \\ \text{Clientes} \end{matrix} \overset{\text{Produtos}}{\begin{bmatrix} 8 & 18 & 35 & 20 \\ 0 & 23 & 32 & 0 \\ 0 & 0 & 0 & 32 \\ 35 & 0 & 0 & 0 \\ 0 & 0 & 21 & 29 \end{bmatrix}} \qquad \text{Oferta} = \overset{\text{Produtos}}{\begin{bmatrix} 31 & 29 & 0 & 0 \\ 40 & 0 & 19 & 38 \\ 37 & 0 & 52 & 5 \\ 51 & 10 & 20 & 18 \\ 28 & 42 & 0 & 14 \\ 0 & 38 & 0 & 57 \\ 57 & 53 & 0 & 0 \\ 14 & 59 & 47 & 10 \\ 11 & 5 & 39 & 20 \\ 36 & 0 & 8 & 41 \end{bmatrix}} \begin{matrix} \\ \text{Mercados} \end{matrix}$$

$$\text{Distância} = \begin{matrix} \\ \text{Depósito (primeira linha) e Clientes} \end{matrix} \overset{\text{Depósito (primeira coluna) e Clientes}}{\begin{bmatrix} 0 & 37 & 43 & 21 & 17 & 47 \\ 37 & 0 & 13 & 16 & 32 & 11 \\ 43 & 13 & 0 & 29 & 45 & 6 \\ 21 & 16 & 29 & 0 & 17 & 26 \\ 17 & 32 & 45 & 17 & 0 & 42 \\ 47 & 11 & 6 & 26 & 42 & 0 \end{bmatrix}}$$

$$\text{Distância} = \overset{\text{Mercados}}{\begin{bmatrix} 43 & 22 & 20 & 10 & 36 & 25 & 40 & 29 & 49 & 51 \end{bmatrix}} \text{Depósito}$$

$$\text{Distância} = \begin{array}{c} \\ \text{Mercados} \\ \\ \\ \\ \\ \\ \\ \\ \\ \text{Mercados} \end{array} \overset{\text{Mercados}}{\begin{bmatrix} 0 & 40 & 28 & 36 & 13 & 18 & 20 & 16 & 14 & 10 \\ 40 & 0 & 25 & 12 & 35 & 23 & 39 & 26 & 48 & 48 \\ 28 & 25 & 0 & 30 & 25 & 13 & 32 & 12 & 37 & 36 \\ 36 & 12 & 30 & 0 & 27 & 18 & 31 & 24 & 41 & 42 \\ 13 & 35 & 25 & 27 & 0 & 26 & 7 & 29 & 14 & 16 \\ 18 & 23 & 13 & 18 & 26 & 0 & 19 & 9 & 26 & 26 \\ 20 & 39 & 32 & 31 & 7 & 19 & 0 & 23 & 12 & 16 \\ 16 & 26 & 12 & 24 & 29 & 9 & 23 & 0 & 27 & 25 \\ 14 & 48 & 37 & 41 & 14 & 26 & 12 & 27 & 0 & 7 \\ 10 & 48 & 36 & 42 & 16 & 26 & 16 & 25 & 7 & 0 \end{bmatrix}}$$

$$\text{Distância} = \begin{array}{c} \\ \text{Clientes} \\ \\ \\ \\ \\ \\ \\ \\ \\ \text{Mercados} \end{array} \overset{\text{Clientes}}{\begin{bmatrix} 16 & 10 & 31 & 42 & 5 \\ 43 & 47 & 28 & 11 & 41 \\ 44 & 38 & 31 & 14 & 33 \\ 47 & 46 & 31 & 23 & 41 \\ 29 & 23 & 44 & 39 & 18 \\ 34 & 28 & 44 & 27 & 23 \\ 36 & 30 & 51 & 46 & 25 \\ 32 & 26 & 43 & 26 & 21 \\ 30 & 24 & 45 & 51 & 19 \\ 26 & 20 & 41 & 50 & 15 \end{bmatrix}}$$

Aplica-se a estes dados a heurística *GRASP2*. As matrizes $S_{original}$, $R_{original}$ e $T_{original}$ representam as soluções S, R e T obtidas através dos algoritmos *GRC*, $APUN2$ e *LSM*, respetivamente. As matrizes $S_{melhorado}$, $R_{melhorado}$ e $T_{melhorado}$ consistem nas soluções S, R e T melhoradas através dos algoritmos *LSC*, *LSM* e *LSCM*. Cada linha, de cada matriz, representa uma rota.

Tem-se assim, no início, a matriz $S_{original}$ obtida através do algoritmo *GRC* constituída por duas rotas só com clientes. A primeira rota é constituída pelos clientes $C3$, $C4$, $C1$ e $C2$ e a segunda apenas pelo cliente $C5$. Aplicando o algoritmo de melhoramento (*LSC*) a estas rotas obtém-se a matriz $S_{melhorado}$.

$$S_{original} = \begin{bmatrix} C3 & C4 & C1 & C2 \\ C5 & & & \end{bmatrix} \qquad S_{melhorado} = \begin{bmatrix} C4 & C2 & C1 & C3 \\ C5 & & & \end{bmatrix}$$

As figuras 4.3 e 4.4 apresentam o depósito, os clientes e as rotas em $S_{original}$ e $S_{melhorado}$, respetivamente.

4. APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Ao aplicar o algoritmo $APUN2$ à solução $S_{melhorado}$ obtém-se a matriz $R_{original}$, apenas com mercados. Esta solução é constituída por duas rotas. A primeira contém os mercados $M2$, $M4$ e $M8$, que servem os clientes $C4$, $C2$, $C1$ e $C3$, enquanto que a outra contém os mercados $M3$ e $M6$, que servem o cliente $C5$. Aplicando o algoritmo LSM a estas rotas, obtém-se a matriz $R_{melhorado}$.

$$R_{original} = \begin{bmatrix} M2 & M4 & M8 \\ M3 & M6 & \end{bmatrix} \qquad R_{melhorado} = \begin{bmatrix} M4 & M8 & M2 \\ M3 & M6 & \end{bmatrix}$$

As figuras 4.5 e 4.6 apresentam o depósito, os mercados, os clientes $C4$ e $C5$ e as rotas em $R_{original}$ e $R_{melhorado}$, respetivamente.

A matriz $T_{original}$, resulta da união das matrizes $S_{melhorado}$ e $R_{melhorado}$. Aplicando o algoritmo de melhoramento ($LSCM$) às duas rotas obtém-se a matriz $T_{melhorado}$.

$$T_{original} = \begin{bmatrix} M4 & M8 & M2 & C4 & C2 & C1 & C3 \\ M3 & M6 & C5 & & & & \end{bmatrix} \qquad T_{melhorado} = \begin{bmatrix} M4 & M2 & C4 & M8 & C2 & C1 & C3 \\ M3 & M6 & C5 & & & & \end{bmatrix}$$

As figuras 4.7 e 4.8 apresentam o depósito, os clientes, os mercados e as rotas em $T_{original}$ e $T_{melhorado}$, respetivamente.

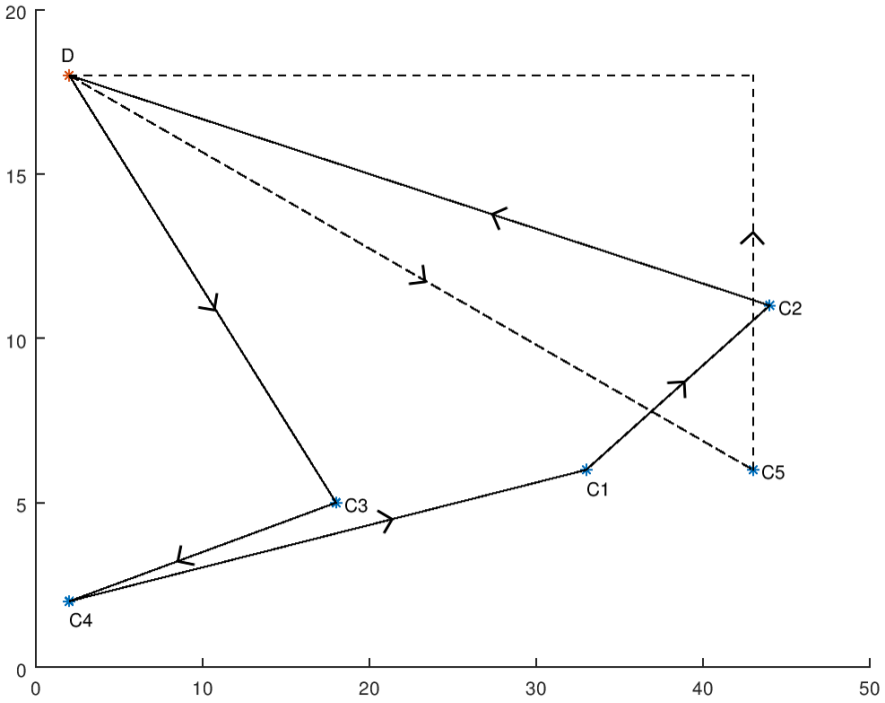

Figura 4.3: Solução $S_{original}$

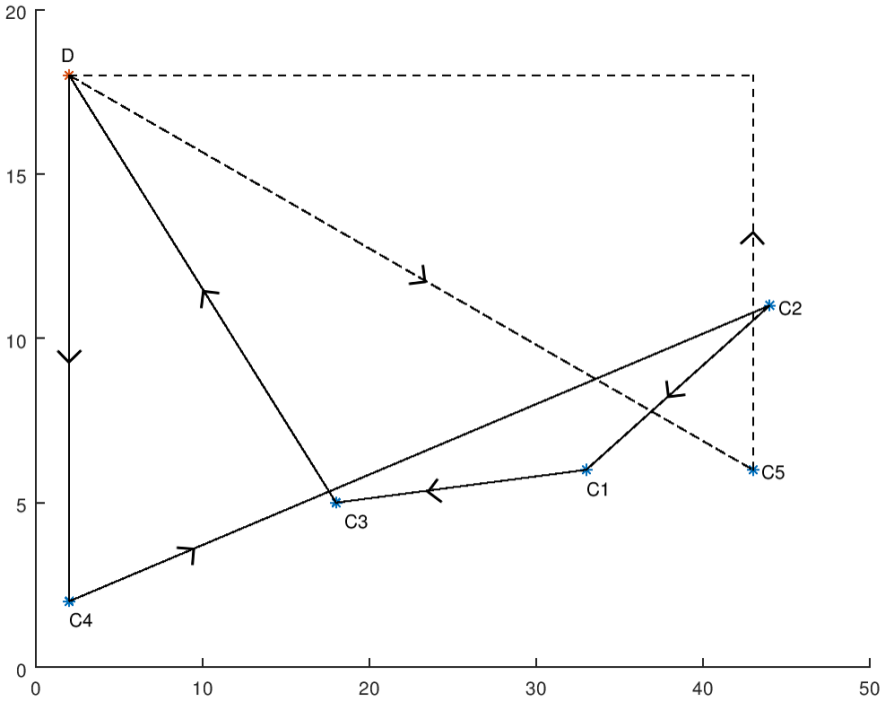

Figura 4.4: Solução $S_{melhorado}$

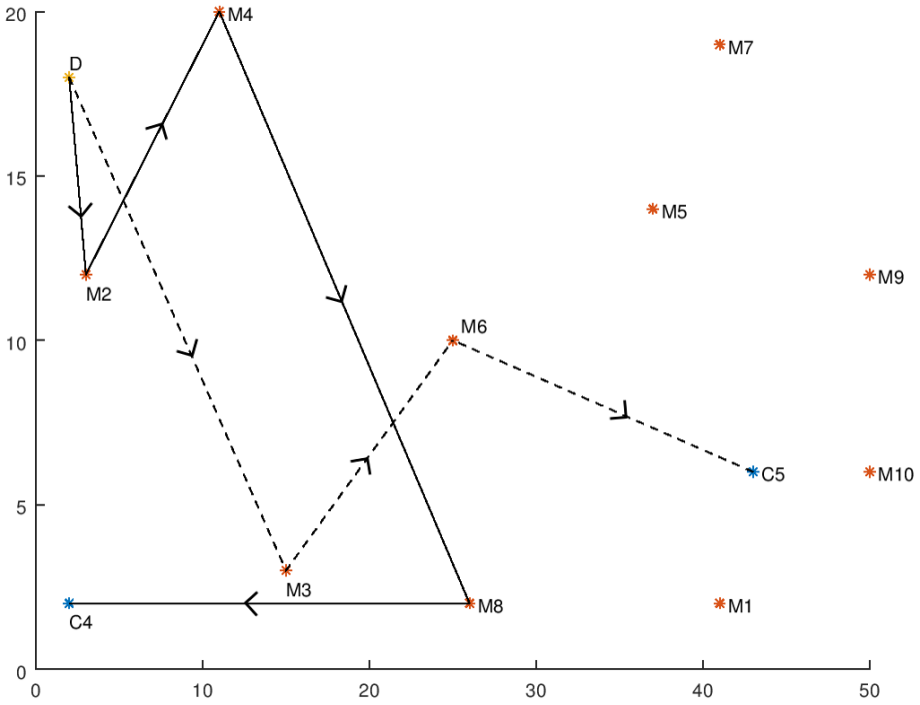

Figura 4.5: Solução $R_{original}$

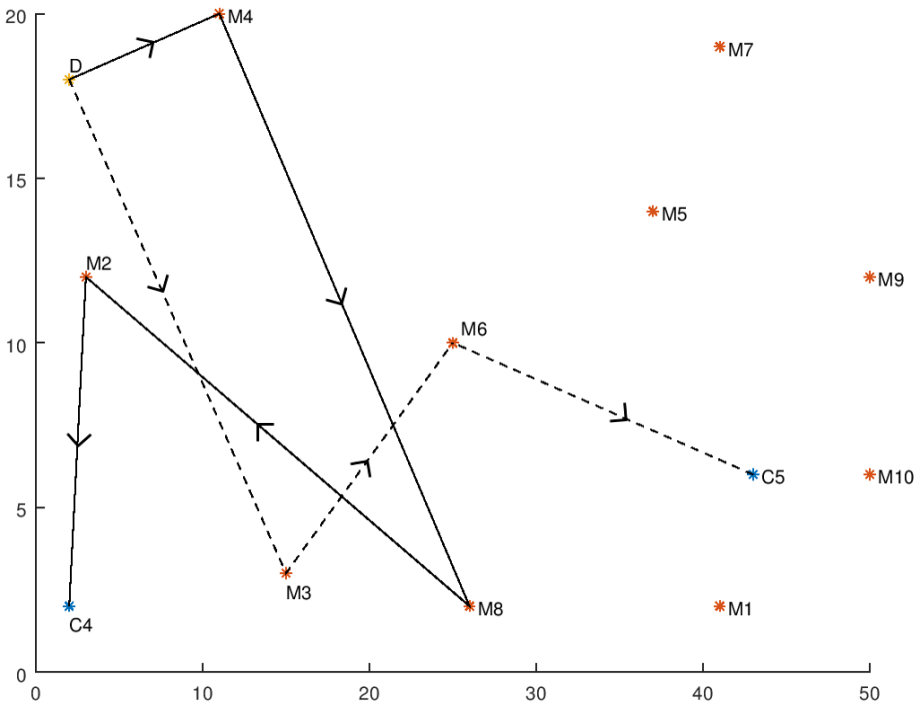

Figura 4.6: Solução $R_{melhorado}$

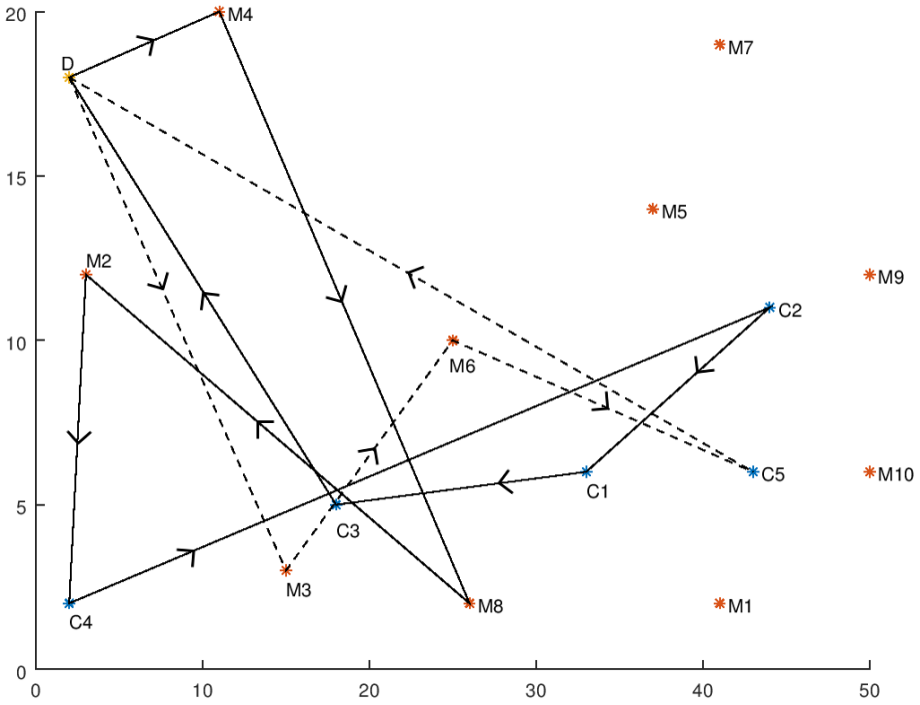

Figura 4.7: Solução $T_{original}$

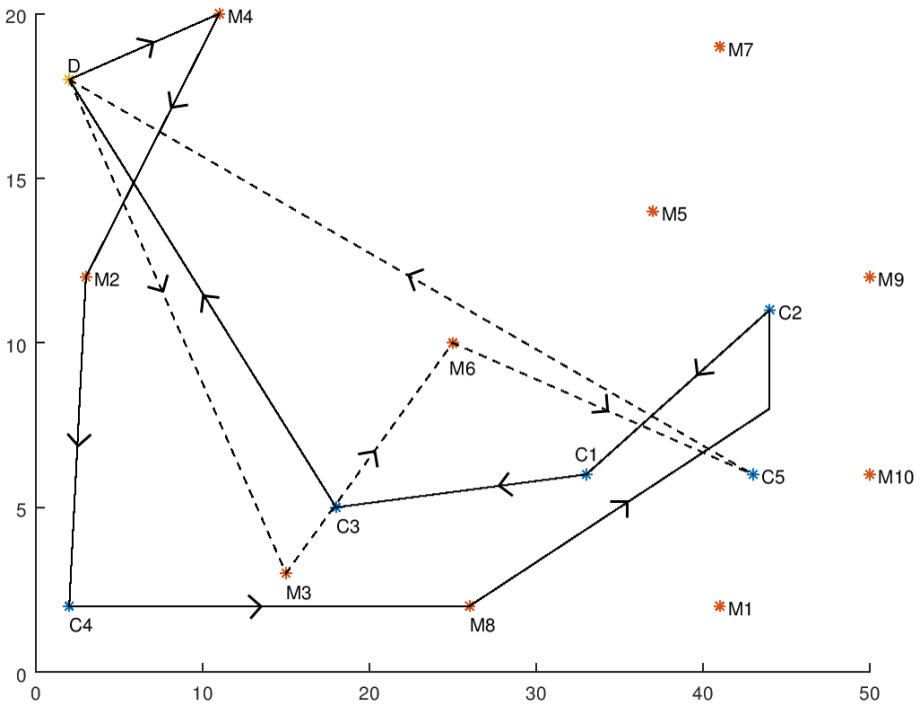

Figura 4.8: Solução $T_{melhorado}$

4.4.2 Apresentação dos Resultados

As soluções obtidas consistem em conjuntos de rotas que incluem pontos de recolha (mercados) e pontos de distribuição (clientes). Os resultados computacionais apresentados foram obtidos usando a heurística *GRASP2*, desenvolvida neste projeto, considerando quatro tipos de dados, três pares de números de clientes e mercados e duas capacidades do veículo. Para cada combinação de tipo de dados, pares e capacidades foram gerados cinco exemplos. O valor usado para α foi de 0.2, igual para os algoritmos *GRC* e $APUN2$. O número de $Iteracoes.max$ foi escolhido consoante o número de clientes e mercados. Deste modo para os pares (5,10), (10,20) e (15,18) foram usados os valores para $Iteracoes.max$ de 390, 526 e 666, respetivamente.

Nas tabelas 4.9, 4.10, 4.11 e 4.12 estão presentes os resultados computacionais obtidos usando a heurística *GRASP2*, para dados do $Tipo_1$, $Tipo_2$, $Tipo_3$ e $Tipo_4$, respetivamente. Nas quatro tabelas, a primeira coluna contém o par de número de clientes e mercados e a capacidade do veículo. Na segunda coluna está presente o valor máximo, mínimo e médio da percentagem de melhoria obtida ao implementar o algoritmo *LSC* partindo da solução *S*, apenas com clientes, obtida através do algoritmo *GRC*. A terceira coluna contém o valor máximo, mínimo e médio da percentagem de melhoria obtida ao implementar o algoritmo *LSM* partindo da solução R, apenas com mercados, obtida através do algoritmo $APUN2$. A quarta coluna apresenta o valor máximo, mínimo e médio da percentagem de melhoria ao implementar o algoritmo *LSCM* partindo da solução T, com mercados e clientes, obtida através do algoritmo *LSM*. Na quinta coluna apresenta-se o valor máximo, mínimo e médio do número de rotas presentes na solução T. Na última coluna está presente o valor máximo, mínimo e médio do tempo computacional (em segundos) para a heurística *GRASP2*, considerando todas as iterações. Os resultados computacionais completos estão presentes no anexo A nas tabelas A.7, A.8, A.9 e A.10.

Tabela 4.9: Resultados Computacionais -$Tipo_1$

Dados		Clientes(%)	Mercados(%)	Clientes/Mercados(%)	N-rotas	Tempo(s)
(5,10)-150	Máximo	10.34	20.20	1.99	4	14.00
	Mínimo	0.00	5.10	0.00	2	8.00
	Média	2.07	12.07	0.40	3	10.80
(5,10)-250	Máximo	9.94	25.87	11.52	3	14.06
	Mínimo	0.00	1.83	0.00	2	9.00
	Média	3.83	11.81	2.86	2	10.62
(10,20)-150	Máximo	3.42	18.11	1.82	7	49.03
	Mínimo	0.00	9.70	0.00	5	35.00
	Média	0.68	13.15	0.36	6	40.82
(10,20)-250	Máximo	7.62	30.96	0.00	4	46.00
	Mínimo	1.17	8.11	0.00	3	32.03
	Média	5.01	21.15	0.00	3	37.22
(15,18)-150	Máximo	0.00	20.55	0.00	10	88.00
	Mínimo	0.00	10.11	0.00	8	80.06
	Média	0.00	14.51	0.00	9	83.52
(15,18)-250	Máximo	5.91	22.76	1.58	6	86.00
	Mínimo	0.57	9.38	0.00	5	69.06
	Média	3.18	15.61	0.32	5	76.44

Tabela 4.10: Resultados Computacionais - $Tipo_2$

Dados		Clientes(%)	Mercados(%)	Clientes/Mercados(%)	N-rotas	Tempo(s)
(5,10)-150	Máximo	0.00	21.82	0.00	4	12.09
	Mínimo	0.00	3.88	0.00	3	9.00
	Média	0.00	11.23	0.00	3	10.64
(5,10)-250	Máximo	0.00	23.60	0.00	2	13.03
	Mínimo	0.00	0.00	0.00	2	9.00
	Média	0.00	10.03	0.00	2	10.42
(10,20)-150	Máximo	2.27	22.29	4.96	10	49.00
	Mínimo	0.00	1.90	0.00	5	34.03
	Média	0.45	10.18	0.99	7	41.42
(10,20)-250	Máximo	10.08	23.01	0.73	4	44.03
	Mínimo	0.00	3.41	0.00	3	33.03
	Média	3.42	14.76	0.15	3	39.23
(15,18)-150	Máximo	1.17	15.69	1.90	12	79.03
	Mínimo	0.00	3.62	0.00	7	62.03
	Média	0.23	10.92	0.38	9	74.83
(15,18)-250	Máximo	4.12	28.47	0.70	6	71.06
	Mínimo	1.63	0.78	0.00	4	52.06
	Média	3.06	18.69	0.16	5	65.43

Tabela 4.11: Resultados Computacionais - $Tipo_3$

Dados		Clientes(%)	Mercados(%)	Clientes/Mercados(%)	N-rotas	Tempo(s)
(5,10)-150	Máximo	11.19	5.69	0.00	2	10.00
	Mínimo	0.00	0.00	0.00	1	7.00
	Média	2.74	2.24	0.00	2	8.41
(5,10)-250	Máximo	22.31	15.56	0.00	1	9.03
	Mínimo	0.00	0.00	0.00	1	7.00
	Média	5.59	3.11	0.00	1	7.81
(10,20)-150	Máximo	10.94	25.53	5.12	4	30.03
	Mínimo	1.54	9.66	0.00	3	23.00
	Média	5.23	18.05	1.02	3	26.61
(10,20)-250	Máximo	14.65	29.90	0.00	3	30.00
	Mínimo	8.30	7.06	0.00	2	25.03
	Média	12.13	14.78	0.00	2	27.22
(15,18)-150	Máximo	13.17	12.17	2.48	6	57.09
	Mínimo	0.00	6.67	0.00	4	49.00
	Média	7.03	9.69	0.84	4	51.25
(15,18)-250	Máximo	9.90	23.26	4.78	3	54.00
	Mínimo	3.16	10.81	0.00	3	49.00
	Média	5.89	16.78	0.96	3	51.22

Tabela 4.12: Resultados Computacionais - $Tipo_4$

Dados		Clientes(%)	Mercados(%)	Clientes/Mercados(%)	N-rotas	Tempo(s)
(5,10)-150	Máximo	19.85	40.63	0.00	2	11.03
	Mínimo	0.00	0.00	0.00	1	7.03
	Média	8.88	20.21	0.00	2	8.63
(5,10)-250	Máximo	19.85	40.63	0.00	1	9.00
	Mínimo	3.41	9.32	0.00	1	8.03
	Média	11.20	26.75	0.00	1	8.24
(10,20)-150	Máximo	5.28	13.16	2.77	5	32.03
	Mínimo	0.00	3.00	0.00	3	21.03
	Média	2.65	7.74	0.68	4	27.02
(10,20)-250	Máximo	14.94	26.13	0.00	3	28.00
	Mínimo	1.89	8.22	0.00	2	20.00
	Média	6.19	14.18	0.00	3	25.03
(15,18)-150	Máximo	4.85	18.79	1.87	6	61.00
	Mínimo	0.39	0.00	0.00	4	41.00
	Média	3.13	10.04	0.43	5	50.21
(15,18)-250	Máximo	9.51	21.22	0.37	4	61.00
	Mínimo	1.92	5.09	0.00	2	44.00
	Média	5.78	11.33	0.07	3	50.00

4.5 Análise dos resultados

Nesta secção apresenta-se uma análise dos resultados computacionais obtidos através da heurística *GRASP2*, desenvolvida neste trabalho de projeto. Na tabela 4.13 apresenta-se, os resultados computacionais considerando três parâmetros: a capacidade do veículo (150, 250), o tipo de dados (1, 2, 3, 4) e o par de número de clientes e mercados ((5,10), (10,20), (15,18)). Para cada valor da capacidade dos veículos, faz-se a média das médias obtidas, fixando cada uma das capacidades e variando os outros dois parâmetros (tipo de dados e número de clientes e mercados). Para cada tipo de dados, faz-se a média das médias obtidas fixando este parâmetro e variando os outros dois parâmetros. Para cada par de número de clientes e mercados, faz-se a média das médias obtidas fixando este parâmetro e variando os outros dois parâmetros.

Na tabela 4.13, as primeira três linhas apresentam a percentagem média de melhoria obtida ao implementar os algoritmos *LSC*, *LSM* e *LSCM* às soluções S, apenas clientes, R, apenas mercados, e T, inclui mercados e clientes, respetivamente. A quarta linha apresenta o número médio de rotas presentes na solução final T. A última linha apresenta o tempo médio de execução total da heurística *GRASP2*, em segundos.

Na tabela 4.14, as três linhas apresentam, fixando um parâmetro e variando os outros dois, a percentagem de casos em que os algoritmo *LSC* (primeira linha), *LSM* (segunda linha) e *LSCM* (terceira linha), produziram uma diminuição no custo associado às solução S, R e T, respetivamente.

A tabela 4.15 indica, fixando um parâmetro e variando os outros dois, a percentagem de rotas que contêm apenas um ou dois clientes.

A heurística *GRASP2* desenvolvida para este projeto, determina usando o algoritmo *GreedyRandomizedConstruction*, a solução S, apenas com clientes, com recurso ao custo incremental. O algoritmo *LocalSearchClientes* tenta melhorar a solução S. Na tabela 4.13 observa-se que relativamente à capaci-

Tabela 4.13: Análises dos Resultados

	Capacidade		Tipo				Par		
	150	250	1	2	3	4	(5,10)	(10,20)	(15,18)
Clientes (%)	2.76	5.44	2.46	1.19	6.44	6.30	4.29	4.47	3.54
Mercados (%)	11.67	14.92	14.72	12.64	10.78	15.04	12.18	14.25	13.45
Clientes/Mercados (%)	0.43	0.38	0.66	0.28	0.47	0.20	0.41	0.40	0.40
N-rotas	5	3	5	5	3	3	2	4	5
Tempo (s)	36.18	34.07	43.24	40.33	28.75	28.19	9.45	33.07	62.86

Tabela 4.14: Casos em que ocorre melhoria

	Capacidade		Tipo				Par		
	150	250	1	2	3	4	(5,10)	(10,20)	(15,18)
Clientes (%)	46.67	83.33	53.33	33.33	86.67	90.00	52.50	70.00	75.00
Mercados (%)	93.33	91.67	100.00	96.67	80.00	93.33	80.00	100.00	97.50
Clientes/Mercados (%)	20.00	13.33	16.67	16.67	16.67	16.67	7.50	15.00	27.50

Tabela 4.15: Rotas com um ou dois clientes

	Capacidade		Tipo				Par		
	150	250	1	2	3	4	(5,10)	(10,20)	(15,18)
1 ou 2 (%)	74.13	29.24	72.73	73.97	24.05	34.83	62.03	61.01	52.97

dade do veículo pode-se concluir que a percentagem média de melhoria é superior quando a capacidade do veículo é igual a 250. Quando a capacidade do veículo é igual a 150, em 46.67% dos casos ocorre uma melhoria na solução *S*, enquanto que quando a capacidade do veículo é igual a 250, a solução *S* é melhorada em 83.33% dos casos. Isto acontece, pois quando as rotas são constituídas por apenas um ou dois clientes, a solução *S* nunca é melhorada. As razões para tal acontecer são porque o início e fim da rota são o depósito e porque as distâncias são simétricas. Nos casos em que a capacidade do veículo é igual a 150, 74.13% das rotas são constituídas por um ou dois clientes. Enquanto que nos casos em que a capacidade do veículo é igual a 250, apenas 29.24% das rotas são constituídas por um ou dois clientes.

O tipo de dados também influencia a percentagem de melhoria obtida através do algoritmo *LSC*. Ao comparar os dados do Tipo_1 com os do Tipo_3, e da mesma forma, os dados do Tipo_2 com os do Tipo_4, que variam apenas nas probabilidades associadas à procura nos clientes, observa-se que a percentagem média de melhoria é sempre superior nos dados do Tipo_3 e do Tipo_4. Isto significa que a percentagem média de melhoria é inferior quando a probabilidade de um cliente querer um produto é 70%. Nos casos em que os dados são do Tipo_1 e do Tipo_2 o melhoramento ocorre, em média, em 43.33% dos casos, enquanto que nos dados do Tipo_3 e do Tipo_4 a solução *S* é melhorada, em média, em 88.34% dos casos. Isto acontece, pois nos casos em que a probabilidade de um cliente querer um produto é 70%, a soma da procura de todos os clientes é superior. Quanto maior é a soma da procura de todos os clientes, menor é o número de clientes em cada rota e um maior número de rotas contém apenas um ou dois clientes. Nos dados do Tipo_1 e do Tipo_2, em média, 73.35% das rotas são constituídas por apenas um ou dois clientes, o que implica, que nessas rotas nunca ocorre melhoria. Enquanto que nos dados do Tipo_3 e do Tipo_4, em média, apenas 29.44% das rotas são constituídas por um ou dois clientes.

O número de clientes e mercados, não permite tirar grandes conclusões, uma vez que, a percentagem média de melhoria não varia mais que 0.93%, entre os vários pares.

O algoritmo *AddPickUpNodes2* adiciona os mercados que servem os clientes em cada rota em S, obtendo assim a solução R, apenas com mercados. Tenta-se melhorar a solução R, através do algoritmo $LocalSearchMercados$. Na tabela 4.13, observa-se que para qualquer um dos parâmetros, a percentagem média de melhoria, é muito semelhante, não permitindo assim tirar grandes conclusões. Em 92.50% dos casos, independente do parâmetro, ocorre melhoramento da solução R.

A solução T, resulta da união das rotas em S e R já melhoradas, ou seja, contém clientes e mercados. O algoritmo $LocalSearchClientesMercados$, tenta melhorar a solução T, inserindo mercados entre clientes. Tal como aconteceu com o algoritmo *LSM*, os resultados obtidos são muito semelhantes para os três parâmetros. Isto deve-se ao facto de que em apenas 20 dos 120 casos, ocorre melhoria da solução T.

O número de rotas, indica, o número total de rotas presentes na solução T. Na tabela 4.13, observa-se que quanto maior for a capacidade do veículo menor será o número de rotas. Isto acontece pois, quanto maior for a capacidade do veículo, maior será a capacidade em cada rota. Consequentemente, cada rota permite conter um maior número de clientes e o número de rota criadas é menor.

O tipo de dados, também interfere no número de rotas que são criadas. Ao comparar os dados do Tipo_1 com os do Tipo_3, e da mesma forma, os dados do Tipo_2 com os do Tipo_4, que variam apenas na probabilidade de um cliente querer um produto, observa-se que o número de rotas criadas, em média, é sempre superior nos dados do Tipo_1 e do Tipo_2. Isto acontece, pois quando a probabilidade de procura de um produto, por parte do cliente, é de 70%, em vez de 40%, a soma da procura de todos os clientes é maior. Isto implica que cada rota, independente da capacidade do veículo, contém um menor número de clientes. Consequentemente, ocorre a criação de um maior número de rotas na solução T.

O parâmetro, número de clientes e mercados, influencia o número de rotas criadas. O número de clientes é que influencia mais o número de rotas criadas. Quanto maior for o número de clientes maior é a soma da procura de todos os clientes. Deste modo, tal como ocorre com o parâmetro probabilidade de um cliente querer um produto, quando a soma da procura de todos os clientes é maior, implica que, cada rota, contém um menor número de clientes. Consequentemente, quanto maior for o número de clientes, a inserir na solução, maior será o número de rotas presentes na solução T final.

Finalmente, analisa-se o tempo computacional (em segundos) associado à heurística *GRASP2*. O número de rotas está diretamente relacionado com o tempo computacional. Quanto maior for o número de rotas criadas, maior é o tempo de execução. Deste modo, em cada parâmetro, o valor associado ao maior número de rotas, está também associado ao maior tempo de execução. No caso, do parâmetro, número de clientes e mercados, há que ter em consideração, que o número de $Iteracoes.max$ realizadas também interfere com o tempo computacional. Quanto maior for o número de clientes, maior é o número de $Iteracoesm.max$ e consequentemente o tempo computacional.

Capítulo 5

Conclusão

O desenvolvimento de sistemas locais e regionais de produção de alimentos, envolve a determinação de rotas de recolha e distribuição de produtos agrícolas em circuitos curtos e de proximidade. Assim é necessário determinar rotas, para vários veículos, em que existem pontos de recolha de produtos alimentares (*pickup*) e pontos onde esses produtos são entregues (*delivery*). Os pontos de recolha são os agricultores e os mercados onde os agricultores vendem diretamente os seus produtos e os pontos onde os produtos são entregues são os clientes. Os clientes são por exemplo: vendas *online*, restaurantes, escolas, mercearias, lojas de organização de produtores, cabazes, entre outros. O método utilizado neste trabalho, para obter soluções admissíveis para este problema, foi a heurística *GRASP*. A heurística *GRASP* foi implementada em linguagem de programação MATLAB e testada considerando dados gerados aleatoriamente.

Na geração aleatória dos dados, consideraram-se diferentes valores para vários parâmetros, de forma a obter diferentes cenários com o objetivo de simular a realidade. São definidos 4 tipos de dados, que variam na probabilidade de procura/oferta de um produto, por parte de um cliente/mercado e na área onde estão inseridos o depósito, os clientes e os mercados. Para todos os conjuntos de dados, o número de produtos e o intervalo da quantidade de procura/oferta dos clientes/mercados é igual. Para cada tipo de dados consideraram-se três pares de números de clientes e mercados e duas capacidades para o veículo.

As rotas obtidas através da heurística *GRASP*, desenvolvida neste trabalho de projeto, e a geração aleatória dos dados respeitam os princípios dos circuitos curtos e de proximidade. O número de intermediários deve ser um ou zero, deste modo, as rotas construídas consistem em rotas de recolha, em agricultores ou mercados onde os agricultores vendem diretamente os seus produtos, e imediata distribuição nos clientes. A distância entre o local de produção e venda deve ser mínima, deste modo, o objetivo é minimizar a distância total percorrida pelos veículos. Além disso, na geração dos dados são consideradas áreas pequenas.

A heurística *GRASP* é uma meta-heurística iterativa, em que cada iteração consiste em duas fases: construção e pesquisa local.

A heurística *GRASP*, desenvolvida para o problema apresentado neste projeto, tem como objetivo determinar rotas que permitem a recolha e posterior distribuição de produtos alimentares, de forma, a minimizar o custo total, ou seja, a distância percorrida pelos veículos. A heurística desenvolvida obtém sempre soluções admissíveis em cada iteração. Consideraram-se no problema em estudo várias hipóteses. A frota de veículos é homogénea. Os veículos, todos com a mesma capacidade, iniciam e terminam a rota no depósito. Em cada rota um mercado é visitado no máximo uma vez. A procura dos clientes deve ser satisfeita e a disponibilidade de cada produto, em cada mercado, não deve ser excedida. O tempo total da rota está de acordo com as restrições relativas ao horário dos motoristas dos veículos.

Na heurística *GRASP* desenvolvida neste trabalho de projeto, determina-se usando o algoritmo *Gre-*

edyRandomizedConstruction, uma solução inicial que consistem em rotas só com clientes escolhidos com recurso ao custo incremental. Recorrendo ao algoritmo *LocalSearchClientes* tenta-se melhorar esta solução reinserindo os clientes com recurso à heurística de inserção de menor custo.

O algoritmo *AddPickUpNodes* cria rotas só com os mercados que servem os clientes de cada rota de clientes. Considerou-se, neste procedimento, duas opções de escolha de mercados: escolha aleatória (*GRASP*1 - *AddPickUpNodes*1) e escolha com recurso ao custo incremental (*GRASP*2 - *AddPickUpNodes*2). Tenta-se melhorar esta solução usando o algoritmo *LocalSearchMercados*, que de modo semelhante ao *LSC*, tenta reinserir os mercados com recurso à heurística de inserção de menor custo. Constrói-se uma solução final que resulta da união das soluções com mercados e clientes. As rotas na solução final, têm presente, primeiro os mercados e só depois os clientes. Aplica-se o algoritmo *LocalSearchClientesMercados*, que tenta melhorar a solução final, inserindo mercados entre clientes.

Na implementação da heurística *GRASP* foi necessário definir dois parâmetros: *Iteracoes.max* e α. Para definir estes parâmetros foram realizados testes computacionais. O parâmetro α, que determina a qualidade dos elementos na lista de candidatos, varia no intervalo [0,1]. Correu-se ambas as heurísticas *GRASP*, para valores de α iguais: 0.1, 0.2, 0.5 e 0.7. Conclui-se que o valor de α a usar era de 0.2. Através deste teste também se conclui que a heurística *GRASP*2 era melhor que a heurística *GRASP*1. Deste modo, os resultados computacionais foram obtidos usando a heurística *GRASP*2. O número de *Iteracoes.max*, depende de vários parâmetros, contudo neste projeto definiu-se o número de *Iteracoes.max* para cada par de número de clientes e mercados. Testes foram efetuados para determinar o valor para *Iteracoes.max*. Conclui-se que para os pares (5,10), (10,20) e (15,20), o número adequado seria, 390, 526 e 666 iterações, respetivamente.

Os resultados computacionais dizem respeito à heurística *GRASP*2, considerando quatro tipos de dados, três pares de número de clientes e mercados e duas capacidades do veículo. Fez-se uma análise dos resultados computacionais, em relação, à percentagem média de melhoria obtida ao implementar os algoritmos *LSC*, *LSM* e *LSCM*, ao número médio de rotas presentes na solução final e ao tempo médio computacional da heurística *GRASP*2. Verificou-se que o algoritmo *LSC* tinha uma maior percentagem média de melhoria, nos casos em que a capacidade do veículo era superior e nos casos em que a soma da procura de todos os clientes era inferior. Tal acontece porque nos casos contrários, existe um maior número de rotas constituídas por apenas um ou dois clientes. Nestes casos, o melhoramento nos clientes nunca é efetuado.

No algoritmo de melhoramento dos mercados (*LSM*), independente do parâmetro, a percentagem média de melhoria era semelhante, não permitindo deste modo tirar grandes conclusões. Da mesma forma para o algoritmo *LSCM*, não foi possível tirar grandes conclusões, pois para cada parâmetro, a percentagem média de melhoria não variava mais que 1%. Isto deve-se ao facto de que apenas em 20 dos 120 casos ocorreu melhoria da solução final.

O número de rotas presentes na solução final é maior nos casos em que a capacidade do veículo é menor e nos casos em que a soma da procura de cada cliente é maior. Verifica-se então que o número de rotas e a percentagem média de melhoria do algoritmo *LSC* estão correlacionadas de forma negativa. Uma vez que, quanto menor é a capacidade do veículo e maior é a soma da procura de cada cliente, maior é o número de rotas, e consequentemente menor é o número de clientes presentes em cada rota. Deste modo, um maior número de rotas tem apenas um ou dois clientes, o que implica uma menor percentagem média de melhoria da solução com clientes.

O tempo computacional da heurística *GRASP*2 está correlacionado, de forma positiva, com o número de rotas criadas. Quanto maior é o número de rotas maior será o tempo computacional. Contudo, ter em conta também que quanto maior é o número de clientes e mercados, maior é o número de *Iteracoes.max*

e isto também interfere com o tempo computacional.

A heurística *GRASP* básica é um procedimento sem memória. Vários melhoramentos podem ser aplicados ao *GRASP* básico tais como o *path relinking* e o *reactive GRASP*. O *Path Relinking*, dada uma solução inicial, cria um caminho, na vizinhança desta solução, entre a solução inicial e soluções consideradas boas. O *Reactive GRASP* permite atualizar o valor de α a cada iteração.

A heurística *GRASP* desenvolvida neste trabalho de projeto para o problema de determinação de rotas com recolha e distribuição de produtos agrícolas também pode ser melhorada. Pode-se verificar se existem, na rota, mercados que são redundantes, isto é, a sua oferta pode ser substituída pela dos outros mercados presentes na mesma rota. Também se pode considerar a fusão de rotas. Existem dois casos: casos mais simples, em que não há mercados em comum entre ambas as rotas e basta unir as rotas e casos em que há mercados em comum em ambas as rotas. Nestes casos, uma vez que cada mercado só pode ser visitado no máximo uma vez em cada rota, a procura dos clientes, servidos pelo mercado em comum, pode ser satisfeita por outros mercados presentes nas rotas.

Bibliografia

[1] Alexandra Prado Coelho (Texto) e Vera Moutinho (Fotografia e vídeo). A distribuição alimentar perdida no seu labirinto. https://www.publico.pt/multimedia/interactivo/alimentacao-na-cidade#a-logica-dos-circuitos-curtos, 29 de abril de 2018. Acedido: 17 de março de 2019.

[2] Guillaume Fourdinier. Recreating the link between producers and consumers. https://medium.com/welcome-to-agricool/recreating-the-link-between-producers-and-consumers-7c97fdadb282, 20 de abril de 2017. Acedido: 10 de março de 2019.

[3] Alexandra Prado Coelho. As nossas cidades são capazes de se alimentar? https://www.publico.pt/2018/01/11/sociedade/noticia/as-nossas-cidades-sao-capazes-de-se-alimentar-1798786, 11 de janeiro de 2018. Acedido: 20 de março de 2019.

[4] Bruno M. Craveiro de Oliveira. Planning the distribution of agricultural products in a short distribution channel. Master's thesis, Universidade de Lisboa, Repositório de teses de mestrado da Universidade de Lisboa, 2013.

[5] SARE Abby Massey. Farm to table: Building local and regional food systems. https://www.sare.org/Learning-Center/Topic-Rooms/Farm-to-Table-Building-Local-and-Regional-Food-Systems, 2015. Acedido: 11 de novembro de 2018.

[6] EPRSLibrary. Local agriculture and short food supply chains. https://epthinktank.eu/2013/10/14/local-agriculture-and-short-food-supply-chains/, 14 de outubro de 2013. Acedido: 15 de novembro de 2018.

[7] H. A. Taha. *Operations Research: An Introduction*. Prentice Hall, 10 edition, 2017.

[8] M. W. P. Savelsberg and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, 1995.

[9] The vehicle routing problem. In P. Toth and D. Vigo, editors, *SIAM Monographs on Discrete Mathematics and Applications*, volume 9. 2002.

[10] I. K. Altinel T. Oncan and G. Laporte. A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers and Operations Research*, 36:637–654, 2009.

[11] Hanif D. Sherali Subhash C. Sarin and Ajay Bhootra. New tighter polynomial length formulations for the asymmetric travelling salesman problem with and without precedence constraints. *Operations Research Letters*, 33:62–70, 2005.

[12] F. Alvelos A. Rais and M.S. Carvalho. New mixed integer-programming model for the pickup-and-delivery problem with transshipment. *European Journal of Operational Research*, 235:530–539, 2014.

[13] Leonidas Pitsoulis and Mauricio G. C. Resende. Greedy randomized adaptive search procedures. In Pano M. Pardalos and Mauricio G.C. Resende, editors, *Handbook of Applied Optimization*, pages 168–183. Oxford University Press, 2002.

[14] Mauricio G. C. Resende and Celso C. Ribeiro. Greedy randomized adaptive search procedures. In Fred Glover and Gary A. Kochenberger, editors, *Handbook of Metaheuristics*, chapter 8, pages 219–249. Springer US, Boston, MA, 2003.

[15] Thomas A. Feo and Mauricio G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[16] Mauricio G. C. Resende and Celso C. Ribeiro. Grasp: Greedy randomized adaptive search procedures. In Edmund K. Burke and Graham Kendall, editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, chapter 11, pages 287–312. Springer US, Boston, MA, 2014.

[17] Marl. `http://www.marl.pt/omarl/`, 2017. Acedido: 9 de novembro de 2018.

[18] Developing local and regional food production systems. `https://frenchfoodintheus.org/1022`, 7 de maio de 2015. Acedido: 15 de março de 2019.

[19] Vítor Andrade. Sistemas alimentares locais e circuitos curtos agroalimentares. `https://www.jornaldenegocios.pt/opiniao/colunistas/economia-social/detalhe/sistemas-alimentares-locais-e-circuitos-curtos-agroalimentares/`, 9 de maio de 2018. Acedido: 20 de março de 2019.

[20] Emília Freire. Venda direta e cadeias curtas ganham relevância. `https://www.vidarural.pt/insights/venda-direta-e-cadeias-curtas-ganham-relevancia/`, 2015. Acedido: 05 de julho de 2019.

[21] Sin Ho and W Szeto. Grasp with path relinking for the selective pickup and delivery problem. *Expert Systems with Applications*, 51:14–25, 2016.

[22] Mauricio G. C. Resende. Greedy randomized adaptive search procedures (grasp). In C. Floudas and P.M. Pardalos, editors, *Encyclopedia of Optimization*, volume 2, pages 373–3382. Kluwer Academic Press, 2001.

Apêndice A

Resultados

Na tabela A.1, apresentam-se os resultados obtidos para o teste que permite determinar o melhor valor para o parâmetro α. Na primeira coluna estão presentes o tipo de dados, o número de clientes e mercados, a capacidade do veículo e o número de iterações realizadas, ou seja, o número de *Iteracoes.max*. As outras colunas indicam, para cada valor de α e para a cada heurística (*GRASP1* e *GRASP2*), o custo associado à solução obtida. A negrito está indicado, para cada heurística, o valor de α que permite determinar a solução associada ao menor custo. A linha Resultado apresenta o número de casos em que, o respetivo valor de α, determina a solução associada ao menor custo.

Nas tabelas A.2, A.3 e A.4 apresentam-se os resultados obtidos para o teste que permite determinar, para cada par de número de clientes e mercados, o número de *Iteracoes.max*. Nas três tabelas, na primeira coluna estão presentes o tipo de dados e a capacidade do veículo. Na segunda coluna está presente o número de *Iteracoes.max* obtido para o respetivo conjunto de dados. Na linha Máximo, está presente, o máximo dos valores de *Iteracoes.max* obtidos a partir dos conjuntos de dados.

Na tabela A.5 apresentam-se os resultados obtidos para a comparação entre a heurística *GRASP1* e *GRASP2*. Na primeira coluna está presente o tipo de dados, o par de número de clientes e mercados, a capacidade do veículo e o número de iterações realizadas. A segunda e terceira coluna apresenta, para cada conjunto de dados, o custo associado à solução obtida correndo a heurística *GRASP1* e *GRASP2*, respetivamente. Na última coluna apresenta-se qual versão da heurística *GRASP* determina a solução associada ao menor custo. Sendo que 1 representa o *GRASP1*, 2 o *GRASP2* e 1/2 nos casos em se obtém o mesmo resultado para ambas as versões da heurística *GRASP*. Na última linha apresenta-se o número de casos em que, a respetiva versão do *GRASP*, determina a solução associada ao menor custo.

Nas tabelas A.7, A.8, A.9 e A.10 apresentam-se, para o Tipo_1, Tipo_2, Tipo_3 e Tipo_4 respetivamente, os resultados computacionais obtidos usando a heurística *GRASP2*. Na primeira coluna está presente o par de número de clientes e mercados, a capacidade do veículo e o número do exemplo. Na tabela A.6 apresenta-se a legenda das outras onze colunas presentes nas tabelas A.7, A.8, A.9 e A.10.

Tabela A.1: Resultados para o teste do parâmetro α

Dados	0.1-*GRASP*1/2	0.2-*GRASP*1/2	0.5-*GRASP*1/2	0.7-*GRASP*1/2
$Tipo_1$-(5,10)-150-250	**245**/245	249/**240**	251/245	249/245
$Tipo_1$-(5,10)-250-250	194/196	190/195	188/187	**179/179**
$Tipo_1$-(10,20)-150-500	435/368	**388/348**	416/373	422/399
$Tipo_1$-(10,20)-250-500	333/270	**319/259**	350/297	324/286
$Tipo_1$-(15,18)-150-600	887/820	**858/808**	900/840	881/856
$Tipo_1$-(15,18)-250-600	500/397	**451/374**	511/449	524/477
$Tipo_2$-(5,10)-150-250	762/704	**738**/697	750/**693**	740/698
$Tipo_2$-(5,10)-250-250	498/479	496/479	501/485	**457/462**
$Tipo_2$-(10,20)-150-500	1289/1253	1286/1249	**1102/1043**	1139/1115
$Tipo_2$-(10,20)-250-500	832/694	814/**684**	831/706	**783**/759
$Tipo_2$-(15,18)-150-600	1606/1300	**1524/1278**	1596/1369	1695/1502
$Tipo_2$-(15,18)-250-600	1171/961	**1121/920**	1145/1047	1224/1062
$Tipo_3$-(5,10)-150-250	243/245	245/246	220/226	**206/198**
$Tipo_3$-(5,10)-250-250	170/171	**169**/171	170/170	170/**169**
$Tipo_3$-(10,20)-150-500	286/**221**	**273**/222	291/256	323/309
$Tipo_3$-(10,20)-250-500	232/**197**	**226**/200	230/218	240/215
$Tipo_3$-(15,18)-150-600	410/326	**404/325**	417/404	451/411
$Tipo_3$-(15,18)-250-600	471/430	413/373	**382/371**	419/401
$Tipo_4$-(5,10)-150-250	407/431	**398**/417	407/**398**	425/407
$Tipo_4$-(5,10)-250-250	283/**265**	**265/265**	284/**265**	283/**265**
$Tipo_4$-(10,20)-150-500	918/**807**	900/820	**891**/857	938/906
$Tipo_4$-(10,20)-250-500	455/411	435/**411**	**434**/423	437/427
$Tipo_4$-(15,18)-150-600	912/808	909/796	896/**794**	**877**/830
$Tipo_4$-(15,18)-250-600	900/813	**894/763**	917/848	922/906
Resultado	5	26	10	10

Tabela A.2: Resultados para o teste do parâmetro *Iteracoes.max* - (5,10)

Dados	*Iteracoes.max*
$Tipo_1$-150	390
$Tipo_1$-250	270
$Tipo_2$-150	10
$Tipo_2$-250	190
$Tipo_3$-150	250
$Tipo_3$-250	30
$Tipo_4$-150	290
$Tipo_4$-250	70
Máximo	390

Tabela A.3: Resultados para o teste do parâmetro *Iteracoes.max* - (10,20)

Dados	*Iteracoes.max*
$Tipo_1$-150	386
$Tipo_1$-250	526
$Tipo_2$-150	226
$Tipo_2$-250	526
$Tipo_3$-150	366
$Tipo_3$-250	286
$Tipo_4$-150	526
$Tipo_4$-250	246
Máximo	526

Tabela A.4: Resultados para o teste do parâmetro *Iteracoes.max* - (15,18)

Dados	*Iteracoes.max*
$Tipo_1$-150	206
$Tipo_1$-250	246
$Tipo_2$-150	186
$Tipo_2$-250	506
$Tipo_3$-150	666
$Tipo_3$-250	366
$Tipo_4$-150	246
$Tipo_4$-250	406
Máximo	666

Tabela A.5: Comparação entre *GRASP*1 e *GRASP*2

Dados	*GRASP1*	*GRASP*2	Resultado
$Tipo_1$-(5,10)-150-250	249	240	2
$Tipo_1$-(5,10)-250-250	190	195	1
$Tipo_1$-(10,20)-150-500	388	348	2
$Tipo_1$-(10,20)-250-500	319	259	2
$Tipo_1$-(15,18)-150-600	858	808	2
$Tipo_1$-(15,18)-250-600	451	374	2
$Tipo_2$-(5,10)-150-250	738	697	2
$Tipo_2$-(5,10)-250-250	496	479	2
$Tipo_2$-(10,20)-150-500	1286	1249	2
$Tipo_2$-(10,20)-250-500	814	684	2
$Tipo_2$-(15,18)-150-600	1524	1278	2
$Tipo_2$-(15,18)-250-600	1121	920	2
$Tipo_3$-(5,10)-150-250	245	246	1
$Tipo_3$-(5,10)-250-250	169	171	1
$Tipo_3$-(10,20)-150-500	273	222	2
$Tipo_3$-(10,20)-250-500	226	200	2
$Tipo_3$-(15,18)-150-600	404	325	2
$Tipo_3$-(15,18)-250-600	413	373	2
$Tipo_4$-(5,10)-150-250	398	417	1
$Tipo_4$-(5,10)-250-250	265	265	1/2
$Tipo_4$-(10,20)-150-500	900	820	2
$Tipo_4$-(10,20)-250-500	435	411	2
$Tipo_4$-(15,18)-150-600	909	796	2
$Tipo_4$-(15,18)-250-600	894	763	2
Total	4	19	

Tabela A.6: Legenda

C-O	Custo da solução S obtida através do algoritmo *GRC*
C-F	Custo da solução S obtida através do algoritmo *LCS*
M-O	Custo da solução R obtida através do algoritmo $APUN2$
M-F	Custo da solução R obtida através do algoritmo *LSM*
C/M-O	Custo da solução T obtida através do algoritmo *LSM*
C/M-F	Custo da solução T obtida através do algoritmo *LSCM*
N-rotas	Número de rotas
1 ou 2	Número de rotas com apenas 1 ou 2 clientes
Tempo	Tempo computacional (em segundos) da heurística *GRASP2*
C-P	Percentagem de melhoria obtida através do algoritmo *LSC*
M-P	Percentagem de melhoria obtida através do algoritmo *LSM*
C/M-P	Percentagem de melhoria obtida através do algoritmo *LSCM*

Tabela A.7: Resultados Computacionais - $Tipo_1$

Dados	C-O	C-F	M-O	M-F	C/M-O	C/M-F	N-rotas	1 ou 2	Tempo(s)	C-P(%)	M-P(%)	C/M-P(%)
(5,10)-150-1	183	183	153	141	251	246	3	3	11.00	0.00	7.84	1.99
(5,10)-150-2	203	182	139	123	245	245	2	1	8.00	10.34	11.51	0.00
(5,10)-150-3	160	160	157	149	245	245	3	3	11.00	0.00	5.10	0.00
(5,10)-150-4	202	202	307	245	348	348	4	4	14.00	0.00	20.20	0.00
(5,10)-150-5	149	149	210	177	266	266	3	3	10.00	0.00	15.71	0.00
(5,10)-250-1	171	154	85	80	180	175	2	1	11.00	9.94	5.88	2.78
(5,10)-250-2	220	206	140	127	269	238	2	1	9.00	6.36	9.29	11.52
(5,10)-250-3	142	139	143	106	202	202	2	1	10.00	2.11	25.87	0.00
(5,10)-250-4	143	143	247	207	298	298	3	3	14.06	0.00	16.19	0.00
(5,10)-250-5	133	132	109	107	195	195	2	1	9.03	0.75	1.83	0.00
(10,20)-150-1	226	226	243	199	350	350	5	3	35.00	0.00	18.11	0.00
(10,20)-150-2	301	301	390	345	522	522	7	7	49.03	0.00	11.54	0.00
(10,20)-150-3	292	282	206	178	354	354	5	4	38.03	3.42	13.59	0.00
(10,20-150-4	327	327	312	272	448	448	6	6	38.00	0.00	12.82	0.00
(10,20)-150-5	332	332	268	242	440	432	6	6	44.03	0.00	9.70	1.82
(10,20)-250-1	225	208	180	143	305	305	3	1	34.03	7.56	20.56	0.00
(10,20)-250-2	257	254	239	165	341	341	4	3	46.00	1.17	30.96	0.00
(10,20)-250-3	223	206	178	124	259	259	3	0	36.00	7.62	30.34	0.00
(10,20)-250-4	237	233	171	144	304	304	3	0	32.03	1.69	15.79	0.00
(10,20)-250-5	270	251	185	170	337	337	4	2	38.03	7.04	8.11	0.00
(15,18)-150-1	373	373	465	418	633	633	9	9	81.19	0.00	10.11	0.00
(15,18)-150-2	315	315	710	613	807	807	8	7	82.25	0.00	13.66	0.00
(15,18)-150-3	498	498	599	513	805	805	10	10	88.00	0.00	14.36	0.00
(15,18)-150-4	286	286	472	375	559	559	8	8	86.09	0.00	20.55	0.00
(15,18)-150-5	453	453	476	410	675	675	10	10	80.06	0.00	13.87	0.00
(15,18)-250-1	304	292	353	312	507	499	5	1	74.00	3.95	11.61	1.58
(15,18)-250-2	254	239	384	348	513	513	5	1	86.00	5.91	9.38	0.0
(15,18)-250-3	351	349	395	340	559	559	6	3	73.03	0.57	13.92	0.00
(15,18)-250-4	226	221	319	254	402	402	5	1	80.09	2.21	20.38	0.00
(15,18)-250-5	275	266	312	241	418	418	5	1	69.06	3.27	22.76	0.00

Tabela A.8: Resultados Computacionais - $Tipo_2$

Dados	C-O	C-F	M-O	M-F	C/M-O	C/M-F	N-rotas	1 ou 2	Tempo(s)	C-P(%)	M-P(%)	C/M-P(%)
(5,10)-150-1	422	422	490	471	694	694	4	4	12.06	0.00	3.88	0.00
(5,10)-150-2	422	422	438	387	655	655	3	3	12.09	0.00	11.64	0.00
(5,10)-150-3	393	393	213	187	446	446	3	3	9.03	0.00	12.21	0.00
(5,10)-150-4	320	320	395	369	562	562	3	3	9.00	0.00	6.58	0.00
(5,10)-150-5	367	367	307	240	490	490	3	3	11.03	0.00	21.82	0.00
(5,10)-250-1	255	255	289	264	434	434	2	1	10.00	0.00	8.65	0.00
(5,10)-250-2	313	313	304	261	479	479	2	1	13.03	0.00	14.14	0.00
(5,10)-250-3	269	269	139	139	321	321	2	1	9.00	0.00	0.00	0.00
(5,10)-250-4	284	284	322	246	442	442	2	1	9.03	0.00	23.60	0.00
(5,10)-250-5	314	314	213	205	433	433	2	1	11.03	0.00	3.76	0.00
(10,20)-150-1	781	781	790	775	1209	1209	6	6	34.03	0.00	1.90	0.00
(10,20)-150-2	484	473	637	495	826	785	5	4	38.00	2.27	22.29	4.96
(10,20)-150-3	571	571	714	607	951	951	7	7	47.06	0.00	14.99	0.00
(10,20-150-4	640	640	453	444	834	834	6	6	39.00	0.00	1.99	0.00
(10,20)-150-5	1330	1330	1159	1046	1711	1711	10	10	49.00	0.00	9.75	0.00
(10,20)-250-1	516	464	410	396	694	694	3	0	33.03	10.08	3.41	0.00
(10,20)-250-2	354	330	386	324	567	567	3	1	33.03	6.78	16.06	0.00
(10,20)-250-3	454	453	453	357	682	677	4	3	44.03	0.22	21.19	0.73
(10,20)-250-4	407	407	365	281	583	583	3	1	44.03	0.00	23.01	0.00
(10,20)-250-5	633	633	554	498	896	896	4	2	42.00	0.00	10.11	0.00
(15,18)-150-1	770	761	608	586	1048	1048	7	5	62.03	1.17	3.62	0.00
(15,18)-150-2	718	718	1000	859	1313	1288	9	9	78.00	0.00	14.10	1.90
(15,18)-150-3	712	712	991	898	1341	1341	8	8	76.03	0.00	9.38	0.00
(15,18)-150-4	854	854	1050	926	1437	1437	8	7	79.03	0.00	11.81	0.00
(15,18)-150-5	1257	1257	1440	1214	1869	1869	12	12	79.03	0.00	15.69	0.00
(15,18)-250-1	547	527	383	380	737	737	4	0	52.06	3.66	0.78	0.00
(15,18)-250-2	613	592	675	550	1002	995	5	1	70.03	3.43	18.52	0.70
(15,18)-250-3	562	548	850	608	984	984	5	1	65.00	2.49	28.47	0.00
(15,18)-250-4	583	559	769	635	976	976	5	1	71.06	4.12	17.43	0.00
(15,18)-250-5	797	784	902	647	1155	1154	6	3	69.00	1.63	28.27	0.09

Tabela A.9: Resultados Computacionais - $Tipo_3$

Dados	C-O	C-F	M-O	M-F	C/M-O	C/M-F	N-rotas	1 ou 2	Tempo(s)	C-P(%)	M-P(%)	C/M-P(%)
(5,10)-150-1	176	176	123	116	244	244	2	1	10.00	0.00	5.69	0.00
(5,10)-150-2	118	118	63	63	147	147	2	1	7.03	0.00	0.00	0.00
(5,10)-150-3	146	145	54	54	171	171	1	0	7.00	0.68	0.00	0.00
(5,10)-150-4	134	119	114	113	188	188	2	1	10.00	11.19	0.88	0.00
(5,10)-150-5	163	160	86	82	182	182	2	1	8.00	1.84	4.65	0.00
(5,10)-250-1	115	114	51	51	125	125	1	0	8.00	0.87	0.00	0.00
(5,10)-250-2	93	93	25	25	110	110	1	0	7.00	0.00	0.00	0.00
(5,10)-250-3	146	145	54	54	171	171	1	0	8.00	0.68	0.00	0.00
(5,10)-250-4	130	101	90	76	166	166	1	0	9.03	22.31	15.56	0.00
(5,10)-250-5	122	117	31	31	127	127	1	0	7.00	4.10	0.00	0.00
(10,20)-150-1	172	163	141	105	224	224	3	1	26.00	5.23	25.53	0.00
(10,20)-150-2	202	196	129	105	230	230	3	1	23.00	2.97	18.60	0.00
(10,20)-150-3	237	224	176	145	316	316	4	2	30.00	5.49	17.61	0.00
(10,20-150-4	192	171	122	99	220	220	3	1	24.00	10.94	18.85	0.00
(10,20)-150-5	195	192	176	159	293	278	4	2	30.03	1.54	9.66	5.12
(10,20)-250-1	157	134	85	79	184	184	2	0	25.03	14.65	7.06	0.00
(10,20)-250-2	180	156	94	85	197	197	2	0	26.03	13.33	9.57	0.00
(10,20)-250-3	229	210	119	101	259	259	3	1	28.03	8.30	15.13	0.00
(10,20)-250-4	169	146	97	68	179	179	2	1	27.00	13.61	29.90	0.00
(10,20)-250-5	158	141	98	86	203	203	2	1	30.00	10.76	12.24	0.00
(15,18)-150-1	281	244	180	168	341	337	4	0	49.00	13.17	6.67	1.17
(15,18)-150-2	246	229	237	218	369	367	4	0	50.03	6.91	8.02	0.54
(15,18)-150-3	255	228	137	123	294	294	4	0	51.09	10.59	10.22	0.00
(15,18)-150-4	246	235	167	148	323	315	4	1	49.03	4.47	11.38	2.48
(15,18)-150-5	453	453	337	296	586	586	6	1	57.09	0.00	12.17	0.00
(15,18)-250-1	265	254	148	132	335	319	3	1	49.03	4.15	10.81	4.78
(15,18)-250-2	191	181	211	184	327	327	3	1	52.06	5.24	12.80	0.00
(15,18)-250-3	202	182	134	106	243	243	3	1	52.00	9.90	20.90	0.00
(15,18)-250-4	243	226	129	99	267	267	3	0	49.00	7.00	23.26	0.00
(15,18)-250-5	285	276	198	166	363	363	3	0	54.00	3.16	16.16	0.00

Tabela A.10: Resultados Computacionais - $Tipo_4$

Dados	C-O	C-F	M-O	M-F	C/M-O	C/M-F	N-rotas	1 ou 2	Tempo(s)	C-P(%)	M-P(%)	C/M-P(%)
(5,10)-150-1	269	269	238	229	417	417	2	1	11.03	0.00	3.78	0.00
(5,10)-150-2	267	214	154	103	226	226	1	0	7.03	19.85	33.12	0.00
(5,10)-150-3	179	170	256	152	274	274	1	0	8.06	5.03	40.63	0.00
(5,10)-150-4	340	310	272	208	406	406	2	1	9.00	8.82	23.53	0.00
(5,10)-150-5	196	175	200	200	313	313	2	1	8.00	10.71	0.00	0.00
(5,10)-250-1	249	219	161	146	323	323	1	0	9.00	12.05	9.32	0.00
(5,10)-250-2	267	214	154	103	226	226	1	0	8.06	19.85	33.12	0.00
(5,10)-250-3	176	170	256	152	274	274	1	0	8.03	3.41	40.63	0.00
(5,10)-250-4	243	213	212	127	265	265	1	0	8.03	12.35	40.09	0.00
(5,10)-250-5	144	132	151	135	235	235	1	0	8.06	8.33	10.60	0.00
(10,20)-150-1	677	650	449	426	819	814	5	4	32.03	3.99	5.12	0.61
(10,20)-150-2	773	773	433	420	875	875	5	2	32.00	0.00	3.00	0.00
(10,20)-150-3	530	502	266	231	581	581	3	0	22.03	5.28	13.16	0.00
(10,20-150-4	416	416	347	302	578	562	5	4	28.00	0.00	12.97	2.77
(10,20)-150-5	400	384	179	171	479	479	3	1	21.03	4.00	4.47	0.00
(10,20)-250-1	477	465	292	268	609	609	3	0	27.03	2.52	8.22	0.00
(10,20)-250-2	529	519	375	277	596	596	3	1	27.03	1.89	26.13	0.00
(10,20)-250-3	462	393	212	190	476	476	2	0	23.06	14.94	10.38	0.00
(10,20)-250-4	344	319	206	181	426	426	3	3	28.00	7.27	12.14	0.00
(10,20)-250-5	370	354	164	141	411	411	2	0	20.00	4.32	14.02	0.00
(15,18)-150-1	526	517	580	471	819	819	5	2	49.00	1.71	18.79	0.00
(15,18)-150-2	639	608	282	282	695	682	4	1	41.00	4.85	0.00	1.87
(15,18)-150-3	437	419	490	438	719	719	5	1	45.03	4.12	10.61	0.00
(15,18)-150-4	761	758	667	555	1008	1005	6	4	61.00	0.39	16.79	0.30
(15,18)-150-5	681	650	574	551	983	983	6	3	55.00	4.55	4.01	0.00
(15,18)-250-1	419	400	300	259	568	568	3	1	48.00	4.53	13.67	0.00
(15,18)-250-2	454	411	216	205	545	543	2	0	44.00	9.47	5.09	0.37
(15,18)-250-3	410	371	276	248	529	529	3	0	45.00	9.51	10.14	0.00
(15,18)-250-4	550	531	542	427	754	754	4	0	61.00	3.45	21.22	0.00
(15,18)-250-5	520	510	384	359	722	722	4	1	52.00	1.92	6.51	0.00