NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE MATEMÁTICA

RUBEN DAVID FERREIRA SALES

Licenciado em Matemática Aplicada à Tecnologia e à Empresa

DESENVOLVIMENTO E APLICAÇÃO DE UMA META-HEURÍSTICA AO HOME HEALTH CARE PROBLEM

MESTRADO EM MATEMÁTICA E APLICAÇÕES: ATUARIADO, ESTATÍSTICA E INVESTIGAÇÃO OPERACIONAL

Universidade NOVA de Lisboa
Março, 2022

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE MATEMÁTICA

DESENVOLVIMENTO E APLICAÇÃO DE UMA META-HEURÍSTICA AO HOME HEALTH CARE PROBLEM

RUBEN DAVID FERREIRA SALES

Licenciado em Matemática Aplicada à Tecnologia e à Empresa

Orientadora: Doutora Maria Isabel Azevedo Rodrigues Gomes
Professora Associada da Faculdade de Ciências e Tecnologia da Universidade NOVA de Lisboa

Júri:

Presidente: Doutora Paula Alexandra da Costa Amaral
Professora Associada da Faculdade de Ciências e Tecnologia da Universidade NOVA de Lisboa

Arguente: Doutor Nelson Fernando Chibeles Pereira Martins
Professor Auxiliar da Faculdade de Ciências e Tecnologia da Universidade NOVA de Lisboa

Orientadora: Doutora Maria Isabel Azevedo Rodrigues Gomes
Professora Associada da Faculdade de Ciências e Tecnologia da Universidade NOVA de Lisboa

MESTRADO EM MATEMÁTICA E APLICAÇÕES: ATUARIADO, ESTATÍSTICA E INVESTIGAÇÃO OPERACIONAL

Universidade NOVA de Lisboa
Março, 2022

Desenvolvimento e aplicação de uma meta-heurística ao Home Health Care Problem

Agradecimentos

Gostaria de agradecer à Prof. Maria Isabel Azevedo Rodrigues Gomes e à Ana Raquel Pena de Aguiar pelas suas orientações durante o desenvolvimento desta dissertação, pelo apoio e paciência nos momentos mais complicados. Estou grato pela enorme disponibilidade e vontade de ajudar.

Um agradecimento também a todos os familiares e amigos pelo apoio ao longo destes meses.

Resumo

Nos últimos anos tem-se assistido a um crescimento na percentagem de população idosa, o que provocou um aumento na procura de cuidados ao domicílio. Como tal, é necessário haver um planeamento eficiente, através da otimização do uso dos recursos, humanos e materiais, bem como das rotas.

O objetivo desta dissertação é desenvolver um algoritmo para melhorar o planeamento das rotas dos cuidados ao domicílio, auxiliando os prestadores de serviço (*caregivers*). A otimização foca na minimização do tempo de viagem. O modelo é uma extensão do *Vehicle Routing Problem* com janelas temporais, sincronização (quando 2 equipas de um *caregiver* se encontram ao mesmo tempo para realizar um serviço que necessita de dois *caregivers*), *caregivers* sem *skills*, para um horizonte temporal de um dia e com um tempo de trabalho diário máximo de 480 minutos por equipa.

De modo a atingir o objetivo, foi aplicado o *Biased Random Key Genetic Algorithm* (BRKGA), o qual é uma extensão do *Genetic Algorithm* (GA).

Para a obtenção dos resultados foi utilizado uma instância de teste com 75 clientes. Para realizar os serviços estão à disposição 15 *caregivers*, entre os quais poderão ser formadas até oito combinações de equipas. As equipas são formadas por equipas de dois *caregivers* e equipas de um *caregiver*. Um objetivo adicional é o de encontrar a melhor combinação de equipas para o problema.

Palavras-chave: *Home Health Care*, Roteamento e Escalonamento, Meta heurística, *Biased Random Key Genetic Algorithm*, Sincronização

Abstract

In the recent years we have seen a growth of the elderly population percentage, which has caused an increase in demand for home care support. As such, there is a need for efficient planning, by optimizing the use of human and material resources, as well as routes.

The goal of this dissertation is to develop an algorithm to improve the planning of home care routes by the caregivers. The optimization focuses on minimizing travel time. The model is an extension of the Vehicle Routing Problem with time windows, synchronization (when two teams of one caregiver meet at the same time do perform a service that needs two caregivers), caregivers without skills, for a time horizon of one day and with a maximum daily working time of 480 minutes per team.

In order to achieve the goal, the Biased Random Key Genetic Algorithm (BRKGA) was applied, which is an extension of the Genetic Algorithm (GA).

A test instance with 75 clients was used to obtain the results. To perform the services 15 caregivers are available, from which can be formed up to eight team combinations. The teams are made up of teams of two caregivers and teams of one caregiver. An additional goal is to find the best combination of teams for the problem.

Key Words: Home Health Care, Routing and Sheduling, Metaheuristic, Biased Random Key Genetic Algorithm, Synchronization

Índice

Índice de Figuras

Índice de Tabelas

1

Introdução

Este capítulo tem como objetivo contextualizar o problema, focando-se nos precedentes do problema em análise, incluindo também o caso de estudo e os objetivos da dissertação.

1.1 Contextualização do Problema

Desde o século XX que se providencia cuidados de saúde ao domicílio, com os médicos a terem a capacidade de transportar todos os medicamentos e recursos necessários para a casa do paciente. No entanto, com os grandes avanços feitos na medicina e das tecnologias associadas, os cuidados de saúde ao domicílio perderam força (Leff & Burton, 1996). Porém, apesar da constante melhoria dos cuidados hospitalares, o hospital pode ser sentido como um local onde os doentes perdem a sua dignidade e intimidade, sendo expostos a um conjunto de pessoas desconhecidas e a um ritmo acelerado da rotina hospitalar (Leff & Burton, 1996).

Com o aumento da esperança média de vida, a população idosa é cada vez mais predominante (Bloom, Canning & Fink, 2011) e este grupo etário está mais predisposto a ser admitido no hospital, apresentando um maior risco para todas as complicações associadas, tais como o risco de apanhar infeções, sofrer quedas, incontinência urinária e declínio funcional. Este aumento também provoca um maior congestionamento nos hospitais, levando a situações de stress e impaciência, o que irá afetar negativamente o aspeto psicológico destas pessoas. Visto tais problemas e sabendo que muitas destas pessoas não têm a autonomia necessária para executar certas tarefas, o que leva à perda de independência e à necessidade de cuidados, os cuidados de saúde ao domicílio mostram ser uma boa solução para conseguir resolver este tipo de ocorrências.

1.2 Caso de Estudo

O principal objectivo desta dissertação é a criação de um algoritmo, o qual possa ajudar a Instituição Particular de Solidariedade Social, APOIO, a melhorar o planeamento dos serviços de cuidados ao domicílio, identificando a ordem das visitas assim como quem

as fará. Este capítulo visa a apresentar o caso de estudo de Aguiar (2017), o qual é uma extensão de Gomes & Ramos (2016).

A APOIO é uma organização sem fins lucrativos que presta serviços ao domicílio e possuí um centro onde alguns pacientes vão passar o dia. Os cuidados ao domicílio englobam serviços como dar medicamentos, ajuda na higiene pessoal, limpeza da casa e a entrega de refeições, previamente preparadas no centro da APOIO. No centro, podem ser fornecidos serviços adicionais mediante um pedido, tais como o transporte de pacientes. Para tal, as visitas aos pacientes foram modeladas como nodos num gráfico. Cada paciente é então definido por um nó e o serviço pago a esse paciente é caracterizado pela duração da visita. Como os idosos valorizam ter uma rotina fixa ao longo da semana e com a intenção de manter essa rotina tanto quanto possível, o início da visita aos pacientes respeita um intervalo de tempo fixo.

A APOIO fornece cerca de 40 serviços diários a 25 pacientes e entrega 100 refeições diariamente. No máximo cada paciente pode requerer quatro serviços. Para clientes que requerem mais do que uma visita, é feito uma réplica do nó para cada serviço adicional, mudando apenas as janelas temporais. Os pacientes são classificados consoante o seu nível de autonomia; os semi-dependentes e os acamados. É esta classificação que indica o número de *caregivers* necessários para a realização dos serviços. Pacientes semi-dependentes necessitam apenas de um *caregiver* enquanto que os acamados necessitam de dois *caregivers*. A APOIO serve 10 pacientes acamados, representando 40% dos pacientes do HHC, os quais representam 55% dos pedidos, com média de 2.2 visitas diárias por paciente. Para os doentes semi-dependentes, há apenas um doente que necessita de mais do que uma visita.

Para a realização dos serviços a APOIO tem disponíveis nove *caregivers* e seis carros, pelo que terá de haver equipas duplas. Atualmente as equipas são compostas por 3 equipas de dois *caregivers* e 3 equipas de um *caregiver*. Os *caregivers* são homogéneos, devido a não ser considerado a preferência de *caregivers*, pelo que não há necessidade de os diferenciar entre os tipos de serviço. Em vez disso, a visita a um paciente caracteriza-se exclusivamente pelo tempo necessário para executar o serviço, o número de assistentes e pela janela de tempo em que esse serviço pode ser iniciado.

Devido ao período de trabalho da APOIO, das 8 horas às 20 horas, e ao tempo de trabalho máximo de cada *caregiver*, 8 horas diárias, torna-se impossível assegurar a continuidade de cuidados. Como não é possível uma equipa realizar todos os serviços diários necessários de um cliente, os quais podem ir até aos quatro serviços, no máximo, a cada cliente serão atribuídas duas equipas. Uma solução seria adicionar uma restrição ao modelo, penalizando a função objetivo no caso de mais do que uma equipa ser atribuída ao mesmo paciente num só dia. Assim, é garantido que é preferível manter a mesma equipa associada a um paciente, a menos que seja impossível face à incompatibilidade entre a última janela de tempo da réplica e o limite do tempo de trabalho da equipa.

Durante o turno de cada equipa, a instituição exigiu uma pausa para almoço que fosse iniciada no centro de dia. Além disso, a hora de início do intervalo deve ter lugar num

intervalo de tempo, definido por uma hora de almoço antecipada e uma hora de almoço posterior, com uma duração pré-definida. Para modelar esta característica, foi criada uma réplica do centro de dia, representando uma actividade caracterizada por uma janela temporal e a duração de, por exemplo, uma hora. Tal restrição não foi tida em conta nesta dissertação. Caso fosse, seria necessário a adição de uma restrição ao modelo que impõe que cada equipa visite esta réplica.

A vasta gama de serviços sociais da organização leva a que os *caregivers* sejam afetados a tarefas para além do âmbito dos serviços do *Home Health Care* (HHC). Uma dessas tarefas é a distribuição do almoço aos clientes ao domicílio, realizada diariamente ao meio-dia e que requer três *caregivers*. Note-se que muitos destes clientes só recebem estas refeições. Espera-se que esta atividade dure uma hora pré-estabelecida, por exemplo, 90 minutos. À semelhança do regulamento anterior, a distribuição é modelada através da geração de uma nova réplica do nó com a respetiva janela de tempo.

Apesar de nos resultados não terem sido usadas as instâncias da APOIO, o algoritmo é facilmente adaptado a ter em conta todas as restrições acima adaptadas, pois não foi tido em conta a pausa para almoço das equipas, nem a valorização da continuidade de cuidados.

1.3 Objetivo e Estrutura da Dissertação

O objetivo desta dissertação é a criação de um algoritmo que seja capaz de resolver o *Home Health Care* (HHC) *Problem*, conseguindo gerar as rotas para o horizonte temporal de um dia, assim como a ordem das visitas e quem as irá realizar. A ter em conta no algoritmo temos as janelas temporais do clientes, a capacidade de todos os *caregivers* serem aptos para visitar qualquer cliente (*caregivers* sem *skills*). Os *caregivers* são divididos em equipas de dois e de um e poderá existir sincronização nas visitas aos pacientes (2 equipas de um *caregiver* podem-se encontrar para realizar serviços em que seja necessário dois *caregivers*). O tempo de trabalho é tido em conta, existindo um tempo máximo de trabalho diário para cada equipa. O algoritmo também tem o objetivo de encontrar a melhor combinação de equipas (imagine-se que temos três *caregivers*, pode haver 3 equipas de um *caregiver* ou 1 equipa de um *caregiver* e 1 equipa de dois *caregivers*), de modo a obter os melhores resultados. A instância usada para os resultados foi o primeiro exemplo de 75 clientes de Mankowska, Meisel & Bierwirth (2014), as quais foram adaptadas ao modelo usado.

A dissertação está estruturada em cinco capítulos.

O capítulo um começa por contextualizar o problema, explicando as motivações e os objetivos, assim como descrever o caso de estudo de Aguiar (2017).

O capítulo dois é a Revisão da Literatura e tem o objetivo de compreender os modelos semelhantes já abordados, assim como as metodologias usadas para os resolver.

Os capítulos três e quatro estão fortemente ligados, enquanto que o capítulo três descreve a meta heurística utilizada para resolver o problema, apresenta o modelo e os

pseudo-códigos, o capítulo quatro é composto pela calibração dos parâmetros do algoritmo e a apresentação dos resultados após a calibração.

O último capítulo incorpora a sistematização de todos os prós e contras desta dissertação, breve resumo dos resultados, para além de sugestões para o trabalho futuro.

2

Revisão da Literatura

2.1 Home Health Care

Envolvendo tanto as necessidades pessoais como médicas que um doente possa necessitar, os cuidados de saúde ao domicílio têm como principal objetivo auxiliar um doente com falta de autonomia com as suas necessidades em casa, permitindo uma melhor qualidade de vida.

O *Home Health Care* (HHC) consiste na prestação de cuidados médicos, paramédicos e serviços sociais aos pacientes ao domicílio. O HHC aumenta a qualidade de vida dos doentes, os quais são assistidos em casa, juntamente com uma redução de custos consideráveis para o sistema de saúde (Comondore et al, 2009). De acordo com Genet et al (2013), na Europa 1% a 5% do orçamento total da saúde pública é gasto em HHC. Deste investimento esperam-se três efeitos: uma diminuição das admissões no hospital, diminuição da duração da hospitalização e aumento da capacidade para os pacientes permanecerem em casa.

O planeamento do HHC pode ser dividido em três áreas: (1) as zonas e os recursos que se possuí, (2) a atribuição dos *caregivers* e (3) o agendamento e definição de rotas (Lanzarone & Matta, 2014). Os principais intervenientes num HHC são (1) os pacientes, (2) as equipas de cuidados de saúde, (3) a entidade responsável pela contratação dos serviços de HHC, (4) a equipa de logística (a qual verifica as necessidades dos pacientes) e (5) as equipas responsáveis pelo bem-estar do paciente (Bricon-Souf et al, 2005).

No planeamento de um serviço de HHC surgem dois problemas principais: o *Operator Assignment Problem* e o *Home Health Care Routing and Scheduling Problem* (HHCRSP). O primeiro consiste em decidir quais os assistentes/cuidadores (*caregivers*) a prestar os serviços de saúde e a que doentes, enquanto que o segundo, o HHCRSP visa agendar as visitas de doentes atribuídas a cada *caregiver*. Estes dois problemas podem ser considerados simultaneamente ou de forma independente. No contexto do transporte, atribuição de rotas e *scheduling*, o grande desafio é a definição de uma estratégia para otimizar a entrega de produtos pelos fornecedores aos clientes. É com base neste problema que surge o *Vehicle Routing Problem* (VRP).

2.1.1 Vehicle Routing Problem

O *Vehicle Routing Problem* (VRP) é um problema de otimização combinatória amplamente estudado, onde o principal foco é a definição da melhor rota para visitar um determinado número de locais, e que ocorre em diversas aplicações, por exemplo, na entrega de gasolina, distribuição de jornais, entrega de alimentos, entre outros (Cissé et al, 2017). De acordo com Bredström & Rönnqvist (2008), o VRP inclui restrições de precedência independentes dos veículos, restrições estas que provocam uma dificuldade acrescida para encontrar soluções admissíveis.

O VRP é geralmente definido através de um grafo $G = (V, A)$ com V o conjunto de vértices e A o conjunto de arestas. O conjunto de vértices representa os clientes a servir e um único depósito a partir do qual o percurso do veículo deve começar e terminar. O principal objetivo é minimizar tanto o número de veículos necessários e minimizar a distância total (ou tempo, dependendo da unidade de medida) percorrida.

O VRP foi inicialmente proposto por Dantzig & Ramse (1959) e a partir daí novas versões foram aparecendo na literatura, das quais se destacam cinco relevantes; (1) *Capacitated Vehicle Routing Problem* (CVRP), em que cada cliente tem uma procura previamente conhecida, a frota de veículos para realizar as rotas de entrega é homogénea e com uma capacidade fixa e cada rota deve partir e chegar ao mesmo depósito (Pillac, Gendreau, Guéret & Medaglia, 2013), (2) *Vehicle Routing Problem with Time Windows* (VRPTW), nesta versão cada cliente i deve ser visitado dentro do intervalo de tempo $[e_i; l_i]$ onde e_i é o tempo mais cedo para iniciar uma visita e l_i é o tempo mais tarde para o seu começo, abordado por Bräysy & Gendreau (2005), (3)*Multi-Depot Vehicle Routing Problem* (MDVRP), semelhante ao VRP mas com múltiplos depósitos, (4) *Open Vehicle Routing Problem* (OVRP), semelhante ao CVRP, mas onde não se exige que o nó final da rota seja o mesmo de partida; (Fleszar, Osman & Hindi, 2009) e (5) *Periodic Vehicle Routing Problem* (PVRP), para além das características típicas do cliente, estão também associadas a um conjunto de datas em que o cliente pode ser servido. A solução do problema reside na atribuição de um horário de visita a cada um dos clientes, para o dia do horizonte temporal (Angelelli & Speranza, 2002).

2.1.2 Home Health Care Routing and Scheduling Problem

O HHCRSP é uma extensão do VRP aumentada por diversas restrições, as quais são específicas no contexto do HHC. Pode ser descrito da seguinte forma; um conjunto de pacientes, dispersos numa área geográfica, os quais necessitam de cuidados, ou seja, visitas em casa, que devem ser prestados pelos cuidadores (*caregivers*). O problema consiste em definir um conjunto de percursos para fornecer os serviços planeados ao longo de um determinado horizonte. Este problema é parecido ao VRP devido ao objetivo principal ser o de fornecer um conjunto de rotas para servir todos os doentes. Segundo Cissé et al (2017), certos estudos dizem que o HHCRSP difere frequentemente do VRP devido à presença de características que geram novas restrições que são mais desafiantes de considerar, como

a continuidade de cuidado, dependências temporais e as *skills* dos *caregivers*. Segundo Mankowska, Meisel & Bierwirth (2014) as principais diferenças entre os problemas considerados são: (1) se se deve assumir uma equipa de funcionários homogénea ou uma equipa heterogénea, (2) se se deve considerar a sincronização de serviços interdependentes ou não, (3) os objetivos a definir no planeamento e (4) a metodologia utilizada para resolver os problemas de otimização resultantes.

As características do sistema de saúde, e consequentemente do HHC, varia de país para país. Nalguns países, a prestação destes serviços é uma responsabilidade da segurança social, noutros é dos municípios ou até mesmo uma mistura das responsabilidades das instituições (Aguiar, 2017). Como tal, a otimização do HHC irá abordar diferentes níveis, o que se reflete nos objetivos e restrições. Nas últimas décadas, foi publicado por vários investigadores modelos matemáticos, que diferem nos níveis previamente falados e também nos métodos de solução utilizados. Esse modelos são *Vehicle Routing Problem with Time Windows* (VRPTW) (Bräysy & Gendreau, 2005), *Multiple Depot Traveling Salesman Problem with Time Windows* (MDTSPTW) (Bektas, 2006), *Periodic Vehicle Routing Problem* (PVRP) (Francis, Smilowitz & Tzur, 2008) e *Periodic Vehicle Routing Problem with Time Windows* (PVRPTW) (Francis, Smilowitz & Tzur, 2008).

Em Fikar & Hirsch (2017) e Cissé et al (2017) algumas das características associadas aos modelos matemáticos do HHCRSP são revistas e as suas características agrupadas. Em termos de restrições foram abordadas as seguintes:

- **Janela temporal**: janela de tempo em que cada cliente deve ser visitado. Também poderá existir para o *caregiver*;

- **Continuidade de cuidado**: assegura que cada paciente é atribuído a um conjunto restrito de *caregivers* durante o horizonte temporal, quer seja ao longo da semana ou se necessitar de mais do que uma visita por dia que seja o mesmo *caregiver* a fazer as várias visitas diárias;

- **Horizonte temporal**: quanto tempo dura o planeamento das rotas. Na revisão de Fikar & Hirsch (2017), concluíram que o contexto de um período único (diário) é o mais estudado na literatura. Contudo, o planeamento diário é um relaxamento do planeamento semanal, uma vez que os serviços são planeados apenas para um dia, e não podem lidar com pausas de descanso entre dias úteis, tempo máximo de trabalho por semana e lealdade (continuidade de cuidado) entre *caregiver* e paciente.

- **Preferência**: o paciente pode ter preferência por certos *caregivers*, quer seja por razões pessoais, incompatibilidade, motivos religiosos, entre outros;

- ***Skill***: corresponde ao facto de para certos serviços, nem todos os *caregivers* terem a competência para os realizar. Quando não há esta diferenciação, todos os *caregivers* são habilitados para realizar qualquer serviço;

- **Equilíbrio do tempo de trabalho**: promove o equilíbrio nas horas de trabalho entre os *caregivers*;
- **Tempo de trabalho**: garante um número máximo de minutos/horas de trabalho que cada *caregiver* poderá realizar no horizonte temporal;
- **Dependência temporal/precedência e serviços disjuntos**: são serviços que podem ter relações entre eles. Por exemplo, um serviço só poderá ser feito dois dias após um certo serviço ou dois serviços não podem ser realizados ao mesmo tempo;
- **Sincronização**: relacionado com a necessidade de um serviço precisar de mais do que um *caregiver* para a sua realização e duas equipas de um *caregiver* encontrarem-se no local à mesma hora para prestarem o serviço.
- ***Breaks***: esta restrição "obriga" o *caregiver* a fazer pausas a meio do serviço, como por exemplo para almoçar.

Os objetivos mencionados nas revisões de Fikar & Hirsch (2017) e Cissé et al (2017) são, para **minimização**:

- A distância viajada ou o tempo da viagem, dependendo da unidade de medida;
- O custo da viagem;
- O tempo de espera;
- O número de *caregivers*;
- As violações de restrições;
- As horas extraordinárias de trabalho;
- O número de serviços por realizar;

E para **maximização**:

- As preferências;
- A satisfação;
- O equilíbrio do tempo de trabalho;
- O número de visitas;
- A continuidade de cuidados.

2.2 Métodos de solução

Nesta secção, vão ser apresentados métodos de solução que têm sido propostos para resolver o HHCRSP. Sendo uma extensão do VRP, o HHCRSP facilmente se torna muito complexo de resolver, apresentando tempos computacionais muito elevados (Mankowska, Meisel & Bierwirth, 2014). Por esta razão, na literatura surgem dois métodos mais comuns para solucionar o problema: algoritmos exatos e meta-heurísticas.

2.2.1 Algoritmos exatos

Diversas famílias de algoritmos exatos foram propostos para resolver o VRP, baseados em *Dynamic Programming* (DP), e *Branch-and-Bound* (BB). Cissé et al (2017), abordaram na revisão feita os modelos recentes, identificaram as principais características, juntamente com as diversas restrições e funções objetivo. Posto isto, apresentaram os métodos desenvolvidos para resolver o HHCRSP e discutiram direções futuras de investigação.

Rasmussen, Justesen, Dohn & Larsen (2012) conceberam uma aproximação *Dynamic Column Generation* (DCG) embutida num *Branch-and-Price framework* (BP) com o objetivo de determinar o horário de menor custos para cada *caregiver* entre os horários admissíveis considerando que as dependências temporais entre as horas de início das visitas têm de ser respeitadas. É introduzido um esquema de *branching* para lidar tanto com as restrições de integralidade como com as de precedência, que são relaxadas no problema principal. Os horários são gerados através de um problema de *pricing*, o qual é idêntico ao percurso do caminho mais curto com janelas temporais. O problema de *pricing* é composto por problemas independentes, como o número de *caregivers*, devido a cada um ter características diferentes.

Trautsamwieser & Hirsch (2014) criaram um algoritmo *branch-price and cut* (BPC) para resolver o HHCRSP para uma semana, no qual os pacientes podem ser visitados várias vezes por semana pelos *caregivers* com as *skills* necessárias para realizar o serviço na janela temporal previamente definida. Os *caregivers* têm a limitação de tempo máximo de trabalho por dia, por semana e por pausas. O problema é baseado numa decomposição *Dantzig-Wolfe* em que o problema principal combina os percursos diários admissíveis dos *caregivers* para definir o horário semanal, minimizando o tempo de trabalho total. Os rotas diárias realizáveis são obtidas através de um problema de *pricing* da decomposição. Um importante contribuição deste trabalho é o facto de se basear num conjunto de *branching rules* aplicadas para obter soluções inteiras. Neste contexto, a solução inicial é obtida com a utilização de um algoritmo *Variable Neighborhood Search* (VNS) utilizado para determinar bons limites superiores ao algoritmo BPC.

Cappanera & Scutellà (2015) desenvolveram uma formulação *Mixed Integer Linear Programming* (MILP) para o planeamento semanal do HHCRSP onde são utilizados padrões de diferentes gerações para visitar os pacientes de acordo com a continuidade dos cuidados e das *skills* dos *caregivers*. Tempo de trabalho máximo dos *caregivers* também é

considerado para se atingir o objetivo de maximizar o equilíbrio do tempo de trabalho.

Wirnitzer, Heckmann, Meyer & Nickel (2016) propuseram um modelo de *mixed integer programming* (MIP) para o planeamento mensal do HHCRSP onde os planos das visitas são predefinidos semanalmente e os *caregivers* são afetados tendo em conta a continuidade de cuidados, disponibilidade, compatibilidade e restrições de capacidade.

Liu, Yuan & Jiang (2017) também propuseram um algoritmo de *branch-and-price* para resolver o HHCRSP, em que cada *caregiver* tem de ter uma pausa para almoço. O problema é decomposto num problema principal usando uma formulação de partição de conjuntos em sub-problemas de *pricing*, os quais são resolvidos através de um algoritmo de *label-correcting* adaptado, para ser capaz de ter em conta as pausas para almoço. Foram propostas várias técnicas de melhoria: *extension rules*, *dominance rules*, estratégias de *cutting-edge column generation* e *hierarchical branching schemes*.

2.2.2 Meta heurísticas

As heurísticas estão associadas aos métodos que, em cada iteração, passam para uma solução melhor na vizinhança da solução anterior, parando quando na vizinhança da atual solução não houver uma melhor solução. Em contraste, as meta-heurísticas permitem a consideração de soluções piores, ou até mesmo soluções não admissíveis. A sua história é falada em Sörensen, Sevaux & Glover (2018).

As meta-heurísticas são métodos de solução que realizam iterações entre procedimentos locais (ou seja, na vizinhança da solução) de melhoria e estratégias de um nível superior para criar um processo capaz de fugir aos ótimos locais (Raidl, Puchinger & Blum, 2010). Uma meta-heurística é também chamado de *framework* de alto nível independente dos problemas uma vez que fornece um conjunto de orientações ou estratégias para desenvolver algoritmos de otimização. O termo é também utilizado para se referir a uma implementação específica de um algoritmo baseado em tal estrutura (ou numa combinação de conceitos de diferentes estruturas) concebido para encontrar uma solução para um problema específico de otimização (Fischetti & Fischetti, 2018). Decompor um problema em sub-problemas menores e desenvolver técnicas especializadas para cada um deles (problema auxiliar) provou ser uma poderosa estratégia de concepção heurística num grande número de ocasiões (Allaoua, Borne, Létocart & Wolfler Calvo, 2013).

Existem diversas formas de abordar os problemas. Acima de tudo, primeiro faz-se uma análise inteligente do problema em questão e das simplificações aceitáveis na sua definição, tenta-se modelar o problema recorrendo a uma formulação matemática que se resolve através de um software de uso geral equipado com um *solver* adequado. Devido à melhoria impressionante dos *solvers* de uso geral nos últimos anos, esta abordagem pode realmente resolver casos de interesse à otimalidade (ou com uma aproximação aceitável) dentro de um tempo de computação razoável. Se não for este o caso, pode-se insistir na abordagem exata e tentar-se obter melhores resultados através da melhoria do modelo e/ou o desenvolvimento de estratégias especializadas (*cutting plans*, *branching*, etc.).

Em alternativa aos métodos exatos, existem então as meta-heurísticas que têm sido extensamente usadas na resolução do HHCRSP.

Nas revisões de Fikar & Hirsch (2017) e Cissé et al (2017) são apresentados muitos destes trabalhos.

Akjiratikarl, Yenradee & Drake (2007) usaram o *Particle Swarm Optimization* (PSO) para resolver o problema. O PSO é uma técnica de *population-based searching* que simula das aves em bandos. Cada indivíduo é chamado de *particle* (partículas) e representa um ponto no espaço de procura. A população chama-se *swarm* (enxame) e representa um conjunto de pontos, os quais são potenciais soluções. O PSO possui uma memória, uma vez que cada partícula se lembra da melhor posição que atingiu. O método combina *local search* com *global search*.

Bräysy, Nakari, Dullaert & Neittaanmäki (2009) abordaram um problema de entrega de refeições a casa dos clientes. Foi modelado como um problema do caixeiro viajante com janelas temporais. Para resolver este problema foi usado uma meta-heurística baseada em *Variable Neighborhood Descent* (VND).

Liu, Xie, Augusto & Rodriguez (2013) abordaram o HHCRSP com *mixed backhauls* onde os medicamentos e dispositivos médicos devem ser entregues a partir da farmácia até à casa dos pacientes e as amostras biológicas recolhidas da casa dos pacientes. Este problema foi resolvido por dois métodos: *Genetic Algorithm* (GA) e *Tabu Search* (TS).

Bard, Shao & Jarrah (2014) aplicaram o *Greedy Randomized Adaptive Search Procedure* (GRASP) para resolver o HHCRSP. Neste problema, consideraram dois tipos de *caregivers*, fisioterapeutas e fisioterapeutas assistentes, dados existirem pacientes que necessitam apenas do fisioterapeuta e outros que necessitam do fisioterapeuta e de pelo menos um fisioterapeuta assistente.

Um algoritmo *Adaptive Variable Neighborhood Search* (AVNS) foi proposto por Mankowska, Meisel & Bierwirth (2014) para resolver o HHCRSP diário. Os serviços terão de ser feitos apenas nas janelas temporais dos cliente, onde existe também um desfasamento de tempo mínimo e máximo entre serviço, com sincronização e onde os serviços exigem *caregivers* com determinadas *skills*.

Kummer, Buriol & de Araújo (2020) aplicaram um *Biased Random Key Genetic Algorithm* (BRKGA) para resolver o VRPTW com *skill* e sincronização. Os clientes podem necessitar até dois serviços distintos. Quando um cliente necessita de dois serviços, os mesmo terão de ser feitos por *caregivers* diferentes. Alguns destes serviços terão de ser feitos simultâneamente, já outros terão de ter um desfasamento temporal entre serviços.

No mundo das meta-heurísticas também é possível usar diferentes meta-heurísticas no mesmo algoritmo, denominando-se métodos híbridos. Foi isso que foi feito por Bertels & Fahle (2006), os quais abordaram o HHCRSP, considerando um modelo onde os pacientes têm uma janela temporal *soft* e uma *hard*, é requerido *caregivers* com *skill* e é considerado a preferência dos *caregivers*. Os autores propõem um algoritmo híbrido baseado em *Constraint Programming* (CP), *Linear Programming* (LP) e *Tabu Search* (TS) ou *Simulated Annealing* (SA).

2.3 Conclusões

A revisão da literatura permitiu uma investigação minuciosa no contexto do HHC. Como foi visto, o HHC é um problema de elevada complexidade e muito particular, pois existem diversas formas de abordar e resolver este problema. As instâncias do mundo real requerem um desenvolvimento das meta-heurísticas com objetivo de obter soluções num período de tempo razoável.

Foram abordadas duas formas de resolver este problema, os algoritmos exatos, como *Dynamic Programming* (DP) e *Branch-and-Bound* (BB), e as meta-heurísticas, como *Particle Swarm Optimization* (PSO), *Variable Neighborhood Descent* (VND), *Genetic Algorithm* (GA), *Tabu Search* (TS), *Greedy Randomized Adaptive Search Procedure* (GRASP), *Adaptive Variable Neighborhood Search* (AVNS) e *Biased Random Key Genetic Algorithm* (BRKGA). Uma terceira abordagem é contudo possível, a qual consiste em utilizar um modelo de optimização linear (por exemplo, um MILP) como um instrumento dentro da *framework* heurística. Esta hibridação do MILP com a meta-heurística leva à abordagem matemática, onde a heurística é construída em torno do modelo MILP. Esta hibridação levou a um novo termo, uma *matheurística* abordada por Fischetti & Fischetti (2018). A *matheurística* tornou-se popular nos últimos anos, como testemunhado pela publicação de volumes dedicados e números especiais de revistas.

No entanto, nem todos os métodos apresentados podem servir de base para abordar o problema abordado neste documento. O algoritmo escolhido para resolver o problema é o *Biased Random Key Genetic Algorithm* (BRKGA), devido às parecenças com o modelo de Kummer, Buriol & de Araújo (2020), à não necessidade de uma população inicial, pois é baseado em *random keys*, ao algoritmo ser de fácil compreensão e aplicação em qualquer *software*, e porque pessoalmente pareceu-me um algoritmo interessante e que se aplicava bem ao modelo. As diferenças deste modelo para o de Kummer, Buriol & de Araújo (2020) são: todos os *caregivers* puderem realizar todos os serviços, pois apenas existe dois tipos de serviços, serviços realizados apenas por um *caregiver* e serviços que necessitam de dois *caregivers*. A composição das equipas também difere pois no modelo deste documento é considerado equipas de um e de dois *caregivers*. Por último a sincronização funciona de forma bastante diferente pois em Kummer, Buriol & de Araújo (2020), havia clientes que necessitavam dos serviços desfasados temporalmente e no modelo desta dissertação os serviços que necessitam de dois *caregivers* são feitos sempre em conjunto. Neste modelo também existe a possibilidade de uma equipa de dois *caregivers* realizar um serviço de um *caregiver*, somando uma penalização do tempo de serviço à função objetivo.

Analisando as características associadas ao HHC propostas em Cissé et al (2017), o modelo abordado tem algumas diferenças em relação ao que se encontra na literatura, uma das principais diferenças diz respeito à continuidade dos cuidados. Outra distinção vem da relação entre a tipologia dos serviços e o nível de competências dos *caregivers*.

3

Meta heurística BRKGA

Este capítulo começa por abordar numa primeira fase o *Genetic Algorithm* antes de explicar o *Biased Random Key Genetic Algorithm*. De seguida é mostrado os parâmetros do algoritmo, bem como o modelo e o pseudo-código.

O termo *Genetic Algorithm*, conhecido por GA, foi primeiramente usado por Holland (1975). O GA é um procedimento iterativo de *local search* aplicado a um conjunto de soluções que se baseia nas propriedades da genética populacional, tentando replicar o paradigma "a sobrevivência do mais apto" ("*survival of the fittest*"). Em cada iteração, o GA descarta algumas soluções de baixa qualidade e gera novas soluções baseadas nas melhores soluções do atual conjunto de soluções. A avaliação da qualidade das soluções é feita numa função específica do problema que é nomeada como *fitness function* (função objetivo/aptidão) (Reeves & Rowe, 2002).

O GA (Reeves, 2010) é inicializado com a matriz da população gerada aleatoriamente (conjunto de cromossomas, *strings*). Depois, para cada cromossoma, é obtido o respetivo valor da função objectivo. Um número pré-determinado de cromossomas é então seleccionado aleatoriamente, e os dois indivíduos com melhor valor de *fitness* são seleccionados. Com os cromossomas escolhidos, são realizadas operações básicas de GA (*crossover* e *mutation*, explicadas adiante) para povoar a geração seguinte (os filhos). O procedimento é repetido durante várias iterações até que uma condição de saída (número máximo de iterações, por exemplo) seja satisfeita.

No caso dos GAs (De Jong, 1993), é utilizada uma população de *strings*, e estas *strings* são frequentemente referidas na literatura do GA como cromossomas. A recombinação das *strings* é realizada utilizando analogias simples de *crossover* e *mutation*, e a pesquisa é orientada pelos resultados da avaliação da função objectivo para cada *string* na população.

GAs aplicam o conceito de "*survival of the fittest*" para encontrar soluções ótimas ou quase ótimas para problemas de otimização combinatória. É feita uma analogia entre uma solução e um indivíduo numa população. Cada indivíduo tem um cromossoma correspondente que codifica a solução. Um cromossoma é constituído por uma cadeia de genes, em que tem associado a ele um valor de *fitness* (De Jong, 1993). Os GAs fazem evoluir um conjunto de indivíduos que constituem uma população ao longo de várias gerações. Em

cada geração, uma nova população é criada combinando elementos da população atual para produzirem descendentes que compõem a próxima geração. A *mutation* também é aplicada em GA na geração da próxima geração, como um meio para escapar aos mínimos locais. O conceito de *"survival of the fittest"* é usado no GA quando os indivíduos são selecionados para se juntarem e produzirem a descendência. Indivíduos são seleccionados aleatoriamente, mas aqueles com melhor valor são preferidos em detrimento daqueles que têm piores valores.

3.1 O Algoritmo BRKGA

Os GAs com *biased random keys* foram introduzidos pela primeira vez por Bean (1994) para resolver problemas de otimização combinatória envolvendo sequenciação. Num BRKGA, os cromossomas são representados como uma cadeia, ou vetor, de números reais gerados aleatoriamente no intervalo [0, 1] (Kummer, Buriol & De Araujo, 2020). Um algoritmo determinístico, chamado descodificador, toma como entrada qualquer cromossoma e associa-lhe uma solução do problema de otimização combinatória. No caso de Bean (1994), o descodificador ordena o vetor de *random keys* e utiliza os índices das chaves ordenadas para representar uma sequência. Como veremos em breve, os descodificadores desempenham um papel importante nos BRKGAs. Na tabela 3.1 está representado um pequeno exemplo de como passar de um vetor de *random keys* para um cromossoma para posteriormente se obter a solução.

Tabela 3.1: De *random keys* para rota

Clientes	1	2	3	4	5	6
Random Keys	**0**	**0.23**	**0.65**	**0.42**	**0.12**	**1**

Como se observa na tabela 3.1, o primeiro e último cliente (número seis) são os depósitos, devido à restrição de um *caregiver* ter de começar e acabar no depósito, e como tal a respetiva *random key* toma o valor de 0 e 1 respetivamente, para quando ordenados o cliente 1 ser o primeiro e, neste exemplo, o cliente 6 ser o último. Assim, a ordem da atribuição dos clientes às equipas neste exemplo é $1 \rightarrow 5 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 6$, sendo o cliente 1 e 6 sempre atribuídos a todas as equipas.

Um BRKGA desenvolve uma população de vetores de *random keys* ao longo de várias gerações. A população inicial é composta por p vetores de *random keys*. Cada gene toma um valor no intervalo real [0,1], gerado aleatoriamente. Após a aptidão de cada indivíduo ser calculada pelo descodificador (função objetivo), a população é dividida em dois grupos de indivíduos: um pequeno grupo de indivíduos (uma percentagem p_e da população) de elite, ou seja, aqueles com os melhores valores de aptidão, e a restante percentagem de

indivíduos $1 - p_e$ de não-elite, tal que $p_e < 1 - p_e$ (Kummer, Buriol & De Araujo, 2020). Para fazer evoluir a população, deve ser produzida uma nova geração de indivíduos. Um BRKGA utiliza uma estratégia elitista, uma vez que todos os indivíduos de elite da geração k são copiados inalterados para a geração $k + 1$. Esta estratégia mantém o registo das boas soluções encontradas durante todas as iterações do algoritmo, resultando numa (espera-se) constante melhoria ao longo das iterações. O critério de paragem usado na aplicação do BRKGA é um limite máximo no número de gerações (iterações do algoritmo) ou quando a melhor solução não mudar durante um certo número de iterações, previamente definido. Quando um dos dois critérios é atingido, o algoritmo pára e devolve a melhor solução encontrada (Gonçalves & Resende, 2011).

No BRKGA (e GA), um mutante é simplesmente um vetor de *random keys* gerado da mesma forma que um elemento da população inicial. Em cada geração é introduzido uma certa percentagem de mutantes na população p_m, os quais irão ser descodificados em soluções admissíveis do problema. Com os indivíduos de elite de $p \cdot p_e$ e os mutantes $p \cdot p_m$ contabilizados na população $k + 1$, é necessário produzir indivíduos adicionais $p \cdot (1 - p_e - p_m)$ para completar os indivíduos p que compõem a população da geração $k + 1$. Isto é feito através da produção de descendentes de $p \cdot (1 - p_e - p_m)$ através do processo de *crossover* (cruzamento). Na figura 3.1 está ilustrado como ocorre a mudança da geração k para $k + 1$.

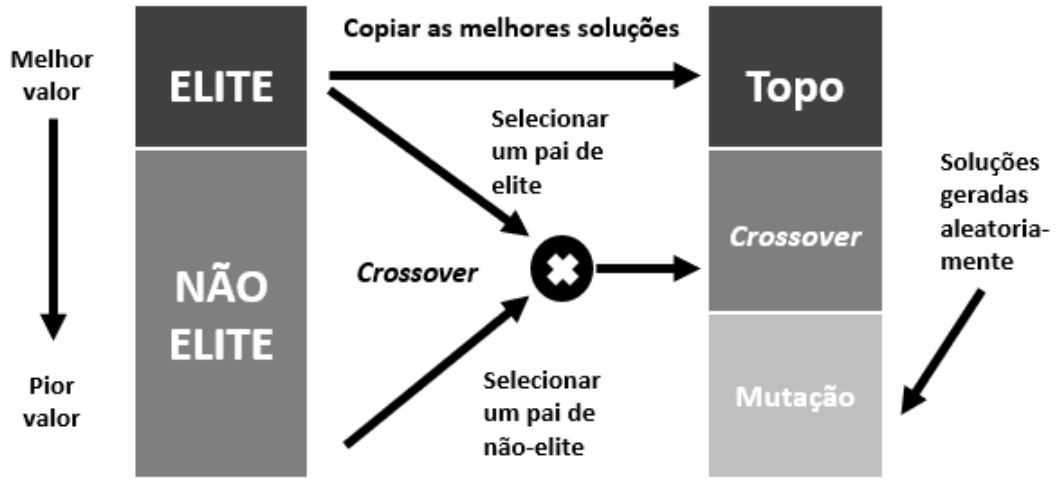


Figura 3.1: Transação da geração k para a $k + 1$ no BRKGA

Como se pode observar na figura 3.1, na parte esquerda está a atual população. Depois de todos os indivíduos serem classificados pelos seus *fitness*, os com melhores valores são colocados na parte ELITE e os restantes indivíduos são colocados na NÃO-ELITE. Os vetores de *random keys* de elite são copiados sem mudança para a próxima população. Vários indivíduos originados pelo operador de mutação são gerados aleatoriamente (*random keys*) e colocados na nova população. O resto da população da geração seguinte é completada através do operador de *crossover*.

Num BRKGA, cada elemento é gerado combinando um elemento selecionado aleatoriamente a partir da partição de ELITE na população atual e outro a partir da partição NÃO-ELITE. Para cada gene (excepto o primeiro e último, pois são o depósito) da descendência gera-se um número aleatório. Seja ρ_e a probabilidade do gene da descendência herdar o gene do seu progenitor de elite. No BRKGA a probabilidade de herdar as características do progenitor de elite é maior do que a probabilidade do progenitor não-elite, donde $\rho_e > 0.5$ (Kummer, Buriol & De Araujo, 2020). Daqui resulta um novo vetor de *random keys*, o qual vai ser descodificado numa solução, que pode ou não ser admissível. Na tabela 3.2 pode-se observar o processo de *crossover* para um vetor com quatro genes.

Tabela 3.2: Exemplo de *crossover*

Cromossoma Elite	0.56	0.43	0.12	0.89
Cromossoma Não-Elite	0.23	0.18	0.62	0.7
Número Aleatório	0.06	0.48	0.60	0.67
Probabilidade de crossover do alelo 0.65	<	<	<	>
Cromossoma resultante do crossover	0.56	0.43	0.12	0.7

Como se pode verificar na tabela 3.2, o primeiro cromossoma refere-se ao indivíduo de elite e o cromossoma 2 ao de não-elite. Neste exemplo, o valor de ρ_e é 0.65, ou seja, o gene em questão do cromossoma descendente herda o gene do progenitor de elite com 65% de probabilidade e do outro progenitor com 35% de probabilidade. Um número é gerado aleatoriamente no intervalo [0, 1] para cada gene, se o resultado for inferior ou igual a 0.65, então o descendente herda a *random key* do cromossoma de elite, caso contrário, herda a *random key* do outro progenitor. Neste exemplo, a descendência herda a *random key* do progenitor de elite nos seus primeiro, segundo e terceiro genes. Quando a população seguinte estiver completa, ou seja, quando tiver p indivíduos, os valores de aptidão são calculados novamente para todos os vetores de *random keys* recentemente criados e a população é novamente dividida em indivíduos de elite e não-elite para iniciar uma nova geração.

3.2 Selecção de parâmetros

Os GA de *random keys* têm parâmetros que necessitam de ser previamente definidos. Estes parâmetros são a dimensão da população p, a percentagem da população de elite p_e, a percentagem da população gerada aleatoriamente (através da *mutation*) p_m, e a probabilidade de herança do gene de elite ρ_e. Gonçalves & Resende (2011) propuseram empiricamente

intervalos de valores recomendados para os parâmetros. Na tabela 3.3 está apresentado esses intervalos para os parâmetros.

Tabela 3.3: Definições dos valores recomendados dos parâmetros

Parâmetros	**Descrição**	**Valor Recomendado**
p	Dimensão da população	$p = an$, onde $a \geq 1 \in \mathbb{R}$ e n o tamanho do cromossoma
p_e	Percentagem da população de elite	$0.1 \leq p_e \leq 0.25$
p_m	Percentagem da população mutante	$0.1 \leq p_m \leq 0.30$
ρ_e	Probabilidade de herança do gene de elite	$0.5 \leq \rho_e \leq 0.8$

3.3 Modelo

3.3.1 Conjuntos

O modelo é composto por dois tipos de entidades: os nós (V) e as equipas (U). As equipas são definidas por um conjunto $a \in U = 1,...,u$, representando as u equipas existentes, que podem ser constituídas por um ou dois assistentes. Por outro lado, o conjunto de nós tem vários subconjuntos a ele associados. Por uma questão de clareza, dada a sua complexidade, o grupo de subconjuntos é detalhado na tabela 3.4:

Tabela 3.4: Definição de nós

V	Conjunto de todos os nós;
V_R	Subconjunto dos nós de partida, $i \in \mathrm{V}_R \in V$;
V_F	Subconjunto dos nós de chegada, $i \in V_F \in V$;
V_P	Subconjunto de pacientes e todas as réplicas, caso existam, $i \in V_P \in V$;
R_i	Subconjunto composto por subconjuntos, cada um contendo exclusivamente o paciente i e as suas réplicas, $\mathrm{R}_i \subset V_P$;

Em primeiro lugar, V_R é composto pelos nós que servem de ponto de partida das equipas. Todas as rotas começam neste nó. Por outro lado, V_F é o subconjunto de nós que servem de ponto de partida para a chegada de todas as equipas. Depois, há um subconjunto associado aos pacientes e às suas réplicas correspondentes, V_P. Finalmente, o subconjunto R é a união de todos os subconjuntos que contêm o paciente e as suas réplicas. Este subconjunto permite a modelação das restrições de preferência da fidelidade. Neste modelo o tempo não é modelado como um conjunto, pois o horizonte temporal deste problema é de apenas um dia.

A rede criada pelo modelo inclui dois elementos, os nós e as ligações entre eles, normalmente conhecidos como arcos. O conjunto que abrange a totalidade das ligações entre os nós é A. Cada arco depende de três dimensões: o nó de partida $i \in V$, o nó de chegada $j \in V$. O conjunto de arcos é também dividido em classes, os quais podem ser reconhecidos através da tabela 3.5:

Tabela 3.5: Definição de arcos

A_R	Arcos válidos de V_R a V_P ou V_F, $(i,j) \in A_R \subset A$;
A_F	Arcos válidos de V_R ou V_P a V_F, $(i,j) \in A_F \subset A$;
A_P	Arcos válidos de V_P a V_P, $(i,j) \in A_P \subset A$;
A_{all}	Todos os arcos válidos, $(i,j) \in A_{all} \subset A$;

A inclusão de um arco num conjunto tem essencialmente em consideração dois parâmetros, denominado nó visitado e a medida de viabilidade do calendário. O primeiro, nó visitado, é representado por NV_i, e indica se um nó é ou não visitado. A principal razão que suporta a sua consideração é que se um nó no arco não for visitado, não há necessidade de o incluir no conjunto. A natureza dos subconjuntos dos arcos inerentes ao modelo são subsequentemente definidos. Os arcos A_R começam no depósito e chegam a um depósito fictício ou a um paciente. Os arcos que partem do depósito ou de um paciente e chegam ao depósito fictício pertencem ao subconjunto A_F. O terceiro subconjunto, A_P inclui apenas os arcos que saem e chegam aos pacientes. Finalmente, o subconjunto A_{all} é formado por todos os arcos anteriormente considerados, $A_{all} = A_R + A_F + A_P$.

3.3.2 Dados do problema

Na tabela 3.6 está representado todos os dados do modelo.

Tabela 3.6: Dados do problema

D_{ij}	Tempo de viagem entre os nós i e j;
w_i	Duração da visita no nó i;
NV_i	Nó visitado: = 1 se o paciente i for visitado;
NS_i	Número de cuidadores que o paciente i necessita;
H	Tempo de trabalho máximo diário de cada equipa;
e_i	Início da janela temporal do paciente i;
l_i	Final da janela temporal do paciente i;
M_{ij}	M grande, um valor suficientemente grande para assegurar a viabilidade da restrição;
NN	Número de nós (depósito é contabilizado 2 vezes);
NE	Número de equipas;

3.3.3 Variáveis

Devem ser tomadas duas decisões. Uma diz respeito à sequência em que os nós são visitados e a outra atribuição duma equipa a um paciente. Por conseguinte, foram definidas três variáveis:

- x_{ija} variável binária que é igual a 1 se a equipa a for atribuída ao arco (i , j); e 0 caso contrário. Esta variável fornecerá os nós da sequência de visita da equipa a;

- S_{ia} uma variável contínua que define a hora de início da visita da equipa a ao nó i;

- NS_i variável binária que é igual 0 se for necessário apenas 1 *caregiver* para realizar o serviço e igual a 1 se for preciso dois *caregivers*;

3.3.4 Formulação do modelo

A característica essencial de um modelo de programação linear são as relações matemáticas que modelam os problemas do mundo real. Na otimização, estas assumem duas naturezas: ou são a função objetivo ou as restrições. A primeira representa uma medida da vantagem ou desvantagem atribuída a uma solução viável, enquanto que as restrições são as condições que devem ser satisfeitas pelas soluções viáveis. As restrições ditarão a forma como a solução é condicionada pelas características do problema real. Recorrendo à nomenclatura anteriormente introduzida, a formulação matemática do modelo é apresentada nesta subsecção. A equação 3.1 representa a função objetivo.

$$min \sum_{i,j,a} ((D_{ij} * x_{ija}) + (w_i * NS_i * x_{ija})) \tag{3.1}$$

O objetivo da função objetivo é minimizar o tempo de viagem (primeiro termo). Contudo, um termo adicional foi tido em conta, como resultado da visita de uma equipa composta por dois *caregivers* a um cliente que necessite apenas de 1 equipa de um *caregiver*, acumulando um valor de penalização igual ao tempo de serviço cada vez que esta situação se verifique.

Como restrições tem-se que todos os nós devem ser visitados apenas uma vez pelas equipas (equação 3.2), todas as equipas devem partir do depósito (equação 3.3), todas as equipas devem chegar ao depósito no final da rota (equação 3.4), todas as equipas que entram num nó devem sair do mesmo (equação 3.5), a equação 3.6 assegura que se a mesma equipa visita os nós i e j, as horas de início correspondentes permitem a deslocação entre os nós (D_{ij}) e o tempo de trabalho no nó i (se uma equipa não visita estes dois nós, M_{ij} é um valor suficientemente grande para tornar a restrição redundante), a equação 3.7 assegura a cada equipa que o tempo de trabalho diário de H minutos é respeitado, a não negatividade (equação 3.8) e as equações 3.9 e 3.10 com que as variáveis sejam binárias.

$$\sum_a \sum_j x_{ija} = 1, \forall i \in V_P : NV_i = 1 \tag{3.2}$$

$$\sum_{j \in V_p} x_{ija} = 1, \forall a \in U, \forall i \in V_R \tag{3.3}$$

$$\sum_{i \in V_p} x_{ija} = 1, \forall a \in U, \forall j \in V_F \tag{3.4}$$

$$\sum_i x_{ija} - \sum_i x_{ija} = 0, \forall a \in U, \forall j \in A_P \cup A_L \tag{3.5}$$

$$s_{ia} + D_{ij} + w_{ij} + M_{ij}(1 - x_{ija}) \leq s_{ja}, \forall a \in U, \forall i, j \in V \tag{3.6}$$

$$s_{ia} - s_{ja} \leq H, \forall a \in U, \forall i \in V_f, j \in V_R \tag{3.7}$$

$$s_{ia} \geq 0, \forall a \in U, \forall i \in V \tag{3.8}$$

$$x_{ija} \in \{0,1\}, \forall a \in U, \forall i,j \in V \tag{3.9}$$

$$NS_i \in \{0,1\}, \forall i \in V \tag{3.10}$$

Existe uma restrição adicional, que tem a ver com o facto de que uma equipa de dois *caregivers* possa realizar um serviço em que apenas é necessário um *caregiver* e que duas equipas de um *caregiver* podem realizar serviços em que é preciso dois *caregivers* (sincronização). Embora esta restrição não esteja enumerada acima, no código do cálculo da rota é tida em conta.

3.4 Pseudo-código

Neste capítulo será apresentado e explicado o pseudo-código do BRKGA (algoritmo 1) e do cálculo do menor custo da rota (algoritmo 2).

O algoritmo BRKGA tem como *input data0*, que guarda o tamanho de cada cromossoma (número de clientes mais duas vezes o depósito), p, o número de cromossomas, p_e, a percentagem de população de elite, p_m, a percentagem de população mutante, ρ_e, a probabilidade de herdar as características do progenitor de elite no operador *crossover*, *iter* o número de iterações que, no máximo, o BRKGA irá correr e *stop*, o qual é um critério de paragem caso a melhor solução não mude *stop* vezes. Como *output* irá guardar o custo da rota com melhor valor da função objetivo, assim como a respetiva rota.

O algoritmo começa por criar duas listas (*Clientes* e *Clientes2*), onde irão ser guardadas as rotas (cromossomas) (linha 1 e 2), uma lista denominada *RandomKeys* que guarda as *random keys* (linha 3) e um *array Fitness*, o qual irá guardar o valor da função objetivo das rotas guardadas em *Clientes* (linha 4).

Posteriormente será gerado aleatoriamente p vetores de *random keys* (linha 5). A primeira *random key* de cada vetor toma o valor 0 e a última o valor 1. De seguida ordena-se as *random keys* do menor valor para o maior e ordena-se os clientes da mesma forma e guarda-se na lista *Clientes* (linha 6), assim obtemos a ordem das visitas para o algoritmo *CalcularRota*. A tabela 3.1 mostra um exemplo de como funciona este processo.

Da linha 7 à 31 o código irá realizar *iter* gerações. Para cada iteração é calculado o valor da função objetivo para todas as rotas da lista *Clientes* através do algoritmo *CalcularRota*, com os valores a serem guardados, respetivamente, no *array Fitness* (linha 8) e a lista *Clientes* a ser ordenada do melhor para o pior valor da função objetivo (linha 9).

Na linha 10 é guardado o melhor valor do *array Fitness* na instância *Best*. Se o valor de *Best* for o mesmo durante *stop* vezes, o BRKGA termina e retorna o *output* (linha 11 a 13). As primeiras $p \cdot p_e$ rotas, que devido à ordenação da linha 8 são as melhores rotas, são guardadas em *Clientes2* (linha 14).

Da linha 15 à 28 o código irá gerar $p \cdot (1 - p_e - p_m)$ cromossomas a partir do operador *crossover*, ilustrado na tabela 3.2. Para cada cromossoma gerado é escolhido um pai da população de elite e um pai da população de não-elite da lista *Clientes* (linha 16 e 17).

O *array Arr* serve para guardar as *random keys* para cada cromossoma gerado através do operador *crossover* (linha 18), para posteriormente ser guardado na lista *RandomKeys*.

Da linha 19 à 26 é o processo de *crossover* para cada cromossoma. Para cada posição h de *arr* (excepto a primeira e última, pois são o depósito) é gerado um número aleatório r (linha 20), se r for menor que ρ_e, na posição h de *arr* guarda-se o valor da *random key* da posição h do pai de elite selecionado (linha 21 e 22), se r for maior que ρ_e, na posição h de *arr* guarda-se o valor da *random key* da posição h do pai de não-elite escolhido (linha 23 e 24). Quando for percorrido todas as posições do *array*, guarda-se *arr* em *RandomKeys* e a respetiva rota em *Clientes2* (linha 27) e volta-se à linha 16 para repetir o processo, até serem gerados $p \cdot (1 - p_e - p_m)$ cromossomas através do operador *crossover*.

Para completar a lista *Clientes2* falta gerar aleatoriamente $p \cdot p_m$ novos vetores de *random keys*, guardar na lista *RandomKeys* e guardar a respetiva rota em *Clientes2* (linha 29). Iguala-se *Clientes2* a *Clientes* (linha 29) e volta-se à linha 8, caso ainda não tenha ocorrido *iter* gerações.

Quando algum critério de paragem for atingido, o código retorna a rota com o melhor valor da função objetivo e o respetivo valor (linha 32).

Algorithm 1 BRKGA

Input: data0, p, p_e, p_m, ρ_e, iter, stop
Output: Custo da rota com melhor valor de *fitness* e respetiva rota

```
Clientes ← Lista que guarda as rotas para cada cromossoma
Clientes2 ← Lista auxiliar que guarda as rotas para cada cromossoma
RandomKeys ← Lista que guarda as random keys para cada cromossoma
Fitness ← Array que guarda o valor fitness das rotas
Gerar aleatoriamente a população inicial (random keys) e guardar em RandomKeys
Passar os vetores da lista RandomKeys para rotas e guardar na lista Clientes
for i in range(iter) do
    Calcular o fitness de cada rota usando a função CalcularRota(Clientes) e guardar na lista Fitness
    Ordenar os Clientes consoante o respetivo valor fitness, do melhor para o pior
    Best ← Guarda o melhor valor do array Fitness
    if Best não mudou nas últimas stop vezes then
        break
    end if
    Guardar as p x p_e melhores rotas (população de elite) em Clientes2
    for j ← p x p_e to p - p x p_m do
        Escolher aleatoriamente um pai da população de elite de Clientes
        Escolher aleatoriamente um pai da população de não elite de Clientes
        Arr ← array auxiliar de dimensão data0 que vai guardar os novos arrays originados pelo crossover
        for h in range(1, data0 - 1) do
            Gerar um número aleatório r
            if r ≤ ρ_e then
                Guardar em Arr na posição h o valor da random key do pai de elite
            else
                Guardar em Arr na posição h o valor da random key do pai de não elite
            end if
        end for
        Guardar Arr em RandomKeys, passar o vetor para rota e guardar em Clientes2
    end for
    Gerar aleatoriamente p x p_m mutantes, guardar as random keys e RandomKeys e a respetiva rota em Clientes2
    Igualar Clientes a Clientes2
end for
return min(Fitness) e a respetiva rota da lista Clientes
```

O segundo algoritmo *CalcularRota* é o algoritmo que para cada *array* de cromossomas, obtido através do algoritmo BRKGA, vai calcular a rota de menor custo e a atribuição para cada equipa, assim como os horários das visitas (*output*). Este algoritmo foi construído de raiz com base numa heurística *greedy*, que leva a que o próximo cliente a visitar se atribua à equipa mais perto no momento.

O descodificador recebe como *input* um *array Clientes*, *array* que contem a rota, *DoubleServ*, *array* que para cada cliente indica se necessita de dois (DoubleServ:=1) ou de um (DoubleServ:=0) *caregiver*, *DistMat*, matriz que guarda a distância, em minutos, entre

cada cliente (relembrando que o primeiro e último cliente é o depósito), *ServTime*, *array* que guarda o tempo de serviço por cliente, *MinWind*, *array* que guarda o tempo mais cedo para começar o serviço para cada cliente, *MaxWind*, *array* que guarda o tempo mais tarde para começar o serviço para cada cliente, *MaxWorkTime*, instância que guarda o tempo de trabalho máximo por equipa.

Na linha 1 é criado um *array*, *Penalização*, que para cada equipa de dois *caregivers*, irá somar uma penalização igual ao *ServTime* caso a equipa em questão vá realizar um serviço a um cliente que necessita apenas de um *caregiver* (nas equipas de um *caregiver* o valor da penalização vai ser sempre zero). Na linha 2 é criado outro *array*, chamado *DistArr*, que guarda a distância percorrida por cada equipa. Na linha 3 é adicionado o cliente um (depósito) à rota de todas as equipas.

Da linha 4 à 19 é o calculo da rota de menor custo para o cromossoma *Clientes*. Na linha 4 é escolhido o cliente *i* a visitar, pela ordem previamente definida do cromossoma *Clientes* (não contando o primeiro e o último, pois são os depósitos). Numa primeira fase (linha 5) é verificado se há equipas que conseguem realizar o serviço dentro das janelas temporais, no caso de nenhuma equipa conseguir realizar o serviço (linha 6), o cliente *i* é atribuído à equipa mais perto (no caso de necessitar de dois *caregivers* para a realização do serviço, a equipa de dois *caregivers* ou as 2 equipas de um *caregiver* mais perto, no caso de 2 equipas de um, é a soma da distância dos dois *caregivers* ao cliente *i*), somar um valor muito grande à posição do *array DistArr* da equipa em questão, de modo a que no BRKGA o valor da função objetivo da rota em questão tenha um mau valor e a rota acabe por ser excluída da geração seguinte, e passar para o próximo cliente (linha 7).

Se houver equipas possíveis para realizar o serviço na janela temporal do cliente *i* e o cliente *i* necessitar de dois *caregivers* (linha 9), é escolhido a equipa de dois ou as 2 equipas de um em que, a distância do último cliente visitado ao cliente *i* (no caso de 2 equipas de um, é a soma da distância dos dois *caregivers*) seja mínima (linha 10). O cliente *i* é adicionado à rota da equipa ou das equipas atribuídas (linha 11).

Se houver equipas possíveis para realizar o serviço e o cliente *i* apenas necessitar de um *caregiver* (linha 12), é escolhido a equipa em que a distância do último cliente visitado a *i* seja mínima (linha 13. Adiciona-se o cliente *i* à rota da respetiva equipa (linha 14). Se neste caso, a equipa atribuída for uma equipa de dois *caregivers* (linha 15), é somado à posição da equipa no *array Penalização* o valor do tempo de serviço do cliente *i* (linha 16).

Repetir da linha 4 à 19 até que todos os clientes sejam atribuídos a uma equipa. Adicionar o último cliente (depósito) à rota de todas as equipas (linha 20) e calcular a distância percorrida por cada equipa e somar a respetiva posição em *DistArr* (linha 21).

Verificar se nenhuma equipa excede o tempo máximo de trabalho (linha 23). Caso alguma equipa exceda, é somado à respetiva posição do *array DistArr* um valor muito grande de forma a que no BRKGA a rota tenho um mau valor *fitness* e acabe por ser excluída da geração seguinte.

Por fim (linha 26) é retornado o valor da função objetivo (soma dos *arrays DistArr* e *Penalização*), bem como a ordem e horários das visitas para cada equipa.

Algorithm 2 CalcularRota

Input: Clientes, DoubleServ, DistMat, ServTime, MinWind, MaxWind, DoubleTeam, MaxWorkTime

Output: Custo da melhor rota com base em Clientes e atribuição da rota de cada equipa

Penalização ← Array que para cada equipa soma uma penalização igual ao ServTime do cliente em questão, caso uma equipa de 2 realize um serviço que de apenas um *caregiver*
DistArr ← *Array* que guarda a distância percorrida para cada equipa de *caregivers*
Adicionar o cliente 1 (depósito) à rota de todas as equipas
for i **in** Clientes **do** ▷ Vai percorrer todos os clientes pela ordem definida a partir das *random keys*, excepto o primeiro e último (depósito)
    Verificar as equipas que conseguem realizar o serviço do cliente *i* dentro das janelas temporais (entre MinWind e MaxWind)
    if Nenhuma equipa for elegível a realizar o serviço **then**
        Atribuir o cliente *i* à equipa mais perto (no caso de necessitar de dois *caregivers*, a equipa de dois ou às duas equipas de um mais perto, no caso de duas equipas de um, é a soma da distância dos dois *caregivers* ao cliente *i*), somar um valor muito grande à posição do *array DistArr* da equipa em questão e passar para o próximo cliente
    end if
    if DoubleServ[i]==1 **then** ▷ O cliente *i* necessita 2 *caregivers*, ou uma equipa de 2 ou duas equipas de 1
        Escolher, entre as equipas elegíveis a realizar o serviço, a equipa de 2 ou as duas equipas de 1 em que, a distância do último cliente visitado ao cliente *i* seja mínima, usando a lista DistMat (no caso de duas equipas de 1, é a soma da distância dos dois *caregivers* ao cliente *i*)
        Adicionar o cliente *i* à rota da ou das equipas escolhidas
    else ▷ Necessita 1 *caregiver*, caso a equipa mais perto seja uma equipa de 2, irá essa equipa realizar o serviço, com uma penalização de ServTime[i]
        Escolher, entre as equipas elegíveis a realizar o serviço, a equipa em que a distância do último cliente visitado a *i* é mínima, usando a lista DistMat
        Adicionar o cliente *i* à rota da ou das equipas escolhidas
        if Se for uma equipa de 2 caregivers **then**
            Somar ServTime[i] à variável Penalização
        end if
    end if
end for
Adicionar o último cliente (depósito) à rota de todas as equipas
Calcular a distância percorrida por cada equipa e guardar em DistArr
Verificar se todas as equipas não excedem o MaxWorkTime
if Se houver uma equipa em que o tempo de serviço seja maior que MaxWorkTime **then**
    Somar ao DistArr da respetiva equipa um valor muito grande, de forma a que no BRKGA a rota seja excluída
end if
return soma(DistArr + Penalização), a ordem das visitas e os horários das visitas de cada equipa

4

Resultados

Este capítulo apresenta a calibração dos parâmetros, na secção 4.1, com o objetivo de obter os resultados. A secção 4.2 apresenta as características do exemplo usado e a secção 4.3 apresenta os diversos resultados obtidos e as conclusões retiradas.

4.1 Calibração dos parâmetros

Esta secção tem como objetivo afinar os parâmetros a usar no algoritmo. Para tal, utilizou-se as instâncias de 10 clientes de Mankowska, Meisel & Bierwirth (2014), adaptadas a este modelo. Como critério de paragem, o código irá terminar ou após 1000 iterações ou quando durante 50 iterações a melhor solução não mudar. O limite de trabalho diário de uma equipa é de 480 minutos.

Após verificar a tabela 3.3, foram escolhidos certos valores para cada parâmetro, dentro dos respetivos intervalos. Para p vai-se testar com os valores 50, 100 e 200, para p_e com os valores 0.15, 0.2 e 0.25, para $p_m = 0.1, 0.2, 0.3$ e para ρ_e com os valores 0.6, 0.7 e 0.8. Com estes valores, é possível fazer 81 combinações, em que para cada combinação foi guardado o número de iterações para comparação. Para cada combinação o código fez apenas uma corrida. Nesta análise foi usado apenas o primeiro exemplo de 10 instâncias de Mankowska, Meisel & Bierwirth (2014). Na tabela 4.1 pode-se verificar o número total de iterações e a média para cada valor de p.

Tabela 4.1: Iterações para p

P	50	100	200
Total	876	662	409
Média	32,4	24,5	15,1

Como se pode observar na tabela 4.1, os testes para $p = 50$ apresentaram um número de iterações de mais do dobro das iterações para populações de dimensão 200 e perto de quatro terços quando comparado com populações de 100 cromossomas. Posto isto, populações de dimensão 50 serão excluídas nas comparações de iterações dos restantes parâmetros.

Na tabela 4.2 está representado a média de iterações para p_e, p_m e ρ_e para p igual a 100 e 200, pois $p = 50$ já foi excluído.

Tabela 4.2: Iterações para p_e, p_m e ρ_e

	Média			Média			Média	
ρe	**P = 100**	**P = 200**	**Pe**	**P = 100**	**P = 200**	**Pm**	**P = 100**	**P = 200**
0.6	23,2	19,2	**0.15**	17,4	13,4	**0.1**	38,2	18,2
0.7	27,9	15,0	**0.2**	30,8	16,9	**0.2**	21,2	17,3
0.8	22,4	11,2	**0.25**	25,3	15,1	**0.3**	14,1	9,9

Observando a tabela 4.2, verifica-se que os valores de ρ_e, tanto para p igual a 100 como 200, o valor $p_e = 0.8$ é o que registou um menor número médio de iterações para atingir o ótimo. Para p_m verifica-se claramente que o valor 0.3 é o que obtém a menor média de iterações. No parâmetro p_e, principalmente para $p = 200$ a média de iterações é muito idêntica para todos os valores de p_e, não sendo retirado deste parâmetro grandes conclusões.

Da análise anterior acabou por não se conseguir concluir relativamente a que valor de p_e escolher. Como tal, será feita uma nova calibração testando para os 10 exemplos de 10 instâncias de Mankowska, Meisel & Bierwirth (2014). Será analisado o número de iterações, o tempo de corrida (em segundos) e a percentagem e distância (em minutos) para as soluções ótimas encontradas (obtidas através da resolução exata do problema), com $p = 100, 200$ e com $p_e = 0.1, 0.15, 0.2$ e 0.2, os quais estão representados, respetivamente nas tabelas 4.3, 4.4 e 4.5. Para que os resultados fossem melhor comparáveis, a semente foi fixada, ou seja, a população inicial vai ser sempre a mesma para qualquer teste, partindo sempre da mesma "casa de partida", tornando a população inicial pseudo-aleatória.

Tabela 4.3: Resultados das iterações para p_e

	Média de iterações	
Pe	**P = 100**	**P = 200**
0.1	15,3	9,9
0.15	16,7	10,8
0.2	16,1	20,8
0.25	23,1	19,7
Média Total	17,8	15,3

Verificando a tabela 4.3, podemos verificar que para $p = 200$, obteve-se a solução num menor número de iterações, e que de um modo geral para valores mais baixos de p_e, menos iterações. Também se observa que para p_e a tomar os valores 0.1 e 0.15 os resultados são idênticos.

Tabela 4.4: Resultados dos tempos de computação, em segundos

			Média de tempo por iteração			
P	**Média tempo**	**Tempo/Iteração**	**Pe = 0.1**	**Pe = 0.15**	**Pe = 0.2**	**Pe = 0.25**
100	5,8	0,3	5,6	5,7	5,7	6,2
200	10,9	0,7	9,9	10,3	11,3	11,9

Em relação ao tempo de computação, medido em segundos, na tabela 4.4 pode-se verificar novamente que para $p_e = 0.1, 0.15$ o tempo de computação são os mais baixos e muito idênticos (para o mesmo p). Em relação ao p, para a população com dimensão 200 demora pouco menos do dobro do tempo quando comparado com uma população de 100 cromossomas.

Tabela 4.5: Distância e percentagem de soluções ótimas

	Soma da distância ao valor ótimo		
Pe	**P=100**	**P=200**	**% Valor ótimo**
0.1	91,7	94,6	40%
0.15	97,5	126,7	40%
0.2	141,0	121,5	30%
0.25	111,8	88,8	30%
Total	442,0	431,6	
% Valor ótimo	30%	40%	

Relativamente à percentagem de soluções ótimas encontradas (tabela 4.5), para p_e a valer 0.1 e 0.15 e p=200 o valor é 40% enquanto que para os outros valores considerados foi de 30% . Em termos de distância (medido em minutos) para o valor ótimo, comparando os valores de p, a distância foi idêntica. Ao observar os valores de p_e, para $p = 100$ os melhores resultados foram obtidos por p_e igual a 0.1 e 0.15 e para $p = 200$ os melhores foram p_e igual a 0.1 e 0.25.

Após observar tais resultados, para p_e o valor escolhido é 0.15, apesar de 0.1 apresentar ligeiramente melhores resultados, mas com o objetivo de se poder levar um maior número de soluções de elite da geração anterior para a seguinte, o valor escolhido será 0.15. Relativamente ao parâmetro p continua a haver uma indecisão sobre o valor a escolher e para tal será realizada uma nova calibração, sem semente fixa, realizando 30 testes para cada p, considerando os valores 100, 125, 150, 175 e 200, para a instância 7 de Mankowska, Meisel & Bierwirth (2014), usando $p_e = 0.15$, p_m=0.3 e ρ_e=0.8.

Tabela 4.6: Calibração do parâmetro *p*

P	Média de Iterações	Média de Tempo	% de Valor ótimo	Tempo/Iteração	Distância ao Valor Ótimo
100	18,2	5,7	10,0%	0,3	183,3
125	14,9	6,7	6,7%	0,5	214,9
150	13,3	7,8	16,7%	0,6	131,8
175	15,0	9,4	20,0%	0,6	103,1
200	14,5	10,6	20,0%	0,7	89,0

Após a realização dos 30 testes para cada valor *p*, verifica-se na tabela 4.6 que em termos do número de iterações, à excepção de $p = 100$, o valor é idêntico. Em termos de média de tempo de computação e tempo por iteração, quanto maior a dimensão da população, maior esse tempo. Relativamente à percentagem de soluções ótimas encontradas, os valores 175 e 200 são os que apresentaram melhores resultados, registando 20%. Apesar disto, a soma da distância das soluções ao valor ótimo apresenta ligeiramente melhores resultados para p=200 do que para p=175.

Posto isto, a escolha do valor de *p* recai sobre uma dimensão de 175 ou de 200 cromossomas, pois ambos apresentam resultados muito idênticos. Com o intuito de poupar no tempo de computação, na parte de obtenção de resultados a dimensão da população será de 175 indivíduos.

4.2 Instâncias

No capítulo passado, foi possível verificar que o algoritmo conseguia chegar à solução ótima, previamente conhecida, numa quantidade razoável de tempo, para instâncias de 10 clientes. Como tal, seria benéfico poder testar o algoritmo para um exemplo mais complexo e sem conhecer a solução ótima.

Esta secção visa apresentar as instâncias usadas para o teste, as quais são o primeiro exemplo de 75 clientes de Mankowska, Meisel & Bierwirth (2014), adaptadas ao modelo.

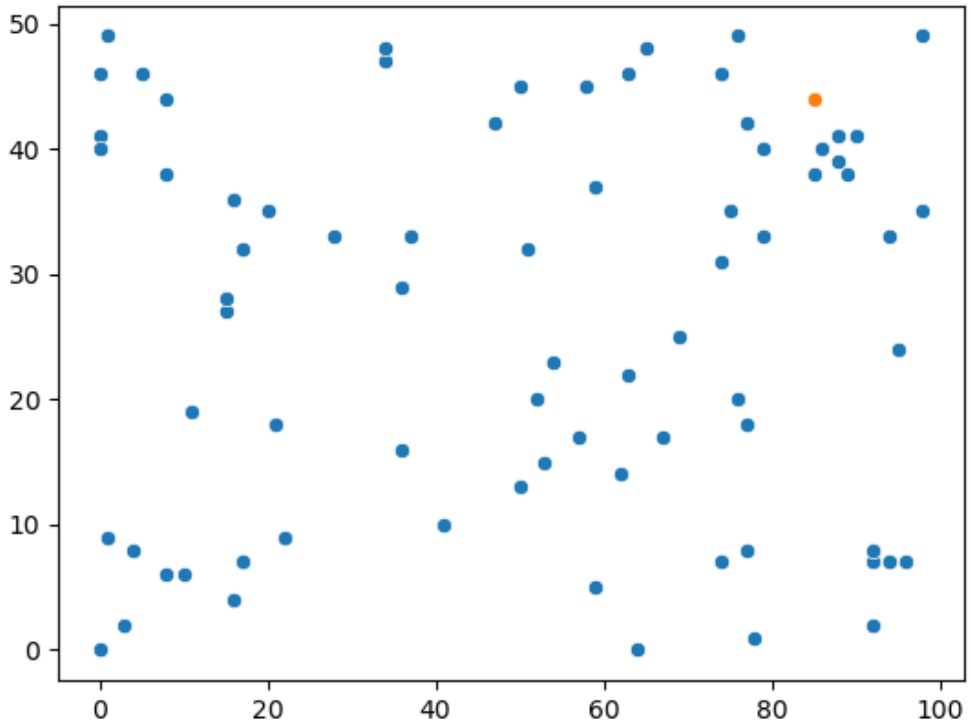
As instâncias são compostas por 75 clientes, mais o depósito contabilizado duas vezes devido a ser o início e término da rota (cliente 1 e cliente 77 são os depósitos), 2 serviços, serviços realizados apenas por um *caregiver* e serviços realizados por dois *caregivers* (1 equipa de dois ou 2 equipas de 1), e 15 *caregivers*. Independentemente do serviço que seja realizado, a duração de cada é sempre de 14 minutos. Dos clientes 54 até ao 76 é necessário dois *caregivers* para a realização do serviço. Como se tem 15 *caregivers*, existe 8 combinações diferentes de equipas as quais serão apresentadas na tabela 4.7.

Tabela 4.7: Combinação de equipas com 15 *caregivers*

Número de equipas com um *caregiver*	**Número de equipas com dois *caregivers***
15	0
13	1
11	2
9	3
7	4
5	5
3	6
1	7

Na figura 4.1 pode-se observar as coordenadas dos clientes. Destacado está um ponto, o qual é o depósito, contabilizado duas vezes.

Figura 4.1: Coordenadas dos clientes



4.3 Resultados

Os resultados foram obtidos com um processador Intel(R) Core(TM) i5-10600 CPU @ 3.30GHz e com uma RAM DE 16GB. O sistema operativo usado foi o *Windows 10* usando o compilador *PyCharm* para criação dos códigos.

Para a obtenção dos resultados, para cada combinação de equipas de *caregivers* foi obtido um total de 20 resultados, em que para cada teste com o algoritmo retorna a melhor solução após 1000 iterações ou se durante 20 iterações a melhor solução não mudar. Os parâmetros do algoritmo foram definidos na secção 4.1, sendo $p = 175$, $p_e = 0.15$, $p_m = 0.3$ e $\rho_e = 0.8$. Os melhores resultados para cada combinação de equipas estão apresentados na tabela 4.8.

Tabela 4.8: Melhores resultados obtidos para cada combinação de equipas

Combinação de equipas	**Valor, em minutos**	**Iterações**	**Tempo, em segundos**
15 equipas de um	2072.6	335	3502
13 equipas de um e 1 equipa de dois	1882.9	218	2056
11 equipas de um e 2 equipas de dois	1738.7	295	2357
9 equipas de um e 3 equipas de dois	1804.4	291	2044
7 equipas de um e 4 equipas de dois	1754.6	352	2107
5 equipas de um e 5 equipas de dois	1819.2	167	903
3 equipas de um e 6 equipas de dois	1896.4	253	1116
1 equipa de um e 7 equipas de dois	2157.9	225	965

Após observar a tabela 4.8, podemos concluir que a quantidade de equipas vai influenciar bastante os resultados e o tempo de computação, o qual é maior para um maior número de equipas e na qualidade dos resultados, pois com um maior número de equipas de 2 *caregivers* iremos ter uma maior penalização quando estas mesmas equipas realizam serviços de apenas um *caregiver*. O melhor resultado foi obtido quando havia 11 equipas de um *caregiver* e 2 equipas de dois *caregivers*. Na tabela 4.9 pode-se observar a distância percorrida, a penalização, duração do serviço e chegada ao depósito de cada equipa.

Tabela 4.9: Dados da melhor solução obtida, em minutos

Equipas	**Distância**	**Penalização**	**Duração do serviço**	**Chegada ao depósito**
Equipa de 2	272.9	42	466.6	629.4
Equipa de 2	167.8	56	317.6	476.5
Equipa de 1	178.2	0	475.2	578.1
Equipa de 1	58.1	0	159.5	514.0
Equipa de 1	20.5	0	48.5	452.2
Equipa de 1	217.8	0	476.1	510.2
Equipa de 1	156.0	0	465.3	578.1
Equipa de 1	113.5	0	446.3	527.4
Equipa de 1	105.8	0	360.3	504.8
Equipa de 1	47.5	0	189.4	359.6
Equipa de 1	49.7	0	241.6	483.8
Equipa de 1	203.5	0	472.3	541.0
Equipa de 1	49.4	0	128.8	359.6

Observando a tabela 4.9, pode-se verificar que não é considerado o equilíbrio no tempo de trabalho, o que provoca uma grande discrepância nos tempos de trabalho entre equipas. Por exemplo a terceira equipa de um *caregiver* tem uma duração de serviço

de 48.5 minutos (com 20.5 minutos em viagens), enquanto que a quarta equipa de um *caregiver* tem uma duração de serviço de 476.1 minutos (a percorrer uma distância de 217.8 minutos). Verificando a coluna da penalização, e relembrando que cada serviço dura 14 minutos, concluímos que a primeira equipa de dois *caregivers* realizou 3 serviços em que apenas era necessário um *caregiver* e que a segunda equipa de dois *caregivers* realizou 4 serviços em que apenas era necessário um *caregiver*. Na tabela 4.10 está representado a ordem das visitas de cada equipa.

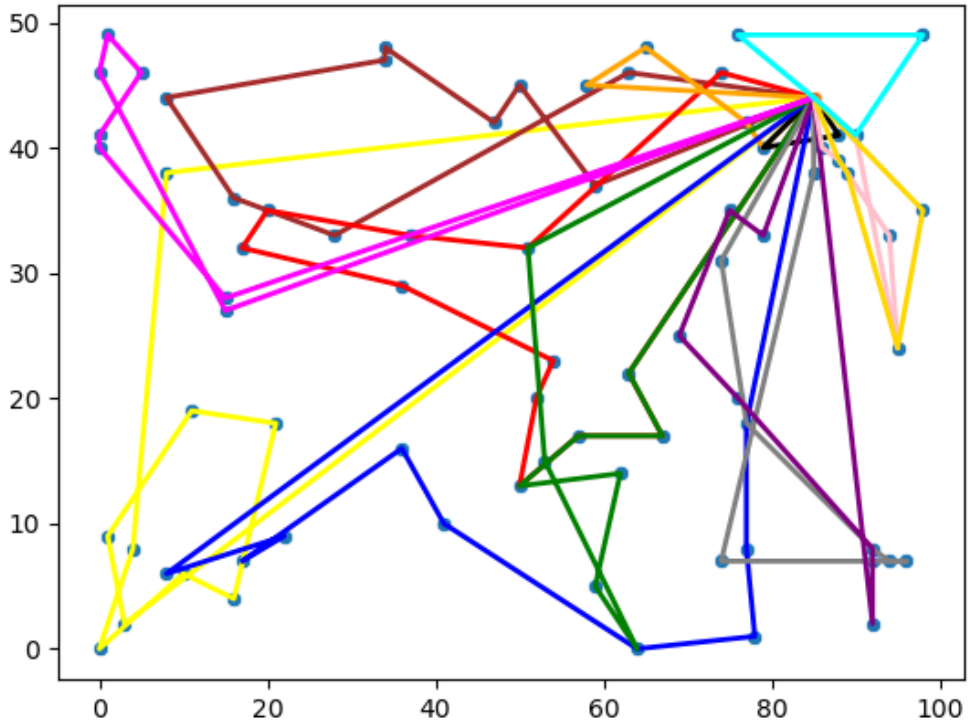
Tabela 4.10: Ordem das visitas de cada equipa

Equipas	**Ordem das visitas**
Equipa de 2	[1, 55, 49, 72, 64, 70, 12, 76, 30, 54, 77]
Equipa de 2	[1, 18, 66, 63, 61, 4, 65, 2, 60, 36, 77]
Equipa de 1	[1, 51, **57**, 15, 24, 46, 28, 3, 26, **67**, **58**, **56**, **73**, 77]
Equipa de 1	[1, **74**, 23, 35, 29, 77]
Equipa de 1	[1, **74**, 41, 77]
Equipa de 1	[1, 34, 38, 7, 50, 42, **69**, 5, 22, **59**, 77]
Equipa de 1	[1, **57**, 48, **69**, 45, 8, **67**, **58**, **56**, **73**, 77]
Equipa de 1	[1, 43, 13, 9, 10, 52, **75**, **59**, 47, 77]
Equipa de 1	[1, 44, 20, 6, 27, **75**, 39, 77]
Equipa de 1	[1, 32, **71**, 53, **68**, **62**, 77]
Equipa de 1	[1, **71**, 19, **68**, 33, 77]
Equipa de 1	[1, 21, 25, 40, 16, 17, 31, 37, 77]
Equipa de 1	[1, 11, 14, **62**, 77]

Após analisar a tabela 4.10, pode-se confirmar que a primeira equipa de dois *caregivers* realizou 3 serviços onde apenas era necessário um *caregiver* aos clientes 12, 30 e 49, enquanto que a segunda equipa de dois *caregivers* realizou 4 serviços desse tipo aos clientes 2, 4, 18 e 36. Em relação à sincronização, a primeira equipa de um *caregiver* juntou-se com outras equipas de um *caregiver* para realizar um serviço de dois *caregivers* aos clientes 56, 57, 58, 67 e 73, a segunda equipa um ao cliente 74, a terceira equipa de um ao cliente 74, a quarta equipa de um aos clientes 59 e 69, a quinta equipa de um aos clientes 56, 57, 58, 67, 69 e 73, a sexta equipa de um aos clientes 59 e 75, a sétima equipa de um ao cliente 75, a oitava equipa de um aos clientes 62, 68 e 71, a nova equipa de um aos clientes 68 e 71, a décima equipa de um não realizou serviços com sincronização e décima primeira equipa de um ao cliente 62.

Na figura 4.2 podemos observar o caminho percorrido por cada equipa. A rota da primeira equipa de dois *caregivers* está representada a amarelo, a da segunda equipa de dois *caregivers* a castanho, a da primeira equipa de um *caregiver* a vermelho, a da segunda equipa de um *caregiver* a laranja, a da terceira equipa de um *caregiver* a preto, a da quarta equipa de um *caregiver* a azul, a da quinta equipa de um *caregiver* a verde, a da sexta equipa de um *caregiver* a cinzento, a da sétima equipa de um *caregiver* a roxo, a da oitava equipa de um *caregiver* a rosa, a da nona equipa de um *caregiver* a dourado, a da décima equipa de um *caregiver* a magenta e a da décima primeira equipa de um *caregiver* a ciano.

Figura 4.2: Rotas atribuídas



5

Conclusão e Trabalho Futuro

Uma grande tendência do mundo atual é o envelhecimento da população. A origem deste acontecimento tem por base a progressiva melhoria da medicina, o que provoca um aumenta na esperança média de vida, e de uma diminuição dos níveis de fertilidade. À medida que a idade avança, há uma tendência a que o número de doenças e a probabilidade de as contrair aumente também. Algumas delas provocam uma grande diminuição da autonomia das pessoas, e mesmo as que não provocam, com o aumentar da idade torna-se cada vez mais difícil realizar as tarefas básicas, como moverem-se, lavarem-se ou alimentarem-se. Um grande provocador da perda de autonomia é a demência, que provoca perda de memória e dificuldade na orientação. A solução para este problema reside frequentemente na contratualização de serviços a instituições que prestam cuidados de saúde ao domicílio. Muitas destas instituições são instituições sem fins lucrativos, financiadas pela segurança social e por doações. Para as instituições poderem realizar os seus serviços, têm de previamente definir os itinerários e atribuir aos *caregivers* os clientes, o que pode não estar otimizado, provocando um maior gasto de custos.

O objetivo desta dissertação de mestrado é desenvolver um algoritmo que ajudasse na atribuição dos *caregivers* e a definir a ordem das rotas. O algoritmo usado foi o *Biased Random Key Genetic Algorithm*, para um horizonte temporal de um dia. A representação dos cromossomas do problema é baseada em *random keys*. O problema consiste num *Vehicle Routing Problem* com janelas temporais e restrições adicionais. Os *caregivers* não necessitam de determinadas *skills*, pois é considerado que todos são capazes de realizar os serviços disponíveis. Os serviços são divididos dois tipos, os que necessitam de um *caregiver* e os que necessitam dois *caregivers* para a sua realização. Serviços que necessitam de dois *caregivers* podem ser feitos através da sincronização, o que significa que 2 equipas de um *caregiver* podem juntar-se para realizar o serviço. Serviços de um *caregiver* também poderão ser realizados por equipas de dois *caregivers*, embora quando ocorra exista uma penalização na função objetivo do tempo de serviço (em minutos). A função objetivo é a minimização da distância percorrida (medida em minutos) e da penalização. Este algoritmo requer certos parâmetros, os quais foram calibrados usando as instâncias de 10 clientes de Mankowska, Meisel & Bierwirth (2014).

Após a calibração dos parâmetros, o algoritmo BRKGA foi usado para obter uma solução para o primeiro exemplo de 75 instâncias de Mankowska, Meisel & Bierwirth (2014), onde o melhor resultado foi obtido quando se tinha 11 equipas de um *caregiver* e 2 equipas de dois *caregivers* com o valor da função objetivo de **1738.7 minutos**.

Em trabalhos futuros, espera-se que a meta-heurística seja testada noutros conjuntos de dados de exemplo da literatura do HHC. Além disso, seria benéfico a proposta de descodificadores mais eficientes que sejam capazes de explorar todo o espaço de solução do problema, podendo resolver para um horizonte temporal maior do que um dia, juntamente com a adição da continuidade de cuidados diária (para casos de clientes que necessitem de mais do que uma visita por dia) e semanal. Como foi visto na calibração de parâmetros, nem sempre se atingiu o valor ótimo do problema em questão, algo muito influenciado pela população inicial. Posto isto, algo a melhorar seria o garantir de um bom ponto de partida para o problema.

Bibliografia

[1] A. R. Aguiar. "Planning home health care services – a routing and scheduling problem Biomedical Engineering". Em: *Instituto Superior Técnico* June (2017).

[2] C. Akjiratikarl, P. Yenradee e P. R. Drake. "PSO-based algorithm for home care worker scheduling in the UK". Em: *Computers and Industrial Engineering* 53.4 (2007), pp. 559–583. ISSN: 03608352. DOI: 10.1016/j.cie.2007.06.002.

[3] H. Allaoua et al. "A matheuristic approach for solving a home health care problem". Em: *Electronic Notes in Discrete Mathematics* 41 (2013), pp. 471–478. ISSN: 15710653. DOI: 10.1016/j.endm.2013.05.127.

[4] E. Angelelli e M. G. Speranza. "The periodic vehicle routing problem with intermediate facilities". Em: *European Journal of Operational Research* 137.2 (2002), pp. 233–247. ISSN: 03772217. DOI: 10.1016/S0377-2217(01)00206-5.

[5] J. F. Bard, Y. Shao e A. I. Jarrah. "A sequential GRASP for the therapist routing and scheduling problem". Em: *Journal of Scheduling* 17.2 (2014), pp. 109–133. ISSN: 10946136. DOI: 10.1007/s10951-013-0345-x.

[6] *Bean, J.C.: Genetic algorithms and random keys for sequencing and optimization. ORSA J. Comput. 6, 154–160 (1994).*

[7] T. Bektas. "The multiple traveling salesman problem: An overview of formulations and solution procedures". Em: *Omega* 34.3 (2006), pp. 209–219. ISSN: 03050483. DOI: 10.1016/j.omega.2004.10.004.

[8] S. Bertels e T. Fahle. "A hybrid setup for a hybrid scenario: Combining heuristics for the home health care problem". Em: *Computers and Operations Research* 33.10 (2006), pp. 2866–2890. ISSN: 03050548. DOI: 10.1016/j.cor.2005.01.015.

[9] D. Bloom, D. Canning e G. Fink. "Implications of Population Aging for Economic Growth". Em: *National Bureau of Economic Research* (2011). ISSN: 0898-2937. DOI: 10.3386/w16705.

[10] O. Bräysy e M. Gendreau. "Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms". Em: *Transportation Science* 39.1 (2005), pp. 104–118. DOI: 10.1287/trsc.1030.0056.

[11] O. Bräysy e M. Gendreau. "Vehicle Routing Problem with Time Windows, Part II: Metaheuristics". Em: *Transportation Science* 39.1 (2005), pp. 119–139. DOI: 10.1287/trsc.1030.0057.

[12] O. Bräysy et al. "An optimization approach for communal home meal delivery service: A case study". Em: *Journal of Computational and Applied Mathematics* 232.1 (2009), pp. 46–53. ISSN: 03770427. DOI: 10.1016/j.cam.2008.10.038.

[13] D. Bredström e M. Rönnqvist. "Combined vehicle routing and scheduling with temporal precedence and synchronization constraints". Em: *European Journal of Operational Research* 191.1 (2008), pp. 19–31. ISSN: 03772217. DOI: 10.1016/j.ejor.2007.07.033.

[14] N. Bricon-Souf et al. "A distributed coordination platform for home care: Analysis, framework and prototype". Em: *International Journal of Medical Informatics* 74.10 (2005), pp. 809–825. ISSN: 13865056. DOI: 10.1016/j.ijmedinf.2005.03.020.

[15] P. Cappanera e M. G. Scutellà. "Joint Assignment, Scheduling, and Routing Models to Home Care Optimization: A Pattern-Based Approach". Em: *Transportation Science* 49.4 (2015), pp. 830–852. DOI: 10.1287/trsc.2014.0548.

[16] M. Cissé et al. "OR problems related to Home Health Care: A review of relevant routing and scheduling problems". Em: *Operations Research for Health Care* 13-14 (2017), pp. 1–22. ISSN: 22116923. DOI: 10.1016/j.orhc.2017.06.001.

[17] V. R. Comondore et al. "Quality of care in for-profit and not-for-profit nursing homes: Systematic review and meta-analysis". Em: *BMJ (Online)* 339.7717 (2009), pp. 381–384. ISSN: 17561833. DOI: 10.1136/bmj.b2732.

[18] G. B. Dantzig e J. H. Ramser. "The Truck Dispatching Problem". Em: *Management Science* 6.1 (1959), pp. 80–91. DOI: 10.1287/mnsc.6.1.80.

[19] C. Fikar e P. Hirsch. "Home health care routing and scheduling: A review". Em: *Computers and Operations Research* 77 (2017), pp. 86–95. ISSN: 03050548. DOI: 10.1016/j.cor.2016.07.019.

[20] M. Fischetti e M. Fischetti. *Martina Fischetti and Matteo Fischetti 5*. 2018. ISBN: 9783319071244.

[21] K. Fleszar, I. H. Osman e K. S. Hindi. "A variable neighbourhood search algorithm for the open vehicle routing problem". Em: *European Journal of Operational Research* 195.3 (2009), pp. 803–809. ISSN: 03772217. DOI: 10.1016/j.ejor.2007.06.064.

[22] P. M. Francis, K. R. Smilowitz e M. Tzur. "The Period Vehicle Routing Problem and its Extensions". Em: *The Vehicle Routing Problem: Latest Advances and New Challenges*. Ed. por B. Golden, S. Raghavan e E. Wasil. Boston, MA: Springer US, 2008, pp. 73–102. ISBN: 978-0-387-77778-8. DOI: 10.1007/978-0-387-77778-8_4.

[23] N. Frontiers. *GENETIC Edited by Lance Chambers*. Vol. II. ISBN: 1558602631.

[24] N. Genet et al. "Home Care across Europe, Case studies". Em: *European Observatory on Health Systems and Policies, WHO* (2013), pp. 1–330.

[25] M. I. Gomes e T. R. P. Ramos. "Modelling and (re-)planning periodic home social care services with loyalty and non-loyalty features". Em: *European Journal of Operational Research* 277.1 (2019), pp. 284–299. ISSN: 03772217. DOI: 10.1016/j.ejor.2019.01.061.

[26] J. F. Gonçalves e M. G. Resende. "Biased random-key genetic algorithms for combinatorial optimization". Em: *Journal of Heuristics* 17.5 (2011), pp. 487–525. ISSN: 13811231. DOI: 10.1007/s10732-010-9143-1.

[27] J. H. HOLLAND. "Adaption in Natural and Artificial Systems". Em: *An Introductory Analysis with Application to Biology, Control and Artificial Intelligence* (1975).

[28] A. F. Kummer, L. S. Buriol e O. C. De Araujo. "A biased random key genetic algorithm applied to the VRPTW with skill requirements and synchronization constraints". Em: *GECCO 2020 - Proceedings of the 2020 Genetic and Evolutionary Computation Conference* 5 (2020), pp. 717–724. DOI: 10.1145/3377930.3390209.

[29] E. Lanzarone e A. Matta. "Robust nurse-to-patient assignment in home care services to minimize overtimes under continuity of care". Em: *Operations Research for Health Care* 3.2 (2014), pp. 48–58. ISSN: 22116923. DOI: 10.1016/j.orhc.2014.01.003.

[30] B. Leff e J. R. Burton. "Acute medical care in the home". Em: *Journal of the American Geriatrics Society* 44.5 (1996), pp. 603–605. ISSN: 15325415. DOI: 10.1111/j.1532-5415.1996.tb01452.x.

[31] R. Liu, B. Yuan e Z. Jiang. "Mathematical model and exact algorithm for the home care worker scheduling and routing problem with lunch break requirements". Em: *International Journal of Production Research* 55.2 (2017), pp. 558–575. DOI: 10.1080/00207543.2016.1213917.

[32] R. Liu et al. "Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care". Em: *European Journal of Operational Research* 230.3 (2013), pp. 475–486. ISSN: 03772217. DOI: 10.1016/j.ejor.2013.04.044.

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

[34] D. S. Mankowska, F. Meisel e C. Bierwirth. "The home health care routing and scheduling problem with interdependent services". Em: *Health Care Management Science* 17.1 (2014), pp. 15–30. ISSN: 13869620. DOI: 10.1007/s10729-013-9243-1.

[35] V. Pillac et al. "A review of dynamic vehicle routing problems". Em: *European Journal of Operational Research* 225.1 (2013), pp. 1–11. ISSN: 03772217. DOI: 10.1016/j.ejor.2012.08.015.

[36] G. R. Raidl, J. Puchinger e C. Blum. *Metaheuristic Hybrids*. 2010, pp. 469–496. ISBN: 9781441916631. DOI: 10.1007/978-1-4419-1665-5_16.

[37] M. S. Rasmussen et al. "The Home Care Crew Scheduling Problem: Preference-based visit clustering and temporal dependencies". Em: *European Journal of Operational Research* 219.3 (2012), pp. 598–610. ISSN: 03772217. DOI: 10.1016/j.ejor.2011.10.048.

[38] C. Reeves e J. E. Rowe. *Genetic algorithms: principles and perspectives: a guide to GA theory*. Vol. 20. Springer Science & Business Media, 2002.

[39] K. Sörensen, M. Sevaux e F. Glover. "A history of metaheuristics". Em: *Handbook of Heuristics* 2-2 (2018), pp. 791–808. DOI: 10.1007/978-3-319-07124-4_4. arXiv: 1704.00853.

[40] A. Trautsamwieser e P. Hirsch. "A Branch-Price-and-Cut approach for solving the medium-term home health care planning problem". Em: *Networks* 64.3 (2014), pp. 143–159. DOI: https://doi.org/10.1002/net.21566.

[41] J. Wirnitzer et al. "Patient-based nurse rostering in home care". Em: *Operations Research for Health Care* 8 (2016), pp. 91–102. ISSN: 22116923. DOI: 10.1016/j.orhc.2015.08.005.

[42] M. Yordanova et al. "Enhancement of xylanase production by sol-gel immobilization of Aspergillus awamori K-1". Em: *Bulgarian Journal of Agricultural Science* 19.SUPPL. 2 (2013), pp. 117–119. ISSN: 13100351.