DM

Transporte Flexível
Modelação, analise e simulação

DISSERTAÇÃO DE MESTRADO

João Diogo Nascimento Silva
MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

fevereiro | 2020

Transporte Flexível
Modelação, analise e simulação

DISSERTAÇÃO DE MESTRADO

João Diogo Nascimento Silva
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTAÇÃO
Leonel Domingos Telo Nóbrega

Índice de conteúdo

Resumo

Neste trabalho é discutido e apresentado todo o processo de modelação e implementação de uma solução que permita resolver tanto a vertente estática, como a dinâmica, do problema de *routing* de transporte flexível, muitas vezes chamado de DARP (*Dial-a-Ride Problem*). De forma geral, este problema requer que um conjunto de clientes efetuem pedidos de serviço, no qual é indicado o ponto origem e destino, assim como os tempos pretendidos para a prestação do serviço. O objetivo é gerar soluções possíveis e viáveis (percursos) para os veículos disponíveis, de forma a minimizar os custos totais e tendo em conta o conjunto de pedidos (e informações relativas aos mesmos) e o conjunto de restrições impostas ao problema. A vertente estática caracteriza-se pelo facto de toda a procura ser conhecida inicialmente, sendo então necessário apenas gerar uma única solução (conjunto de percursos) com custos mínimos. A outra vertente (dinâmica) distingue-se pelo facto de a procura se ir revelando dinamicamente ao longo da execução do serviço, sendo então necessário ir efetuando reajustes em tempo real aos percursos de cada veículo tendo em conta o surgimento de novos pedidos. Este tipo de problema permite dar um papel ativo ao cliente, permitindo melhorar a qualidade do serviço, uma vez que são tidas em conta as necessidades do mesmo. Um outro aspeto vantajoso deste tipo de sistemas, é permitir aos operadores de transporte conhecer a procura atual, levando a que possa ser otimizada a sua frota de veículos e conseguir dar resposta às necessidades identificadas.

De forma a validar, testar e analisar a solução proposta numa situação mais aproximada da real e tendo em conta diferentes configurações, foi ainda modelada e desenvolvida uma simulação de eventos discretos (DES), que reflete o funcionamento de um sistema de transporte flexível, estando neste documento descrito todos os aspetos fundamentais para a modelação e implementação da mesma. Para além disto, ainda são apresentados um conjunto de testes efetuados a alguns mecanismos (algoritmos) de procura de soluções que permitem otimizar os percursos obtidos para o problema formulado para uma abordagem estática e tendo ainda em conta diferentes configurações do mesmo. Por fim, são ainda apresentados os resultados dos testes efetuados em simulação para a abordagem dinâmica do problema de *routing* formulado de forma a compreender a influência que o aumento da procura tem na solução proposta.

Palavras-Chave

Transporte Flexível, DARP, Problemas de *Routing* de Veículos, Simulação e Otimização

Abstract

With this work, we discuss and present the whole modelling and implementation process of a solution for the static and dynamic Dial-a-Ride Problem (DARP). Overall, this problem requires a set of customers to perform set of transportation requests, where in each request, it is provided the customer desired pickup and drop-off locations, as well as the desired times to be fulfilled for the service being requested. The objective is to generate a set of viable routes, considering a set of available vehicles and a set of transportation requests, as well as some additional restrictions imposed on the problem, while minimizing the total cost of the solution. In the static DARP, all the requests are known initially, meaning that we only need to generate the solution (set of routes) once. While on the dynamic approach of the DARP, the requests are revealed dynamically throughout the execution of the transportation service, meaning that there is the need to carry out adjustments on each vehicle route in real time, with the appearance of new requests. This routing problem gives an active role to the customer by considering its needs, which, in way, improves the quality of the service provided. Another advantage of this type of system is the fact that transportation providers, are able to know the current demand and needs related to it, thus, making it possible for them to optimize their vehicle fleet in order to respond to it.

In order to validate, analyze and evaluate the proposed solution for the DARP using different configurations and in a context closer to the real world, we also propose in this document, all the modeling and development process of a Discrete Event Simulation that reflects in a way a flexible transportation system operation. Additionally, we also present the results of a set of tests done using different configurations, to some mechanisms (search algorithms) for finding and optimizing solutions, for the static approach of the modeled DARP. Finally, we also present the results for the tests performed using simulation for the dynamic DARP, in order to understand the influence that the increase in the demand has in the proposed DARP solution.

Keywords

Flexible Transportation, Dial-a-Ride Problem, Vehicle Routing Problem, Simulation and Optimization

Lista de abreviaturas

DARP	Dial-a-Ride Problem
VRP	Vehicle Routing Problem
OR-Tools	Operations Research Tools
NP	Non-deterministic Polynomial-time
DES	Discrete-Event Simulation
CS	Continuous Simulation

Agradecimentos

Primeiro de tudo, gostaria de agradecer ao Professor Dr. Leonel Domingos Telo Nóbrega, por todo o apoio, atenção e orientação providenciada ao longo do desenvolvimento deste trabalho.

Também gostaria de agradecer aos meus pais, por todo o apoio e também suporte económico providenciado durante toda a minha instrução académica.

Por fim, gostaria de agradecer à empresa Horários do Funchal, pelo fornecimento de alguns dados reais que foram utilizados nesta dissertação.

Índice de figuras

Índice de tabelas

1 Introdução

1.1 Motivação

Atualmente, a maior parte da mobilidade dos cidadãos, dentro das zonas urbanas, é efetuada através de dois principais tipos de transporte: transportes públicos (e.g. autocarros ou metro), e/ou transporte individual (e.g. carro particular). Ambos estes meios de transporte apresentam algumas vantagens e desvantagens de um em relação ao outro para as pessoas que os utilizam. Na Região Autónoma da Madeira (RAM) grande parte da população, devido a vários fatores, utiliza preferencialmente o transporte individual em detrimento do transporte público. Alguns destes fatores, que contribuem para esta preferência, incluem: o contexto social da população, a curta distância entre as diferentes localidades, o facto de as pessoas terem de utilizar diferentes operadoras de serviço de transporte público (dependendo da localidade de origem ou de destino), entre outros.

A adoção da utilização de transportes públicos por grande parte da população é de extrema importância do ponto de vista ambiental, como forma de permitir reduzir as emissões de gases poluentes, assim como reduzir o congestionamento urbano e permitindo também reduzir a probabilidade de acidentes rodoviários, uma vez que menos veículos circularão na estrada. Para além disto, o transporte público tem vantagens a nível económico, na qual a utilização do mesmo é geralmente mais barata do que a utilização do transporte individual, não existindo então custos de manutenção nem de abastecimento para o cliente, só tendo este custos de deslocamento através de passes ou bilhetes. Adicionalmente, o transporte público também é de forma geral, considerado mais seguro do que o transporte individual, providenciando assim uma certa garantia de segurança aos seus utilizadores.

No entanto, o transporte público também apresenta algumas desvantagens, tornando assim o transporte individual por vezes preferível. Um dos seus grandes problemas é a inflexibilidade de percursos e horários, uma vez que os transportes públicos têm percursos e horários já previamente definidos, aos quais as pessoas ficam sujeitas. Este problema leva a que as pessoas demorem mais tempo a chegar aos seus destinos e também tenham que esperar pelos transportes, aumentando assim o tempo "perdido" durante o dia para se deslocarem. Um outro problema, também relacionado com este, advém do facto de por vezes certos utentes terem que apanhar um conjunto de diferentes autocarros (com diferentes percursos) de forma a conseguirem chegar ao seu local destino, ficando assim sujeitos aos já referidos horários e percursos inflexíveis, aumentando assim ainda mais o tempo perdido.

Existem também problemas relacionados com a procura, como a superlotação ou falta de procura sendo estes um grande problema tanto para os utentes como para as operadoras de transporte público. Por um lado, a superlotação, que geralmente ocorre nas horas de ponta, pode provocar desconforto para os utentes, por outro lado a falta de procura, que ocorre geralmente em horas baixas como períodos noturnos, geralmente têm custos operacionais adicionais para os operadores de transporte público, uma vez que a utilização do meio de transporte não está a ser maximizada ou aproveitada.

O conceito de transporte flexível surge então como uma alternativa eficaz ao transporte público tradicional, visando resolver alguns dos seus problemas. Esta alternativa é caracterizada pela existência de percursos e horários flexíveis, gerados e otimizados dinamicamente, sendo

adaptados às necessidades reais dos clientes. Este aspeto flexível permite então reduzir os problemas relacionados com o tempo "perdido" pelos clientes em deslocamento, reduzindo assim os tempos de espera para a realização do serviço e o tempo que os utentes "despendem" dentro dos meios de transporte. Adicionalmente, esta flexibilidade também é uma mais valia para os utentes que necessitavam de apanhar um conjunto de diferentes autocarros (com diferentes carreiras) para chegar ao seu destino, permitindo, eventualmente, que todo o trajeto seja realizado por um único autocarro. Como podemos ver, este tipo de estratégia confere um papel ativo ao cliente, que ao manifestar a sua necessidade de serviço de transporte, permite-lhe também de certa forma influenciar a forma como o serviço será prestado e programado.

Uma vez que a procura é conhecida previamente, este tipo de sistema também permite combater problemas relacionados com a mesma (como a superlotação ou períodos de pouca procura), permitindo assim aos fornecedores de mobilidade como serviço, otimizarem a sua frota de veículos (em serviço) de forma a dar resposta à procura atual e ao tipo de cliente, utilizando assim apenas os recursos necessários. Isto leva a que os fornecedores de serviço consigam ter uma redução nos custos operacionais, sendo assim também uma mais valia para os mesmos do ponto de vista económico.

Um outro problema relacionado com os sistemas de transporte público tradicionais, surge no transporte de pessoas com dificuldades motoras ou idosos, que por vezes necessitam de veículos com equipamentos especiais, como por exemplo suporte para cadeira de rodas. Para este tipo de serviço, é necessário realizar a marcação com alguma antecedência e em geral utilizando a rede telefónica. Um sistema de transporte flexível seria também uma mais valia para este tipo de cliente, uma vez que a procura e as necessidades de serviço a ser prestado é conhecida inicialmente, permitindo assim, simplificar o processo de programação de serviço e geração de percursos, automatizando-o e utilizando os recursos apropriados para o tipo de cliente em questão.

Como podemos constatar o transporte flexível permite resolver alguns dos problemas inerentes ao transporte público, tendo este, vantagens tanto para os utentes como também para os fornecedores de serviço, e, de certa forma, também permitindo haver uma maior aproximação entre o transporte público e o transporte individual.

No entanto, de forma a conseguir criar um sistema de gestão e operação de serviços de transporte flexível, é necessária a criação e desenvolvimento de estratégias tecnológicas que permitam dar resposta às necessidades deste mesmo sistema e em tempo oportuno. O sistema em questão terá que permitir gerir mais eficientemente e de certa forma, automatizar o processo de geração e programação do serviço, e ainda permitir lidar com todos os intervenientes do serviço, isto é, o condutor, o operador de serviço e o cliente. Contudo, é importante notar que este documento irá se focar mais nas estratégias de geração de percursos, isto é, as estratégias para a automatização e geração de percursos viáveis e em curtos intervalos de tempo, tendo em conta a procura atualmente conhecida. Acreditamos que com a criação e implementação de um sistema deste tipo na RAM, seria então possível fazer com que mais pessoas começassem a utilizar os serviços de transporte público (flexível), devido ao facto que este tipo de sistema permite ir buscar o melhor de dois mundos (o transporte público e o individual), permitindo aumentar a satisfação do cliente, e consequentemente favorecendo

então a adoção do transporte público em relação ao transporte individual, permitindo assim combater a atual tendência.

1.2 Contexto

Este trabalho tem por base o projeto MiNiVAN, o qual tem como objetivo o desenvolvimento de um conjunto de serviços de transporte flexível na cidade do Funchal (Região Autónoma da Madeira, Portugal), e advindo este da colaboração entre os Horários do Funchal, Altice Labs e a Universidade da Madeira. Este conjunto de serviços permitiria um melhor aproveitamento dos recursos de mobilidade existentes, diversificar a oferta de serviços de transporte público, permitindo então aumentar a satisfação dos utilizadores e consequentemente combater o problema já referido da preferência do transporte individual sobre o transporte público.

Os serviços de mobilidade flexível do projeto MiNiVAN abrangem cinco casos de uso principais:

- Transporte pontual de pessoas sem meios de transporte próprio e sem qualquer tipo de limitação física (e.g. populações dispersas, longe dos centros urbanos);
- Transporte de doentes para consultas regulares nos centros de saúde ou hospitais (e.g. doentes autónomos para consultas pré-agendadas);
- Transporte de crianças nos circuitos casa-escola-atividade-casa;
- Transporte durante períodos de baixa procura, como períodos noturnos ou fins-de-semana;
- Integração multioperador de transportes coletivos e individuais de passageiros concessionados.

1.3 Problema

O nome mais frequentemente dado na literatura atual, ao problema de geração e agendamento de percursos de veículos para transporte flexível é de Dial-a-Ride Problem (DARP), sendo este uma especialização do *Vehicle Routing Problem* (VRP) [1]. O grande aspeto diferenciador entre o DARP e outros problemas de geração e agendamento de percursos de veículos (VRPs), é que este tem em conta o aspeto humano, isto é a satisfação do cliente, estando este problema também muitas vezes associado ao transporte de doentes ou idosos. De forma geral, requer que um conjunto de clientes efetuem pedidos de serviço, cada um indicando o seu local de origem - onde será recolhido - e o seu local destino - para onde será transportado. Para além da localização, geralmente é necessário que cada cliente defina ainda algum tipo de restrição temporal, normalmente indicando a partir de que momento este estará no local de origem, e quando é que este pretende estar no local destino. Após serem sabidos os pedidos, o transporte será efetuado por um ou mais veículos, conforme a necessidade de cada um, transportando um ou mais clientes, fornecendo assim uma espécie de serviço partilhado. O problema de otimização consiste em gerar para um dado número de veículos, um conjunto de percursos viáveis e possíveis, com custos mínimos, de forma a acomodarem todos ou grande parte dos pedidos de serviço conhecidos, e tendo ainda em conta um conjunto de restrições

existentes. Note-se que o problema modelado do DARP estará definido em mais detalhe no capítulo 3.

É ainda importante notar que, este problema é também caracterizado por duas vertentes: a vertente estática e a vertente dinâmica. Sendo que na primeira vertente, toda a informação necessária para a construção do percurso é conhecida inicialmente. Relativamente à vertente dinâmica, toda a informação necessária para a construção dos percursos é revelada dinamicamente durante a execução, sendo assim necessário efetuar reajustes em tempo real (ou tempo quase real), aos percursos atuais de forma a satisfazer os pedidos de serviço atuais. Sendo que ambas estas vertentes apresentam necessidades a nível computacional (desempenho), de forma a conseguir resolver os problemas em questão, gerando soluções fiáveis e possíveis, e em tempo oportuno. No entanto, é notório que a vertente dinâmica apresentará maiores necessidades de desempenho, uma vez que esta necessita de uma geração continuada de novas soluções e em tempo real (ou quase real), conforme as novas informações de serviço vão sendo reveladas durante a execução, de forma a tentar satisfazer todos os pedidos de serviço e restrições inerentes ao problema.

Existem vários algoritmos/mecanismos na literatura atual com vista a encontrar uma ou mais soluções, viáveis e possíveis, para este tipo de problema, tendo em conta o tipo de vertente (estática ou dinâmica), as restrições e também os objetivos do mesmo. Podendo estes algoritmos advirem de diferentes tipos e cada um tendo diferentes características, podendo estes serem: algoritmos exatos, meta heurísticos, heurísticos, genéticos entre outros. Note-se que estes algoritmos serão apresentados e discutidos em mais detalhe nos próximos capítulos. A validação para cada um destes algoritmos em contexto real é de forma geral muito difícil e também dispendiosa, devido à necessidade da existência de diversos recursos (e.g. veículos, clientes e custos operacionais), levando então a que a simulação surja como uma abordagem preferível, pelo menos inicialmente.

A criação de um ambiente de simulação torna-se numa mais valia do ponto de vista de validação e avaliação do sistema modelado, permitindo testá-lo, tendo em conta diferentes aspetos estocásticos do problema. A simulação permite de certa forma “imitar” (simular) virtualmente o contexto real do problema, utilizando diferentes configurações, permitindo então tirar conclusões acerca do sistema modelado, e ainda inferir se existe algum algoritmo que se melhor aplique ao problema em questão tendo em conta diferentes critérios. Adicionalmente, é importante notar que este problema está presente num ambiente bastante dinâmico, onde eventos estocásticos podem ocorrer em qualquer momento, levando a que a criação de um ambiente de simulação que reflita a ocorrência dos mesmos, permita também melhor compreender os efeitos que os mesmos têm sobre o sistema modelado. Para além disto, a simulação é também uma mais valia do ponto de vista de avaliação dos serviços de transporte do projeto MiNiVAN, permitindo então simulá-los e consequentemente avaliá-los de forma a compreender a viabilidade dos mesmos e ainda perceber como é que cada serviço se comportaria numa situação aproximadamente real.

1.4 Contribuições

Com este trabalho é pretendida a formulação de uma solução para o problema de *routing* de transporte flexível, mais geralmente conhecido como *DARP*. Sendo que esta solução deverá permitir a construção de percursos (soluções) tendo em conta um conjunto distinto de

características e restrições (e.g. recursos, pedidos de serviço, intervalos de tempo) inerentes à mesma, e ainda a resolução de ambas as abordagens encontradas para este problema de *routing* (estática e dinâmica). Para além disto, serão efetuados testes a nível computacional de forma a tentar compreender que mecanismos (algoritmos) de procura de soluções permitem de certa forma melhor otimizar as soluções obtidas, tendo em conta um intervalo de tempo específico para a obtenção de estas soluções.

Por fim, de forma a validar, testar e analisar a solução proposta e aspetos inerentes à mesma, propomo-nos ainda à modelação e implementação de um simulador, que permita simular o contexto bastante dinâmico, no qual os sistemas de transporte flexível se enquadram. Com os testes em simulação é pretendido tentar compreender de que forma é que o aumento da procura influência a solução proposta para um contexto simulado mais aproximado do real.

1.5 Estrutura do trabalho

O segundo capítulo deste documento, "Estado de arte", providencia uma visão muito geral acerca do problema de *routing* de veículos para transporte flexível (*DARP*), indicando as principais características e restrições encontradas na literatura atual, como também possíveis soluções e mecanismos utilizados para resolução do mesmo. Para além disto é ainda providenciada uma breve descrição acerca das diferentes abordagens de simulação encontradas na literatura. Por fim, são ainda referidas algumas ferramentas existentes para a modelação e resolução de problemas de *routing* atualmente.

O terceiro capítulo, "Analise e desenho da solução", providencia algum raciocínio acerca do problema formulado para o *DARP*, a abordagem tomada como também alguns requisitos necessários para a construção da solução proposta.

O quarto capítulo, "Desenvolvimento", refere todos os aspetos relevantes para a modelação, construção e implementação da solução proposta para a resolução do problema de *routing* de transporte flexível formulado.

O quinto capítulo, "Resultados", descreve a forma como os testes foram efetuados, como também algum raciocínio acerca dos resultados obtidos para os mesmos. Para além disto é ainda, indicado algumas abordagens para melhorar as soluções obtidas tendo em conta diferentes aspetos.

Por fim, o sexto capítulo, "Conclusão e Trabalho futuro", irá apresentar as conclusões tiradas acerca do trabalho efetuado, como também alguns aspetos que seriam interessantes explorar no futuro, de certa forma a permitir melhorar o trabalho aqui efetuado.

2 Estado de arte

A primeira secção deste capítulo (2.1), aborda as diferentes características dos diferentes problemas de *routing* de veículos de transporte flexível (DARP), encontradas na literatura atual.

A secção seguinte (2.2), refere algumas das estratégias encontrada na literatura atual, utilizadas para resolver diferentes problemas de *routing* de veículos de transporte flexível (DARP), estando esta secção dividida em duas subseções. A primeira subsecção, "Algoritmos aplicados ao DARP estático", apresenta as estratégias utilizadas para resolver diferentes problemas aplicados à vertente estática do DARP. Sendo que, a segunda subseção, "Algoritmos aplicados DARP dinâmico" apresenta as estratégias usadas para resolver diferentes problemas aplicados à vertente dinâmica do DARP. Ademais, no final de cada subsecção, é ainda apresentada uma tabela resumo, contendo as principais características dos diferentes problemas de *routing* considerados e ainda a estratégia utilizada para resolver os mesmos.

Por fim, a terceira secção (2.3) deste capítulo, aborda um pouco o tema da simulação e suas principais abordagens. Para além disto, são ainda apresentados os dois principais métodos para representação do tempo em simulação.

2.1 Problema de *routing* de veículos (DARP)

O Dial-a-Ride Problem (DARP) caracteriza o problema do desenho de percursos e do agendamento de serviços de mobilidade de transporte flexível, para um dado número de utilizadores, com origem e destinos previamente especificados. Sendo que a grande diferença entre o DARP e outros problemas de geração de percursos (*routing*), é o facto de este incluir o aspeto humano, isto é, a satisfação do cliente no planeamento dos seus serviços, sendo este tipo de problema também muitas vezes associado ao transporte de doentes ou de idosos.

Existem várias formulações distintas do DARP na literatura atual, no entanto o conceito geral do DARP necessita que um dado número de clientes, efetue um conjunto de pedidos de serviço. Nos quais, cada cliente indica que necessita de ser transportado desde uma localização inicial (origem) para uma localização final (destino), de forma a respeitar algum tipo de restrições temporais. As restrições temporais mais tipicamente utilizadas, são as chamadas restrições de intervalos de tempo (*Time Windows*), que consistem em, para cada cliente, definir um intervalo de tempo para o mesmo ser "recolhido" na sua localização inicial e o intervalo de tempo desejado para ser "entregue" na sua localização final. O serviço de transporte será efetuado por um ou mais veículos, conforme a necessidade, sendo que cada veículo transportará um ou mais clientes, fornecendo assim, uma espécie de serviço partilhado. O objetivo é gerar um conjunto de percursos viáveis, com custos mínimos que acomodem todos os pedidos atualmente existentes e tendo em conta um conjunto de restrições [1].

A maioria das formulações deste problema são normalmente caracterizados por dois objetivos conflituosos: a minimização dos custos operacionais e a minimização da inconveniência para os utilizadores [1], havendo então, a necessidade de encontrar um equilíbrio entre ambos estes objetivos. Sendo estes objetivos geralmente uma combinação entre várias métricas que os permitam refletir.

Este problema será agora descrito em mais detalhe e tendo em conta as características presentes na literatura atual. De forma geral o DARP assume que existe uma frota com n veículos

(n > 0), podendo esta frota ser homogénea – veículos semelhantes com a mesma capacidade – ou heterogénea – veículos com capacidades e/ou características distintas. Note-se que existem formulações para este problema na literatura que consideram apenas um único veículo, no entanto, estas não serão abordadas por não serem pertinentes para um serviço de transporte flexível, uma vez que este geralmente irá lidar com uma frota de veículos.

Cada veículo tem um ponto de partida inicial base, de onde cada um parte, e ainda um ponto de chegada final para onde o mesmo retorna no fim do seu serviço. Estes pontos podem serem partilhados por todos os veículos, ou, cada veículo poderá ter um ponto de partida e de chegada distinto em relação aos outros veículos. É importante referir que existem formulações, nas quais o ponto de partida de um dado veículo, poderá também ser o seu ponto de chegada final para onde o mesmo retorna.

Os clientes ao efetuarem um pedido de serviço, definem qual o seu ponto de "recolha" e o seu ponto "destino" e ainda algum tipo de restrição temporal que indica quando é que estes pretendem ser "recolhidos" e quando é que estes pretendem chegar ao seu destino. No que toca às restrições temporais estas são geralmente feitas através da imposição de limites intervalos de tempo. Estes podem ser impostos tanto na "entrega" como na "recolha" do cliente como em, ou em apenas numa delas como é possível verificar na Tabela 1.

Restrição de intervalo de tempo	**Total + Referência**
Na "recolha" e "entrega" do cliente	N = 12 ([2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13])
Na "recolha" ou "entrega" do cliente	N = 8 ([1] [14] [15] [16] [17] [18] [19][20])

Tabela 1 - Restrições de intervalo de tempo presentes na literatura considerada

Através desta tabela é possível compreender que para a literatura considerada, o intervalo de tempo imposto tanto na "recolha" como também na "entrega do cliente" é o mais prevalente (N=12).

A Tabela 2 demonstra as diferentes restrições e/ou características encontradas na literatura considerada para as formulações deste tipo de problema (excluindo as já faladas restrições de intervalo de tempo).

Restrição ou característica	**Total + Referência**
Capacidade do veículo	N = 19 ([1] [2][14] [15] [3][4] [5] [6] [16] [17] [7] [18] [19] [9] [10] [11] [12][20] [13])
Veículos homogéneos	N = 16 ([1] [2][14] [15] [3] [4] [16] [17] [7] [8] [19] [9] [10] [11] [12] [20])
Veículos heterogéneos	N = 4 ([5] [6] [18] [13])
Tempo máximo de viagem do cliente	N = 11 ([1] [14] [15] [6] [16] [17] [7] [18] [19] [11] [20])
Tempo máximo do percurso	N = 9 ([1] [14] [15] [5] [6] [7] [18] [19] [13])
Tempo máximo de trabalho do condutor	N = 2 ([5] [13])
Pausas de trabalho do condutor entre percursos	N = 2 ([5] [13])
Tempo máximo de espera ou atraso do serviço (em relação ao tempo desejado pelo cliente)	N = 1 ([16]))
Tempo máximo para aceitação/recuso de cada pedido de serviço	N = 1 ([20])
Limite no tempo máximo de excesso do tempo de viagem (tempo de viagem real – tempo mínimo necessário para servir o pedido)	N = 1 ([8])
Limite máximo em relação ao desvio do tempo de serviço desejado	N = 1 ([8])

Tabela 2 - Principais restrições e/ou características das formulações do Dial-a-Ride Problem (DARP) existentes na literatura

Para além de restrições temporais, podem também existir restrições de capacidade do veículo, que como o nome indica impõe um limite máximo ao número de pessoas que um veículo pode transportar em um determinado momento, sendo este tipo de restrição bastante comum na literatura considerada (N = 19). Note-se que apenas uma referência, a [8], não considerou este tipo de restrição, na qual os autores do mesmo o justificam por não acharem este tipo de restrição pertinente para o seu problema, pois os veículos utilizados têm grande capacidade (80 lugares) e são raras as situações em que estes se encontram cheios.

Podem ainda existir restrições impostas nos tempos dos clientes como, a duração máxima do serviço para o cliente – o tempo máximo que o cliente passa no veículo desde a entrada no mesmo, até a chegada ao seu destino – ou nos tempos de espera ou atraso do serviço para os mesmos – definição de um tempo limite máximo de espera pelo serviço ou de atraso do serviço para o cliente como acontece em [16]. Sendo a primeira restrição indicada relativamente comum na literatura (N=11). Estas restrições são bastante importantes do ponto de vista do

cliente, pois estas permitem melhorar a qualidade do serviço dos mesmos ao reduzir, ou pelo menos limitar a inconveniência para o mesmo.

Existem ainda limites impostos ao percurso como é o caso da duração máxima do percurso – definição de um tempo limite máximo para um percurso de um determinado veículo - sendo este tipo de restrição também relativamente comum (N=9).

Ainda é possível constatar que existem algumas formulações que impõem restrições relacionadas com os condutores dos veículos como tempos máximos de trabalho dos mesmos e ainda pausas de trabalho entre cada serviço como nas referências [5] e [13].

Ademais, é importante referir que cada pedido de serviço efetuado por cada cliente, impõe ainda três restrições inerentes ao mesmo, sendo estas três restrições as seguintes:

- A restrição de precedência – indica que um cliente deve ser "recolhido" no seu ponto inicial, antes de ser "entregue" no seu ponto destino;
- A restrição do mesmo veículo - na qual o veículo que efetua a "recolha" do cliente deve ser o mesmo que o transporta para o seu destino;
- A restrição de único serviço – indica que o pedido deve ser servido uma única vez.

Note-se que estas restrições não se encontram na Tabela 2, por estarem geralmente presentes em todas as diferentes formulações do DARP existentes na literatura.

De forma geral, o objetivo do DARP é gerar um conjunto de percursos viáveis e possíveis tendo em conta as restrições do problema e indo ao encontro de algum tipo de função objetivo. Relativamente a estas funções objetivos, na literatura considerada, estas abrangem na sua grande maioria dois objetivos conflituosos: a minimização dos custos operacionais e a minimização da inconveniência para os clientes. Sendo que algumas das referências encontradas, por vezes apenas consideram um destes objetivos, enquanto outras têm em conta ambos os objetivos.

A Tabela 3 apresenta os diferentes objetivos encontrados na literatura considerada e suas respetivas referências.

Objetivo	**Total + Referência**
Minimização dos custos operacionais	N = 4 ([1] [15] [19] [20])
Minimização da inconveniência para os clientes	N = 3 ([3] [4] [8])
Combinação entre Minimização dos custos operacionais e inconveniência para o cliente	N = 13 ([2][14] [5] [6] [16] [17] [7] [9] [10] [11] [12] [13] [18])

Tabela 3 – Principais objetivos na literatura considerada

Como é possível constatar, cerca de 2/3 (N=13) das referências consideradas utilizam algum tipo de função objetivo que suporta ambos estes objetivos conflituosos. Cerca de 1/5 das referências consideradas suporta apenas a minimização dos custos operacionais, enquanto que apenas 1/6 suporta a minimização da inconveniência para os clientes. No entanto, é importante notar que para os problemas nos quais apenas é considerado o objetivo da minimização dos custos operacionais, embora a inconveniência do cliente não esteja diretamente presente na

função objetivo, esta será sempre tida em conta inerentemente, isto devido à existência das restrições de intervalos de tempo. Pois o cumprimento das restrições de intervalo de tempo especificadas pelos clientes, permite de certa forma reduzir a inconveniência para os mesmos.

De forma a que seja possível minimizar qualquer um destes objetivos, é necessário encontrar métricas que o permitam fazer. Relativamente à minimização dos custos operacionais as métricas geralmente mais comuns encontradas na literatura são: a minimização da distância total percorrida por cada veículo e a minimização do número de veículos utilizados. Para além destas métricas, existem ainda as referências [13] e [5] cujas funções objetivo consideram ainda custos associados a utilização de cada funcionário que efetua o serviço. As referências [9] e [10] tentam ainda minimizar custos operacionais ao minimizar o número de paragens visitadas.

No que diz respeito à inconveniência dos clientes, uma das métricas mais comuns é a minimização dos tempos de viagem dos mesmos ([14] [4] [6] [16] [8] [9] [10]) – isto é, o tempo que o cliente passa dentro do veículo enquanto o seu serviço está a ser efetuado. Uma outra métrica que também é por vezes utilizada, é a minimização do excesso do tempo de viagem do cliente ([2] [4] [6] [16] [8]) – diferença entre o tempo de viagem do cliente que este passa no veículo se a viagem fosse efetuado diretamente e o tempo real de viagem do cliente. A minimização do tempos de espera dos clientes pelo serviço ([2] e [5]), é também uma outra métrica por vezes utilizada para medir a inconveniência para os clientes. Para além das já referidas métricas, a minimização do desvio do tempo de serviço ([8]) – diferença entre os intervalos de tempo de serviço desejado e os intervalos do tempo de serviço reais – também pode ser utilizada como uma das formas para medir a inconveniência para os clientes.

Para os problemas cujo objetivo tem a minimização de ambos os já referidos objetivos conflituosos, esta minimização, geralmente é efetuada através da junção/combinação de algumas das já referidas métricas de forma a que acomodem ambos os objetivos.

Note-se que para quando múltiplas métricas são consideradas, é necessário ainda utilizar funções multiobjetivo que permitam minimizar cada um destes objetivos (métricas). Estas funções multiobjectivos são geralmente obtidas através de uma combinação entre as diferentes métricas consideradas ao transformá-las num único objetivo de forma a minimizar o seu custo total. Um dos métodos mais utilizados para este fim é a atribuição de somas ponderadas, isto é atribuir um rácio a cada uma das métricas consideradas conforme a sua importância (maior rácio = maior importância), como em [6] [16] e [4].

É ainda importante referir que, este problema é também caracterizado em duas vertentes: a vertente estática e a vertente dinâmica. Sendo que na primeira vertente, toda a informação necessária para a construção do percurso é conhecida inicialmente. Sendo que para o segundo caso, toda a informação necessária para a construção dos percursos é revelada dinamicamente durante a execução, sendo assim necessário efetuar reajustes em tempo real (ou tempo quase real), aos percursos atuais de forma a satisfazer os pedidos de serviço atuais, sendo então necessária a aplicação de mecanismos de reotimização dos percursos. No entanto, é importante salientar que problemas deste tipo puramente dinâmicos raramente existem uma vez que um subconjunto dos pedidos serão sempre geralmente conhecidos previamente [21].

Na Tabela 4 encontram-se representadas ambas as vertentes, e seu respetivo número de referências encontradas na literatura considerada.

Abordagem	**Total + Referência**
Estática	N = 11 ([1] [2][14] [15] [3][4] [5] [6] [16] [17] [7])
Dinâmica	N = 9 ([8] [18] [19] [9] [10] [11] [12][20] [13])

Tabela 4 – Principais abordagens do Dial-a-Ride Problem e referências consideradas

Podemos ver que no que toca à distribuição de ambas as abordagens a quantidade de artigos considerados para ambas são semelhantes (N = 11 para o DARP estático e N = 9 para o DARP dinâmico). Saliente-se ainda que por vezes existem referências nas quais os autores tratam de um problema do DARP estático e nos quais os mesmos, posteriormente publicam um outro artigo de forma a tratar a abordagem dinâmica para esse mesmo problema como é o caso de [5] e [13].

2.2 Algoritmos utilizados para solucionar o DARP

De forma geral o grande objetivo dos algoritmos para ambas as vertentes do Dial-a-Ride Problem é encontrar uma solução viável e por sua vez possível, que respeite as restrições do problema e que sejam geradas em tempo oportuno, apresentando assim algumas necessidades de desempenho. É importante referir que o encontro de soluções para o DARP é considerado um problema NP-complexo (*NP-hard*) [12], o que faz com que o tempo para encontrar uma solução possa variar bastante, uma vez que este é não determinístico e polinomial. Isto também leva a que o encontro de soluções ótimas possa demorar imenso tempo, dependendo do problema em questão. Assim sendo, na literatura atual, tem sido colocada grande ênfase na descoberta de métodos e soluções eficientes e eficazes para a resolução deste tipo de problema. Note-se que o chamado "tempo oportuno" para a geração de soluções será relativo dependendo das necessidades de cada sistema e também do tipo de abordagem em que o método ou algoritmo se insere. No entanto, é notório que os algoritmos que se inserem na abordagem dinâmica irão apresentar maiores necessidades de desempenho, uma vez que estes necessitam de uma geração continuada de novas soluções e em tempo real (ou pelo menos quase real), conforme as novas informações de serviço se vão revelando ao longo da execução, de forma a tentar satisfazer todos pedidos de serviço e restrições do problema.

Um outro aspeto diferenciador da abordagem estática em relação à dinâmica, é que os algoritmos dinâmicos necessitam de algum tipo de mecanismo de reotimização. Sendo que o processo de reotimização ocorre geralmente no momento em que surgem novos pedidos dinâmicos no qual será necessário alterar os percursos atuais dos veículos de forma a acomodá-los. De modo geral este processo pode ser resumido da seguinte forma:

- Construir um novo percurso atual para um dado veículo de forma a que acomode todos os pedidos que já tenham sido delegados ao mesmo (pedidos já aceites) e os novos pedidos (pedidos dinâmicos) que acabaram de surgir;

- Após obter um percurso inicial que acomode todos esses pedidos, será efetuada a reotimização de forma a tentar encontrar uma melhor solução para esse percurso inicial.

Neste capítulo serão brevemente descritos os algoritmos/métodos encontrados na literatura considerada para resolução de ambas as abordagens (estática e dinâmica) do DARP.

2.2.1 Algoritmos aplicados ao DARP estático

Os autores de [4] utilizaram um algoritmo heurístico de inserção para resolução do DARP estático tendo como principal objetivo minimizar a inconveniência para o cliente. Este algoritmo alterna entre duas fases: fase de geração de percursos para os veículos e fase de agendamento. O número de clientes considerado nos testes efetuados foi de 85. Este algoritmo conseguiu reduzir a inconveniência para os clientes em cerca de 40% e os custos operacionais na realização do serviço para cerca de 14% em relação a um outro algoritmo que foi testado utilizando o mesmo *dataset*.

Em [17] foi também utilizado um algoritmo heurístico de inserção, com duas fases principais para a resolução do DARP estático. A primeira fase é a fase de inserção, onde são inseridos os clientes num percurso de um veículo e a segunda é a fase de otimização, que consiste em otimizar os percursos tentando minimizar o custo da inserção de um cliente no percurso de um veículo. Estes consideraram como objetivo tanto a minimização dos custos do operador como também a inconveniência para o cliente. Foram efetuados testes para 250 clientes com dados simulados e para 2617 clientes utilizando dados reais. Para os dados simulados (n=250) o desempenho computacional foi cerca de 20 segundos para cade execução. Para os dados reais (n=2617) as soluções foram geradas em cerca de 12 minutos. No entanto é realçado que na altura da publicação desta referência (1986), os autores não tinham forma de comparação com outras soluções, logo não sabem se as soluções encontradas são “ótimas” ou se aproximam das mesmas.

Em [14] os autores utilizaram um algoritmo meta heurístico: pesquisa Tabu de forma a resolver o DARP estático cujo objetivo é uma combinação entre minimização do custo do percurso e a inconveniência para o cliente e no qual o número de clientes para os diferentes testes efetuados encontrava-se entre 24 e 295. Estes utilizaram 3 abordagens distintas: P1, no qual aplicavam apenas os dois primeiros passos do algoritmo, P2, onde eram aplicados os seis primeiros passos do algoritmo e minimizada a duração do percurso e P3, onde era aplicado o processo completo do algoritmo de forma a minimizar a duração do percurso e tempos de viagem dos clientes. Estes indicaram que P1 era o mais leve a nível computacional, mas também o que produzia piores resultados, sendo P2 melhor que P1 em relação às soluções geradas e sendo este um pouco mais lento. P3 foi o que melhores resultados produziu, no entanto, tendo também um maior custo computacional, sendo então mais lento. Para as maiores instâncias (n=295) o tempo computacional para a geração de uma solução de P3 foi relativamente alto, estando este entre 192 e 268 minutos.

Na referência [16] foi utilizado um algoritmo heurístico de inserção paralela baseado em arrependimento para resolver um problema do DARP de grande escala, sendo objetivo minimizar uma combinação entre os custos operacionais e a inconveniência para o cliente. No qual este algoritmo foi testado para instância entre 500 e 1000 clientes, tendo este, de forma

geral permitido encontrar melhores soluções em relação a outros algoritmos considerados, e permitindo também aumentar a ocupação de cada veículo. Também foi demonstrado que este algoritmo é também superior a um algoritmo clássico de inserção para problemas de grande escala e obtendo resultados semelhantes para problemas mais pequenos. No que toca ao desempenho do algoritmo, este demonstrou um tempo computacional de 195 minutos para a instância mais complexa, onde o número de clientes considerados era 1000.

O artigo [1] aplica um algoritmo exato de *branch-and-cut* de forma a encontrar soluções viáveis para o DARP estático cujo objetivo é minimizar o custo total do percurso. Os autores indicam que a sua abordagem funciona bem para pequenas instâncias onde o número de clientes é menor que 100 ($n < 100$), conseguindo obter resultados ótimos para $n = 30$ e ainda realçam que a utilização de *user cuts* melhora significativamente o desempenho do algoritmo. Relativamente ao tempo computacional estes apresentam tempos relativamente altos, sendo estes cerca de 240 minutos para resolver as maiores instâncias.

Em [5] foi utilizado um algoritmo heurístico mais focado para resolver problemas do DARP estático de grande escala, no qual este era caracterizado por quatro fases:

1. Fase de pré-processamento - fase na qual algumas constantes que serão usadas frequentemente são computadas;
2. Fase de construção – fase na qual é obtida uma solução inicial viável;
3. Fase de melhoramento - é responsável por pôr em prática uma estratégia de pesquisa local que explora a região local de pesquisa de soluções e uma estratégia de diversificação de forma a direcionar a pesquisa local para outras regiões onde possa haver melhores soluções;
4. Fase de intensificação - é onde a solução final será refinada.

O objetivo principal era a redução de os custos operacionais e inconveniência para o cliente. O número de clientes considerado para os testes estava entre os 50 e 2000 clientes, tendo estes conseguido resolver instância para 2000 clientes dentro de 10 horas.

Os autores de [6] aplicaram um algoritmo heurístico de inserções e reinserções paralelas constituído por três fases para a resolução do DARP estático cujo objetivo era minimizar tanto a inconveniência do cliente como também os custos operacionais. Estas fases eram:

1. Fase de ordenação - onde eram calculados e classificados cada pedido para uma ordem particular de inserção,
2. Fase de inserção paralela - onde é efetuada a inserção dos pedidos nos percursos já existentes;
3. Fase de pós-otimização local - onde é utilizada uma estratégia de pesquisa local de forma a otimizar a função objetivo tentando assim melhorar a solução atual de forma a encontrar uma solução ótima local.

O número de clientes considerado para os testes foi de 150, sendo que o algoritmo proposto demonstrou vantagens, permitindo reduzir os custos da função objetivo em relação a um algoritmo clássico de inserção paralela.

Os autores de [2] optaram por uma abordagem *cluster-first*, *route-second*: onde na primeira fase é utilizado um algoritmo genético de forma a atribuir veículos aos clientes, e na segunda fase é utilizado uma modificação do algoritmo de pesquisa heurístico: *space-time*

nearest-neightbour de forma a encontrar percursos viáveis e possíveis para esses veículos. Sendo o objetivo minimizar uma combinação entre a inconveniência para o cliente e o custo do percurso. Foram efetuados testes para instâncias de 24 até 144 clientes, tendo sido utilizado o mesmo *dataset* que os autores de [14]. Em relação a [14], os autores de [2] indicam que conseguiram diminuir a inconveniência para o cliente refletindo-se na redução dos tempos de espera e de viagem dos clientes, no entanto tendo um pequeno aumento no custo total do percurso. Sendo que estes também apresentam melhorias nos tempos computacionais em relação a [14] para instâncias com maior duração de percurso e pior desempenho para instâncias com menos duração. O tempo computacional para a maior instância considerada foi de 66 minutos para o algoritmo proposto e de 92 minutos para essa mesma instância em [14], ou seja permitindo poupar cerca de 26 minutos de computação.

Em [3] foi utilizado um algoritmo heurístico com vista a minimizar a inconveniência para o cliente. Este algoritmo sucintamente resume-se a:

1. Resolver o problema de atribuição entre veículos e clientes conhecidos;
2. Inserir os sub-percursos gerados no percurso principal;
3. Modificar os caminhos impossíveis de forma a torná-los possíveis.

O número de clientes considerado para os testes esteve entre 10 e 180. Os autores omitiram ainda o tempo de computação indicando que este é em média menor que 0.05 segundos e ainda relatam que em relação a um algoritmo heurístico de inserção básico, o algoritmo proposto permite servir mais pedidos, permitindo então diminuir ainda mais inconveniência para os clientes.

Os autores de [15] optaram pelo método exato *branch-and-cut* aplicando uma estratégia *best-bound* de forma a encontrar soluções viáveis para o problema estático cujo objetivo era a minimização dos custos operacionais. Estes consideram duas formulações do problema (PDTW1 e PDTW2) e ainda comparam as soluções de cada uma dessas formulações aos resultados de [1], sendo notória uma melhoria no desempenho computacional em relação aos mesmos. O número de clientes considerados para estes testes foi entre 16 e 96. O tempo computacional para as maiores instâncias consideradas esteve entre os 84 minutos e 120 minutos dependendo da formulação considerada, sendo que PDTW2 apresentou melhor desempenho em relação a PDTW1.

Na referência [7] foi utilizado um algoritmo heurístico de duas fases baseado no esquema de decomposição de cenários para a resolução do DARP estático. Sendo este artigo particularmente interessente, no qual os autores introduzem a noção de dois tipos de pedidos distintos, *early requests* – pedidos efetuados no dia anterior ou com bastante antecedência - e *late requests* – pedidos efetuados no dia do percurso até ao prazo final da reserva do serviço. Na primeira fase foi utilizada a pesquisa Tabu, onde é construído o percurso inicial utilizando os pedidos já efetuados (*early requests*). Relativamente à segunda fase, é utilizado um mecanismo de coordenação, de forma a construir possíveis desvios no percurso atual ao serem inseridos os novos pedidos (*late requests)*. A função objetivo tem como principais objetivos a minimização dos custos operacionais e da inconveniência para os clientes. No entanto, é importante notar que para a inserção de novos pedidos (*late requests*) esta função objetivo tenta minimizar o custo da inserção dos novos pedidos de forma a os satisfazer e tentando assim também minimizar a inconveniência para os outros clientes que já efetuaram o pedido antecipadamente

(*early requests*). No que toca aos testes, os autores utilizaram um número de clientes entre 24 e 144 e ainda utilizaram 3 classes de instâncias distintas na qual em cada uma delas era variada a quantidade de pedidos conhecidos inicialmente (*early requests*) desde 20% até 80%. Com estes testes os autores conseguiram compreender que o esforço computacional necessário para obter uma solução é influenciado pelo surgimento de novos pedidos (*late requests*), sendo então este esforço cada vez maior quanto maior o número de novos pedidos dinâmicos.

A Tabela 5 apresenta um resumo das principais características das diferentes referências encontradas para o DARP estático.

Referência	**Objetivo**	**Intervalos de tempo**	**Outras restrições**	**Algoritmo**	**Número de instâncias de teste**
[4]	Minimiza a inconveniência para o cliente.	Limites superiores impostos na recolha e entrega do cliente.	Capacidade do veículo.	Algoritmo heurístico que alterna entre duas fases: fase de geração de percursos e de agendamento.	$n = 85$
[17]	Minimiza uma combinação entre custos operacionais e inconveniência para o cliente.	Na recolha ou entrega do cliente.	Capacidade do veículo. Tempo máximo de viagem do cliente.	Algoritmo heurístico de inserção com 2 fases: fase de inserção e fase de otimização.	$n = 250$ e $n = 2617$
[14]	Minimiza uma combinação entre custos operacionais e inconveniência para o cliente.	Na recolha ou entrega do cliente	Capacidade do veículo. Duração máxima do percurso. Tempo máximo de viagem do cliente.	Pesquisa Tabu (*Tabu Search*).	$24 \leq n \leq 295$
[16]	Minimiza uma combinação entre custos operacionais e inconveniência para o cliente.	Na recolha ou entrega do cliente.	Capacidade do veículo. Tempo máximo de viagem do cliente. Tempo máximo de espera/atraso no local de recolha ou entrega do cliente.	Algoritmo heurístico de inserção paralela baseado em arrependimento.	$n = 500$ e $n = 1000$
[1]	Minimiza os custos operacionais.	Na recolha ou entrega do cliente	Capacidade do veículo. Duração máxima do percurso. Tempo máximo de viagem do cliente.	*Branch-and-cut.*	$16 \leq n \leq 36$
[5]	Minimiza uma combinação entre custos operacionais e	Na recolha e entrega do cliente.	Capacidade do veículo. Diferentes tipos de veículos. Duração máxima do percurso. Pausas do condutor entre	Algoritmo heurístico com 4 fases principais: pré-processamento, fase de construção, fase de	$50 \leq n \leq 2000$

	inconveniência para o cliente.		viagens e tempos de trabalho máximos do condutor.	melhoramento e fase de intensificação.	
[6]	Minimiza uma combinação entre custos operacionais e inconveniência para o cliente.	Na recolha e entrega do cliente.	Capacidade do veículo. Duração máxima do percurso. Tempo máximo de viagem do cliente. Vários tipos de veículos.	Algoritmo heurístico de 3 fases: Fase de ordenação, Fase de heurística de inserção paralela e Fase de pós-otimização local.	$n = 150$
[2]	Minimiza uma combinação entre custos operacionais e inconveniência para o cliente.	Na recolha e entrega do cliente	Capacidade do veículo.	Utilizando uma abordagem *cluster-first, route-second*. Na fase cluster-first é utilizado um algoritmo genético. Sendo na fase de roteamento utilizada uma versão modificada da heurística *space-time nearest-neighbour*.	$24 \leq n \leq 144$
[3]	Minimiza a inconveniência para o cliente.	Na recolha e entrega do cliente.	Capacidade do veículo.	Algoritmo heurístico que se resume a: Resolver problema de atribuição, inserir sub-percursos no percurso principal, modificar os caminhos inviáveis de forma a os tornar viáveis.	$10 \leq n \leq 180$
[15]	Minimiza os custos operacionais.	Na recolha ou entrega do cliente.	Capacidade do veículo. Duração máxima do percurso. Tempo máximo de viagem do cliente.	*Branch-and-cut* utilizando a estratégia *best-bound*.	$16 \leq n \leq 96$
[7]	Minimiza uma combinação entre custos operacionais e inconveniência para o cliente.	Na recolha e entrega do cliente.	Duração máxima do percurso. Tempo máximo de viagem do cliente. Capacidade do veículo.	Algoritmo heurístico de duas fases baseado no esquema de decomposição de cenários, utilizando pesquisa tabu na primeira fase e mecanismo de coordenação na segunda fase.	$24 \leq n \leq 144$

Tabela 5 - Tabela resumo com as principais características das referências consideradas para o DARP estático

2.2.2 Algoritmos aplicados ao DARP dinâmico

Em [18] foi utilizado um algoritmo heurístico de inserção baseado no de [17], mas neste caso, sendo este utilizado de forma a resolver vertente dinâmica do DARP, com o objetivo minimizar tanto a inconveniência como também os custos operacionais. Sendo este algoritmo um pouco melhorado de forma aumentar um pouco o seu desempenho computacional e também a sua flexibilidade em relação ao algoritmo proposto em [17], no entanto os autores notam que a qualidade da solução produzida poderá não ser superior à do mesmo. Os testes foram efetuados para 300 clientes, onde era possível determinar uma solução em menos de 10 segundos e sendo ainda possível inserir um novo cliente (dinâmico) em menos de 1 segundo.

Na referência [11] foi utilizado um algoritmo baseado num modelo de lógica *fuzzy* para resolver o DARP dinâmico, tendo este como objetivo a minimização dos custos operacionais e inconveniência para o cliente. Este modelo tem dois algoritmos principais de raciocínio aproximado: o primeiro algoritmo toma decisões acerca de qual o veículo que será delegado o novo pedido, sendo que o segundo é utilizado para a construção do novo percurso e itinerário para o veículo que foi selecionado pelo primeiro algoritmo para servir o novo pedido. Tendo ambos estes algoritmos um total de 9 regras de lógica *fuzzy*. Os testes foram gerados para um intervalo desde as 05:00h até as 23:00h, tendo sido gerados 900 pedidos de serviço nesse intervalo de tempo.

Em [19] foi adotada a utilização de um algoritmo de pesquisa Tabu baseado no de [14], tido sido este estendido para resolução do DARP dinâmico e ainda utilizando computação paralela de forma a melhorar o desempenho. O objetivo principal era a minimização dos custos operacionais. O algoritmo funciona da seguinte forma: inicialmente é gerada uma solução inicial com os pedidos já conhecidos, quando surge um novo pedido (dinâmico), é efetuada uma verificação de viabilidade de forma a encontrar uma solução viável com a inclusão desse novo pedido. Assim que tenha sido decidido se esse pedido pode ser aceite ou não, é efetuada pós-otimização de forma a tentar melhorar a solução atual. Os testes foram efetuados usando as instâncias de teste de [14], onde o número de clientes estava entre 24 e 295, para além disto, estes ainda testaram com instâncias aleatoriamente geradas. Os autores impuseram um limite de 30 segundos para efetuar a verificação de viabilidade, e comparam os seus resultados com as soluções para os casos em que não são impostos tempos limites para esta verificação. De forma geral as soluções geradas com o limite imposto, são um pouco piores em relação às que não tem limite, pois estas servem menos clientes no total, sendo esta diferença cerca de 2.5% do total de clientes servidos. Os autores ainda referem que as instâncias aleatoriamente geradas são geralmente mais difíceis de resolver do que as reais.

Os autores de [8] optaram pela utilização de um algoritmo de inserção com duas fases para resolver o DARP dinâmico, permitindo inserir novos pedidos (dinâmicos) e reotimizar o percurso à medida que os mesmos surgem. A primeira fase é responsável por gerar e selecionar o melhor percurso atual encontrado com os pedidos já conhecidos. Sendo a segunda fase responsável por tratar os novos pedidos que surgem, de forma a minimizar a inconveniência para os clientes que já têm os seus pedidos designados a um veículo. Note-se que sempre que um cliente novo é inserido, é necessário voltar à primeira fase de forma a atualizar o percurso atual. Estes apresentam diferentes testes para uma procura total de entre 25 e 50 utilizadores, e variando também a quantidade de novos dinâmicos pedidos que surgem entre 5 até 15 para

esse total de utilizadores. Estes ainda comparam a qualidade da solução ao comparar o custo total da solução obtida com os novos pedidos em relação à solução se os pedidos efetuados fossem todos estáticos, ou seja, conhecidos antecipadamente. Como era de esperar este custo aumenta com a introdução de novos pedidos dinâmicos, sendo este mais aproximado do custo da solução estática para quando existem menos pedidos dinâmicos.

Em [13], os autores propõe uma heurística rápida e eficiente para o processo de reotimização dos percursos e itinerários dos veículos. Sendo que esta heurística adota uma estratégia propriamente organizada de procura local e utiliza estratégias de diversificação que permitem guiar a pesquisa local para fora de ótimas locais, permitindo assim encontrar melhores soluções. A heurística utilizada é semelhante à de [5], mas sendo desta vez utilizada para resolver o DARP dinâmico de forma a minimizar os custos operacionais e a inconveniência para o cliente. Para além disto, estes ainda consideram um conjunto distinto de eventos estocásticos como: o surgimento de novos pedidos, a velocidade do veículo, o tempo de serviço, avarias nos veículos, cancelamento de pedidos, entre outros. Sendo que cada um destes eventos tem a sua prioridade. Os testes foram efetuados em simulação entre as 7:00h e até depois das 17:00h, acabando quando todos os pedidos tivessem sido servidos. No geral os resultados obtidos demonstraram que o algoritmo proposto apresenta melhor desempenho para quando existem pouco pedidos dinâmico. Sendo que o aumento da quantidade de pedidos dinâmicos faz com que a solução/desempenho do algoritmo proposto se vá deteriorando. Em relação aos tempos computacionais, o algoritmo proposto conseguiu atingir requisitos de aplicações reais, nos quais a maioria dos eventos eram tratados em menos de 1 minuto.

Os autores de [9] utilizaram um algoritmo chamado de *Multi-Objective Simulated Annealing* (MOSA) como forma de resolver o DARP dinâmico, sendo o principal objetivo do mesmo tanto a minimização da inconveniência do cliente como também a minimização dos custos operacionais. Este algoritmo é caracterizado por reduzir o tempo computacional, sendo assim útil para a vertente dinâmica do DARP. Existem duas fases principais do algoritmo:

1. Fase de atribuição de pedidos de serviço - é utilizado um método de classificação baseado na aproximação de intervalos de tempo dos pedidos de serviço. Sendo também nesta fase gerada uma solução inicial para o problema, através de uma heurística de distribuição.
2. Fase de planeamento de percurso - é utilizada uma estratégia de pesquisa local na estrutura do algoritmo MOSA de forma a gerar o melhor percurso para o veículo.

No entanto é importante notar que o algoritmo proposto não permite efetuar desvios ao percurso de um veículo de forma a ir servir novos pedidos que surgem, sendo que a partir do momento que um veículo se encontra em serviço não é possível alterar o seu percurso. Os testes foram efetuados em tempo de simulação de 4 horas, onde eram gerados 20 pedidos por hora, na qual esta simulação decorria em tempo real. Os autores compararam o algoritmo proposto com um algoritmo de pesquisa local básico e uma heurística de inserção básica para as diferentes instâncias consideradas. Em geral para esses testes, o MOSA mostrou-se superior em relação a ambos esses algoritmos.

Os autores de [12] propõem quatro algoritmos estocásticos meta heurísticos distintos de forma a resolver o DARP dinâmico tendo como objetivo minimizar tanto a inconveniência para o cliente como também os custos operacionais. Estando este DARP mais focado no transporte

de doentes de casa para o hospital ou vice-versa. No problema foi ainda incluído um conjunto de informação estocástica acerca do transporte de retorno para casa, isto é, do hospital para casa, baseando-se em dados históricos. Assim cada pedido de viagem desde casa para o hospital tinha uma certa probabilidade de causar a existência de um pedido do hospital para casa. Estes algoritmos incluem: *Dynamic* VNS, o S-VNS (*Stochastic VNS*) dinâmico e o MPA (*Multiple Plan Approach*) e MSA (*Multiple Scenario Approach*). Sendo a ideia de os autores comparar os quatro algoritmos e ainda compreender se a utilização de informação estocástica teria algum benefício. O número médio de pedidos utilizado em simulação para os testes estava entre os 340 e 470 pedidos. Os autores impuseram diferentes limites em relação ao período de tempo em que a informação estocástica sobre o futuro é tida em conta, tendo estes afirmado que este período deve ser relativamente pequeno, indicando que este não deve ultrapassar os 20 minutos para as instâncias de teste utilizadas. No que toca aos quatro métodos, o que melhor resultado apresentou foi o S-VNS, demonstrando assim que a utilização de informação estocástica pode ser vantajosa para problemas deste tipo.

Em [10] os autores utilizaram o mesmo algoritmo presente em [9], *Multi-Objective Simulated Annealing* (MOSA) constituído também pelas mesmas duas fases (fase de atribuição de pedidos e fase de planeamento de percurso), sendo a grande diferença, o facto de este ter sido estendido de forma a poder utilizar a pesquisa Tabu (*Tabu Search*), obtendo assim o algoritmo híbrido denominado por MOSA-TS para a resolução do DARP dinâmico. O objetivo do mesmo era a minimização dos custos operacionais como também a inconveniência para o cliente. Foram também efetuados testes para um tempo de simulação de 4 horas, onde eram gerados 20 pedidos por hora ou 35 pedidos por hora, sendo esta simulação efetuada em tempo real. Para cada um destes testes ainda os autores compararam os resultados do seu MOSA-TS com os obtidos pelo MOSA de [9]. Para a instância de teste de 20 pedidos por hora e 35 pedidos por hora, no geral o MOSA-TS apresentou melhores resultados em relação às soluções geradas. Os autores justificam esta melhoria pela utilização da pesquisa Tabu, na qual esta permite melhor explorar o espaço de soluções. Como já seria de esperar, os autores notam que o desempenho da pesquisa Tabu se deteriora com o aumento do número de pedidos, sendo isto possível verificar nas diferenças entra as soluções geradas para as diferentes instâncias de teste em que o número de pedidos é aumentado.

Em [20] é proposto um algoritmo baseado em arrependimento (*regret-based*) chamado de OR-DARP (*Online Regret Dial-a-Ride Problem*) de forma a resolver o DARP dinâmico, no qual os autores indicam que este deve ser usado para situações onde exista bastante procura de serviço. Tendo este algoritmo como objetivo principal a minimização dos custos operacionais, e sendo este composto por dois algoritmos principais:

- Algoritmo 1: algoritmo heurístico de inserção rápida online, responsável por lidar com os pedidos de serviço que surgem, tomando decisões de aceitação/recusa acerca dos mesmos;
- Algoritmo 2: é um algoritmo de otimização baseado em arrependimento responsável por ir continuamente melhorando as soluções dos percursos conforme os novos pedidos surgem.

Este algoritmo diferencia-se de todos os outros na literatura considerada, pela existência de um conceito no processo de otimização, sendo que a otimização não é efetuada baseada em intervalos de tempo constantes (e.g. 30 segundos para otimizar), mas sim opera continuamente de forma a melhorar a solução tomando proveito mesmo dos mais pequenos fragmentos de

tempo de espera ou inativo entre cada pedido de serviço que surge. Os testes foram efetuados definindo um tempo limite para a aceitação de cada pedido de 1 minuto, e utilizando o algoritmo OR-DARP com processo de otimização e OR-DARP sem o processo de otimização de forma a compreender de que forma é que a existência deste processo influencia o algoritmo. O número de pedidos esteve entre 94 e 1619 para os testes considerados. Os autores conseguiram perceber que o processo de otimização tem as suas limitações para instâncias cujo número de pedido é relativamente alto, no qual não é possível aceitar todos os pedidos dentro do tempo estipulado de 1 minuto, ocorrendo o oposto para o OR-DARP sem otimização onde é possível aceitar todos os pedidos. No entanto, para pequenas instâncias os OR-DARP com otimização mostrou-se superior ao OR-DARP sem otimização, permitindo encontrar melhores soluções.

A Tabela 6 apresenta um pequeno resumo acerca das características principais das referências consideradas para a vertente dinâmica do DARP.

Referência	**Objetivo**	**Intervalos de tempo**	**Outras restrições**	**Algoritmo**	**Número de instâncias de teste**
[18]	Minimiza uma combinação entre custos operacionais e inconveniência para o cliente.	Na recolha ou entrega do cliente.	Vários tipos de veículos. Capacidade do veículo. Tempo máximo de viagem do cliente. Duração máxima do percurso.	Algoritmo heurístico de inserção.	$n = 300$
[11]	Minimiza uma combinação entre custos operacionais e inconveniência para o cliente.	Na recolha e entrega do cliente	Capacidade do veículo. Tempo máximo de viagem do cliente.	Algoritmo de raciocínio aproximado usando lógica *fuzzy* constituído por dois algoritmos: Algoritmo 1: toma decisões acerca de qual o veículo onde o novo pedido dinâmico será inserido. Algoritmo 2: constrói um novo percurso e itinerário para o veículo escolhido de forma a servir o novo pedido.	$n = 900$
[19]	Minimiza os custos operacionais.	Na recolha ou entrega do cliente.	Tempo máximo de viagem do cliente. Capacidade do veículo. Duração máxima do percurso.	Algoritmo de pesquisa de tabu paralela com processo de verificação de viabilidade de inserção e pós-otimização.	$24 \leq n \leq 295$
[8]	Minimiza a inconveniência para o cliente.	Na recolha e entrega do cliente.	Limite superior no excesso do tempo de viagem do cliente e no desvio do	Algoritmo de inserção duas fases que permite inserir novos pedidos dinâmico e reotimiza o percurso atual à medida	$25 \leq n \leq 50$

			tempo de serviço desejado.	que estes pedidos surgem.	
[13]	Minimiza uma combinação entre custos operacionais e inconveniência para o cliente.	Na recolha e entrega do cliente.	Capacidade do veículo. Duração máxima do percurso. Tempos máximos de trabalho do condutor. Pausas de trabalho entre percursos. Diferentes tipos de veículos.	Heurística rápida e eficiente que utiliza estratégias de busca local e diversificação propriamente organizadas.	$0.1 \leq n \leq 1$ por minuto
[9]	Minimiza uma combinação entre custos operacionais e inconveniência para o cliente.	Na recolha e entrega do cliente.	Capacidade do veículo.	Algoritmo *Multi-Objective Simulated Annealing* (MOSA)	$n = 20$ por hora
[12]	Minimiza uma combinação entre custos operacionais e inconveniência para o cliente.	Na recolha e entrega do cliente.	Capacidade do veículo. Tempo máximo limite para efetuar um desvio de forma a servir um pedido dinâmico.	Propõe diferentes algoritmos estocásticos meta-heurísticos: *Dynamic* VNS, S-VNS, MAS e MPA.	$340 \leq n \leq 470$
[10]	Minimiza uma combinação entre custos operacionais e inconveniência para o cliente.	Na recolha e entrega do cliente	Capacidade do veículo.	Algoritmo híbrido *Multi-Objective Simulated Annealing Tabu Search* (MOSA-TS).	$n = 20$ e $n = 35$ por hora
[20]	Minimiza os custos operacionais.	Na recolha do cliente.	Capacidade do veículo. Tempo máximo de viagem do cliente. Tempo limite para a aceitação/recusa de cada pedido.	Algoritmo OR-DARP com dois sub-módulos: Algoritmo de inserção rápida de pedidos e algoritmo de otimização baseado em arrependimento	$94 \leq n \leq 1619$

Tabela 6 - Tabela resumo com as principais características das referências consideradas para o DARP dinâmico

2.3 Simulação

Como já referido a simulação torna-se bastante importante do ponto de vista económico, permitindo minimizar custos ao reduzir os recursos utilizados, pois não é necessário utilizar recursos físicos, mas sim simulados. Adicionalmente, é também importante do ponto de vista de avaliação e validação das diferentes abordagens que solucionem o DARP, permitindo testá-las, analisá-las e validá-las virtualmente para que depois possam ser implementadas na realidade.

No entanto de forma a obter um modelo de simulação que se aproxime da situação real, torna-se então necessário estudar e modelar o contexto real do problema. Segundo os autores de [22, p. 893], para obter um modelo de simulação que se aproxime da realidade, de forma geral e sucintamente, é recomendado seguir o seguinte processo:

1. Efetuar uma análise preliminar do contexto/sistema de forma a gerar uma representação aproximada do mesmo;
2. Utilizar a simulação para testar diferentes representações específicas do sistema de forma a compreender qual a que se aproxima mais da realidade;
3. Após a seleção e obtenção da representação mais aproximada do contexto real, o sistema selecionado será continuamente testado e otimizado de forma a obter a representação final do mesmo, que será a que mais se aproximará da realidade.

Foram encontradas duas abordagens principais de simulação na literatura estudada:

- **Simulação de eventos discretos (DES - *Discrete-Event Simulation*)** - é geralmente o tipo de simulação mais utilizado, e é aquele no qual as mudanças do estado do sistema ocorrem instantaneamente em pontos aleatórios de tempo devido à ocorrência de eventos discretos [22, p. 894]. Por outras palavras este tipo de simulação faz com que o tempo de simulação, de certa forma "salte" de evento para evento, alterando então o estado do sistema à medida que os mesmos ocorrem;
- **Simulação continua (CS – *Continuous Simulation*)** – é o tipo de simulação no qual o estado do sistema irá se alterar continuamente ao longo do tempo [22, p. 894]. Assim, neste segundo tipo de simulação os eventos alteram o estado do sistema continuamente, geralmente seguindo algum tipo de função matemática contínua, sendo então o tempo continuo, e não existindo os já referidos "saltos" temporais que ocorrem na simulação de eventos discretos.

Ainda, é importante notar que muitas vezes é possível simular o comportamento de um sistema contínuo, utilizando a simulação de eventos discretos. De facto, é possível através da ocorrência de eventos discretos ocasionais, nos quais estes permitem então ocorrer mudanças aproximadamente continuas ao estado do sistema que está a ser simulado [22, p. 894]. Esta peculiaridade leva a que a simulação de eventos discretos seja muitas vezes preferível à simulação contínua, pois permite que a mesma seja utilizada para modelar tanto sistemas contínuos, como sistemas discretos, tornando-a mais flexível.

Por fim, é importante ainda referir que existem dois métodos principais de representação do tempo no que toca às simulações de eventos discretos (DES), nos quais estes podem ser os seguintes:

- **Incremento de tempo fixo (*Fixed-time incrementing*)** – consiste em continuamente avançar o tempo por um determinado valor relativamente pequeno e fixo, e posteriormente atualizar o estado do sistema de forma a tratar os eventos que ocorreram durante esse tempo que foi avançado [22, p. 900];
- **Incremento do próximo evento (*Next-event incrementing*)** - resume-se a avançar o tempo de evento em evento, no qual este é incrementado para o tempo do próximo evento a ser tratado, ocorrendo então o tratamento do mesmo e atualizando o estado atual do sistema, sendo isto efetuado até que todos os eventos tenham sido tratados [22, p. 902].

3 Analise e desenho da solução

Como já referido, este trabalho tem como objetivo a formulação de uma solução para um problema de *routing* de transporte flexível. É também pretendido testar a referida solução, de forma a analisar a viabilidade da mesma num contexto simulado e aproximado do real. O conhecimento obtido através da simulação servirá de base a uma análise de adequação e adaptabilidade ao contexto real.

3.1 Especificação do Problema

Será seguidamente descrito o problema de *routing* de transporte flexível formulado. Este considera um conjunto paragens existentes num mapa, sendo estas representadas pela sua localização geográfica. Este problema está restringido por um dado número de veículos disponíveis para a realização de cada serviço. Este número de veículos poderá ser variável, dependendo do momento em que é requisitado o serviço e ainda da capacidade de cada veículo, que poderão ter diferentes capacidades máximas de clientes suportados. O objetivo é servir um conjunto de pedidos de clientes utilizando um dado número de veículos, construir e otimizar um conjunto de percursos (viáveis e possíveis) que os sirvam, de modo a respeitar um conjunto de restrições impostas e ainda de modo a minimizar os custos operacionais e maximizar a ocupação de cada veículo.

No que toca aos pedidos de serviço, estes são caracterizados por conter principalmente, um conjunto de informações geográficas e temporais relativas à realização do serviço de cada cliente. Relativamente às informações geográficas, estas incluem os seguintes dados:

- **Paragem de "recolha" (ou origem) do cliente** – indica a paragem ou ponto (origem), onde o cliente será "recolhido" inicialmente por um dado veículo;
- **Paragem de "entrega" (ou destino) do cliente** – indica a paragem ou ponto (destino), onde o cliente será "entregue" pelo veículo que o recolheu inicialmente.

No que diz respeito às informações temporais inerentes a cada pedido de serviço (de cada cliente ou utilizador), estas incluem as seguintes informações:

- **Hora de "recolha" desejada pelo cliente** – indica a hora a partir da qual o cliente estará presente na paragem origem para que possa ser "recolhido";
- **Hora de "entrega" desejada pelo cliente** – indica a hora desejada pelo cliente para chegar à paragem destino ou de "entrega".

Com isto, é possível compreender que ambas estas informações providenciam, de certa forma, algum tipo de restrições temporais e geográficas que devem ser cumpridas em relação à realização de cada pedido de serviço. Adicionalmente, foram ainda impostas restrições relativas ao tempo de serviço de cada cliente, no qual o tempo de serviço para um dado pedido não poderá ultrapassar um certo valor estabelecido. Note-se que, esta restrição foi imposta de forma a melhorar a qualidade do serviço para os clientes, restringindo assim o tempo total máximo que o mesmo poderá passar dentro de um dado veículo enquanto se encontra a ser servido.

Para além disto, é ainda importante referir que cada percurso de cada veículo poderá começar e acabar em paragens arbitrárias ou em paragens previamente estipuladas como sendo

as paragens base iniciais e finais do percurso desse veículo, e sendo ainda possível que estas paragens sejam distintas para cada um dos veículos considerados.

A partir do que foi descrito é possível conceber um conjunto de restrições que compõem o modelo do problema considerado:

1. **Restrição de Intervalo de tempo**: um veículo deverá chegar e partir de uma determinada paragem dentro de um intervalo de tempo definido. Este intervalo contém a hora mínima de chegada a uma paragem e a hora máxima de partida da paragem para um determinado veículo;
2. **Restrição de Precedência**: para um determinado pedido de serviço de um cliente, um veículo deverá passar primeiro pela paragem origem ou de "recolha" desse cliente antes de passar pela paragem destino ou de "entrega" do mesmo;
3. **Restrição do mesmo veículo**: o veículo que passa pela paragem origem de um determinado pedido de serviço (de um cliente), deverá ser o mesmo veículo a passar pela paragem destino do mesmo;
4. **Restrição de Capacidade**: cada veículo tem uma capacidade máxima de passageiros que podem ser transportados, podendo existir veículos com diferentes capacidades;
5. **Restrição de tempo máximo de viagem do utilizador**: são impostos limites no tempo que o utilizador passa dentro de um veículo enquanto o serviço é realizado, de forma a tentar reduzir ou pelo menos limitar a inconveniência para os mesmos;
6. **Restrição do número de veículos**: o problema encontra-se restringido por um dado número de veículos para realizar o conjunto de pedidos de serviço considerados;
7. **Restrição das paragens de começo e fim**: o percurso de cada veículo começa e acaba em paragens previamente estipuladas ou em paragens arbitrarias, podendo a que estas variem de veículo para veículo.

É importante notar que, existem algumas restrições acima apresentadas que não se encontram explicitamente impostas na descrição do problema, nomeadamente as restrições de precedência e a restrição do mesmo veículo. No entanto, estas estão inerentemente presentes no problema, pois são necessárias de modo a garantir coerência na realização de cada pedido de serviço.

3.2 Abordagem

De forma a construir um simulador que permita "mimicar" um sistema de transporte flexível real, foi primeiro efetuado uma análise pormenorizada do contexto em que o problema se insere, melhorando a compreensão do problema e de modo a descobrir as necessidades (requisitos) que o mesmo irá impor para um sistema deste tipo.

Inicialmente foram tomadas algumas decisões acerca da tecnologia que seria utilizada. Foi decido que seria utilizada a biblioteca *Google Operations-Research Tools* (*OR-Tools)*[1], que possibilita a modelação do problema de *routing* anteriormente formulado (em 3.1. Especificação do Problema), e ainda permitiria resolver o mesmo, possibilitando então a obtenção de soluções de percursos para o mesmo. Esta biblioteca foi escolhida por mostrar-se bastante flexível, permitindo facilmente modelar e modificar diferentes problemas de *routing*. Ademais, esta

[1] https://developers.google.com/optimization/

biblioteca contém ainda um vasto conjunto de algoritmos de geração de soluções já implementados, retirando assim a necessidade de os ter que implementar. Adicionalmente, o facto de esta estar disponível para quatro linguagens de programação possibilitou uma grande flexibilidade em relação à escolha da linguagem de implementação do simulador. Para além disto, o facto desta biblioteca ser desenvolvida e mantida pela *Google*, uma das maiores empresas mundiais a nível tecnológico, ser *open-source* e conter boa documentação com exemplos de utilização, foram também fatores que contribuíram para a escolha da mesma. Relativamente à linguagem de programação de implementação do simulador, foi optado por utilizar C#, simplesmente por preferência.

Depois de decidida a tecnologia começou-se por modelar e implementar os aspetos inerentes ao simulador, tendo sido seguida uma abordagem de simulação de eventos discretos. Esta escolha deve-se pelo facto de esta ser mais flexível relativamente à simulação contínua e por ser mais adequada ao contexto do problema, uma vez que o estado do simulador só se irá alterar quando ocorrerem certos eventos estocásticos discretos (e.g. surgimento de um novo pedido de serviço), e não continuamente ao longo do tempo. Relativamente à representação temporal foi decidido seguir uma abordagem de incremento para o próximo evento (*next-event incrementing*), pois esta foi considerada mais fácil de compreender e de implementar. Note-se que, o processo e as decisões tomadas acerca da modelação e implementação do simulador estarão presentes na próxima secção (Desenvolvimento).

É ainda importante referir que, foi-nos disponibilizado dados reais pela empresa Horários do Funchal acerca da sua rede de transporte público para serem utilizados na simulação. Estes dados incluíam um conjunto de informações como: a identificação e localização geográfica de cada paragem presente na cidade do Funchal, todas as carreiras e seus respetivos horários. Para além disto, continha ainda os diferentes percursos para cada carreira, incluindo o conjunto de todas as paragens que um veículo tem que visitar de forma a efetuar a mesma (serviço de transporte público convencional).

3.3 Requisitos

Um simulador de um sistema de transporte flexível irá obviamente apresentar alguns requisitos funcionais e não funcionais, que deverão ser cumpridos de forma a conseguir obter um sistema que se aproxime da realidade e que vá de encontro as necessidades estabelecidas. Os requisitos funcionais preocupam-se com as funcionalidades do referido sistema, descrevendo assim algum tipo de necessidade funcional que o sistema apresente. Sendo que os requisitos não funcionais preocupam-se com as necessidades que não podem ser cumpridas através de funcionalidades do sistema, sendo estes geralmente mensuráveis, definindo características do sistema em questão e impondo também algumas restrições ao mesmo [24].

Após a realização de uma análise pormenorizada do contexto do problema e levantamento das necessidades do referido sistema foi possível conceber um conjunto de requisitos funcionais e não funcionais, nos quais estes serão abaixo apresentados. Relativamente aos requisitos funcionais, foram encontradas as seguintes necessidades:

- **R1** - O sistema deverá permitir simular um percurso de um veículo, isto é, permitir que um veículo visite um conjunto de paragens inerentes ao seu percurso;

- **R2** - O sistema deverá permitir simular a chegada de um veículo a uma paragem, como também a saída do mesmo;
- **R3** - O sistema deverá permitir simular a geração de pedidos de serviço seguindo algum tipo de processo estocástico ou distribuição;
- **R4** - Cada pedido de serviço de cada utilizador deverá conter as seguintes informações: a identificação do utilizador, a paragem origem, a paragem destino, a hora de "recolha" desejada e a hora de "entrega" desejada;
- **R5** - O sistema deverá permitir simular a entrada como também a saída de cada cliente de um veículo;
- **R6** - O sistema deverá permitir gerar percursos de forma a servir um conjunto de pedidos existentes e tendo em conta um conjunto de veículos disponíveis;
- **R7** - O sistema deverá ter estratégias que permitam otimizar os percursos gerados de forma a encontrar uma melhor solução;
- **R8** - O sistema deverá permitir a inserção de pedidos dinâmicos de serviço a percursos já concebidos, otimizando-os novamente;
- **R9** - O sistema deverá permitir simular um conjunto de serviços para um determinado período de tempo;
- **R10** - As estratégias de geração de percursos (*routing*) deverão minimizar os custos operacionais e maximizar a utilização da capacidade de cada veículo;
- **R11** - As estratégias de geração de percursos (*routing*) deverão respeitar um conjunto de restrições impostas (como as referidas anteriormente no capítulo 3.1. Especificação do Problema);
- **R12** - O sistema deverá permitir de alguma forma visualizar os percursos gerados, isto é, visualmente ou textualmente;
- **R13** - O sistema deverá permitir visualizar estatísticas referentes ao serviço dos percursos simulados (e.g. distância total percorrida, média de tempos de espera dos utilizadores, tempos de viagem do utilizador, etc.);

Quanto aos requisitos não funcionais foram formulados os seguintes:

- **R14** - O sistema deverá conseguir dar resposta dentro de 1 minuto a um novo pedido dinâmico que surja;
- **R15** - O sistema deverá conseguir inserir um novo pedido (dinâmico) num percurso já existente dentro de 1 minuto;
- **R16** – O sistema deverá continuar com a simulação, sendo tratada alguma exceção em caso de falha.

4 Desenvolvimento

Neste capítulo será brevemente descrito todo o processo de modelação e implementação da simulação de eventos discretos, que reflita o contexto de um sistema de transporte flexível. Para além disto, serão abordados aspetos sobre a implementação do modelo de acordo com o problema formulado anteriormente (em Especificação do Problema), e do *Solver* que será utilizado para resolver esse mesmo problema, tendo em conta o conjunto de pedidos que vão surgindo ao longo da simulação.

4.1 Simulação

Como já referido o objetivo da simulação é conseguir simular virtualmente um contexto (aproximadamente) real de um sistema de transporte flexível. De forma a conseguir modelar a simulação de um sistema deste tipo é primeiro necessário compreender como é que um serviço de transporte público convencional é efetuado. A prestação de um serviço de transporte público é semelhante à prestação de um serviço de transporte flexível, no sentido em que existe um ou mais veículos que têm um conjunto de paragens que terão de visitar, sendo estas paragens já conhecidas antes de esse mesmo veículo as visitar. A grande diferença entre estes dois tipos de sistema é que, no sistema transporte público convencional, a procura só é conhecida quando o veículo chega à paragem, sendo o percurso fixo, e tendo que passar por paragens ou locais onde poderão não existir clientes para serem "entregues" ou "recolhidos". Relativamente ao sistema de transporte flexível, os pedidos de serviço já são conhecidos anteriormente à visita a cada paragem, tendo o percurso do veículo sido construído considerando os pedidos e as restrições temporais que estes implicam, não existindo assim a necessidade de passar por paragens desnecessárias, e permitindo ainda a reotimazação dos percursos com o surgimento de novos pedidos. Assim, embora um percurso de transporte flexível seja continuamente adaptado com o surgimento de novos pedidos, a verdade é que o veículo saberá sempre qual a próxima paragem a ser visitada no momento de saída da paragem anterior.

Após um estudo e análise do contexto deste problema, foi possível formular um conjunto de eventos discretos necessários para o processo de simulação de um serviço de transporte flexível. O objetivo deste estudo era compreender quais os eventos que influenciam ou alteram a prestação do serviço de transporte e também como é que este serviço é prestado de forma a que seja possível simulá-lo. Com este estudo, foi então, possível compreender que existem três intervenientes principais na prestação de um serviço de transporte: a paragem, o cliente e o veículo em questão. Note-se, no entanto, que o motorista do veículo poderia também ser considerado um interveniente, no entanto no modelo de simulação desenvolvido, o mesmo foi omitido, por motivos de simplificação, embora este esteja de certa forma relacionado com a prestação do serviço, pois é este que irá dirigir o veículo. Para além do mais, o condutor acabará também por não ter grande influência na maneira como o serviço será construído e executado no problema do DARP formulado anteriormente, uma vez que este não tem em consideração aspetos relativos ao mesmo. Deste ponto de vista, o veículo possui a informação necessária para representar e simular a prestação do serviço.

Assim, com a análise efetuada do contexto do problema, foi possível formular os seguintes eventos essenciais necessários para a simulação de um serviço ou percurso de transporte flexível:

- **Evento 1: Evento de chegada de um veículo a uma paragem** - Chegada de um veículo v a uma determinada paragem p;
- **Evento 2: Evento de partida de um veículo de uma paragem** - Partida de um veículo v de uma determinada paragem p;
- **Evento 3: Evento de entrada de um cliente num veículo** - Entrada de um cliente c num determinado veículo v, ocorrendo numa determinada paragem p;
- **Evento 4: Evento de saída de um cliente num veículo** - Saída de um cliente c de um determinado veículo v, ocorrendo numa determinada paragem p;
- **Evento 5: Evento de surgimento de um novo pedido de serviço** - Surgimento de um novo pedido de serviço dinâmico para um cliente c.

Estes eventos discretos, acima apresentados são tratados de diferentes maneiras, dependendo do tipo de evento em questão, contudo, qualquer um destes tem um conjunto de informações comuns como:

- **Tempo** - indica o tempo em que o evento ocorre;
- **Categoria** – identifica o tipo de evento que ocorre;
- **Etiqueta de tratamento** – identifica se o evento já foi tratado ou não.

Para além destas informações, dependendo do tipo de evento em questão, cada evento poderá ainda conter algumas informações adicionais que irão ser utilizadas para auxiliar no seu tratamento, que serão referidas posteriormente na especificação do tratamento de cada um dos diferentes eventos.

É importante salientar que sempre que é gerado um novo evento, este é inserido numa lista ordenada de eventos, que contem todos os eventos a serem tratados. No que diz respeito à ordenação dos mesmos, esta é efetuada por ordem crescente do tempo de ocorrência de cada evento presente nessa mesma lista. Notemos ainda que, quando um evento é tratado, dependendo do evento em questão, poderá também ser necessário, nessa altura gerar os eventos seguintes a serem tratados. Relativamente à ordem de tratamento de eventos, esta será efetuada ao percorrer sequencialmente a lista de eventos ordenada temporalmente.

4.1.1 Modelo de dados da Simulação

Antes de falar dos diferentes eventos e dos tratamentos dos mesmos em mais detalhe, primeiro é importante compreender o modelo de dados das diferentes entidades constituintes da simulação, isto é, o Cliente, a Paragem e o Veículo. Para além disto, neste subcapítulo, ainda serão abordados os objetos de dados que contêm informações importantes acerca do estado, ou configuração da simulação. Notemos ainda que, nem todas as informações contidas em cada um destes objetos poderão estar representadas, pelo que serão apenas referidas as consideradas mais relevantes para a compreensão dos mesmos.

4.1.1.1 Paragem

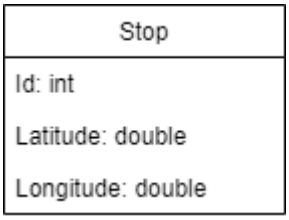

Figura 1 - Objeto de dados de uma Paragem (Stop)

O objeto de dados de uma paragem (*Stop*) é responsável por conter um conjunto de informações acerca da mesma incluindo:

- **Id** – identificador único que identifica a paragem a que o objeto se refere;
- **Latitude** – coordenada geográfica da latitude da paragem;
- **Longitude** – coordenada geográfica da longitude da paragem.

4.1.1.2 Cliente

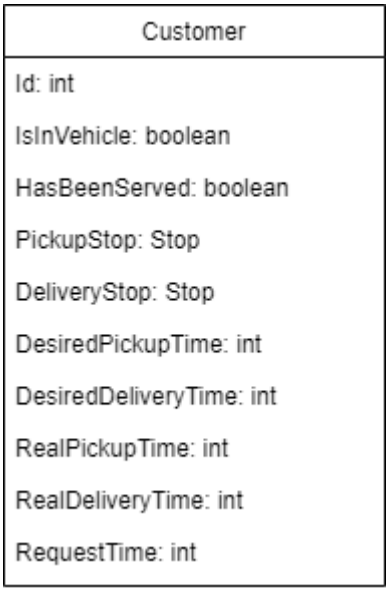

Figura 2 - Objeto de dados de um Cliente (Customer)

O objeto de dados de um cliente (*Customer*) serve para representar as informações relativas ao mesmo e ao seu pedido de serviço, sendo que este inclui as seguintes informações:

- **Id** – identificador único que identifica o cliente a que o objeto se refere;
- **IsInVehicle** – booleano que indica se o cliente se encontra atualmente dentro de um veículo. O cliente encontra-se no estado “dentro de um veículo”, se este for verdadeiro ou no estado “fora de um veículo” se for falso;
- **HasBeenServed** – booleano que indica se o pedido de serviço do cliente já foi servido. Se este for verdadeiro, o pedido de serviço já foi servido, caso contrário, indica que o mesmo ainda não foi servido;
- **PickupStop -** identifica a paragem de “recolha” (ou origem) do cliente, na qual este pretende ser “recolhido”;

- **DeliveryStop** – identifica a paragem de “entrega” (ou destino) do cliente, na qual este pretende ser “entregue”;
- **DesiredPickupTime** – indica o tempo desejado pelo cliente para ser “recolhido” na paragem de “recolha”;
- **DesiredDeliveryTime** – indica o tempo desejado pelo cliente para chegar ao seu destino (paragem de “entrega”);
- **RealPickupTime** – indica o tempo (real) ocorrido na simulação, no qual o cliente foi “recolhido” na paragem origem por um dado veículo;
- **RealDeliveryTime** – indica o tempo (real) ocorrido na simulação, no qual o cliente foi “entregue” na paragem destino por um dado veículo;
- **RequestTime** – indica o tempo em que foi efetuado o pedido de serviço do cliente.

4.1.1.3 Veículo

Figura 3 - Objeto de dados um veículo (Vehicle)

O objeto de dados de um veículo (*Vehicle*) demonstra as informações relativas ao mesmo e ao estado do seu serviço. Este objeto inclui as seguintes informações:

- **Id** – Identificador único que identifica o veículo;
- **AverageSpeed** – velocidade média a que o veículo se desloca (e.g. 40km/h);
- **Capacity** – a capacidade máxima de clientes suportada pelo veículo;
- **IsFull** – booleano que indica que um veículo se encontra cheio para um determinado momento (se for verdadeiro);

- **IsIdle –** booleano que indica se um veículo está parado ou em movimento. Um veículo está em movimento se este for falso, caso contrário, o veículo encontra-se parado;
- **CurrentLoad –** indica o número de clientes que se encontram dentro do veículo atualmente;
- **Route** – contém o conjunto de todas as paragens que devem ser visitadas (percurso) pelo veículo de forma a servir um conjunto de clientes;
- **CustomersInside -** contém o conjunto de todos os clientes que se encontram atualmente dentro do veículo;
- **CustomersToBeServed** - contém o conjunto de todos os clientes que já foram aceites previamente para serem servidos pelo veículo, mas que ainda não estão dentro do mesmo a ser servidos atualmente;
- **ExpectedTimeWindows –** contém o conjunto dos intervalos de tempo esperados para o veículo chegar e partir de cada paragem. Note-se que estes intervalos de tempo são dados pelo *Solver* ao resolver um problema de *routing* e estes são apenas uma estimativa dos tempos de chegada e partida de cada paragem, pelo que serão diferentes em relação aos reais;
- **RealTimeWindows –** contém o conjunto dos intervalos de tempo reais da simulação, de chegada e partida do veículo de cada paragem. Estes serão os intervalos de tempo que indicam em que momento na simulação o veículo chegou e partiu de uma dada paragem, sendo que estes geralmente serão diferentes e superiores em relação aos intervalos de tempo esperados (*ExpectedTimeWindows*);
- **CurrentStop** – identifica a paragem em que o veículo se encontra atualmente;
- **NextStop –** identifica a próxima paragem a ser visitada pelo veículo;
- **StartStop –** identifica a paragem inicial de onde o veículo parte inicialmente no seu percurso;
- **EndStop** – identifica a paragem final para onde o veículo retorna no final do seu percurso;

A Figura 4 representa o diagrama que permite descrever o estado de um veículo tendo em conta a ocorrência de diferentes eventos.

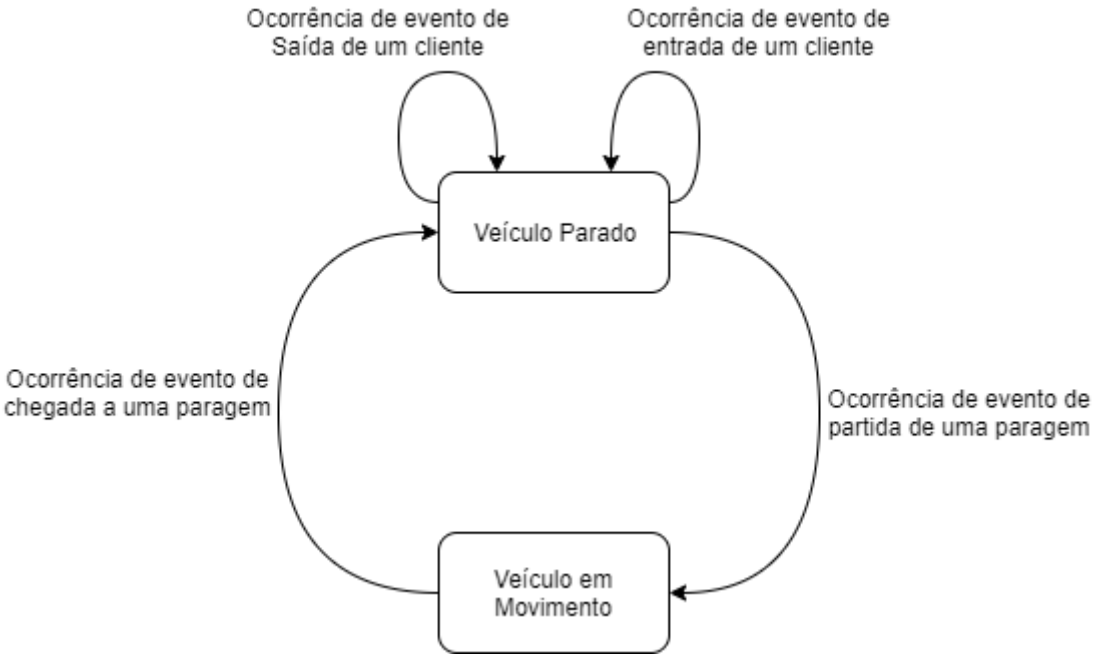

Figura 4 - Diagrama de estados de um veículo tendo em conta a ocorrência de diferentes eventos

Através da Figura 4, é possível perceber que um veículo encontrar-se-á no estado "parado", após a ocorrência de um evento de chegada a uma paragem e antes da ocorrência de um evento de partida de uma paragem. O veículo irá estar no estado "em movimento" imediatamente após a ocorrência de um evento de partida de uma paragem e antes da ocorrência do evento de chegada a uma paragem. Para além disto, ainda é possível ver que os eventos de entrada de um cliente ou saída de um cliente numa paragem, só podem ocorrer no momento em que um veículo se encontra "parado" numa dada paragem. Note-se que estes eventos estarão descritos em mais detalhe no próximo subcapítulo.

4.1.1.4 Contexto de simulação

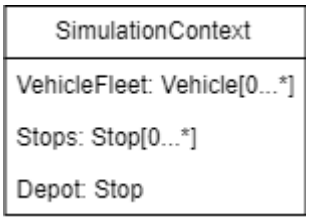

Figura 5 - Objeto de dados do contexto da Simulação (SimulationContext)

O objeto de dados do contexto da simulação (*SimulationContext*), como o nome indica, é responsável por armazenar um conjunto de informações relativa ao contexto atual em que a simulação se encontra. Este objeto inclui as seguintes informações:

- **VehicleFleet** – contém o conjunto de todos os objetos veículos da simulação;
- **Stops** – contém o conjunto de todas as paragens consideradas para a simulação, nas quais poderão surgir um novo pedido de serviço;

- **Depot** – paragem base onde todos os veículos começam e terminam o seu percurso, na qual esta será atribuída como paragem inicial (*StartStop*) e de fim de percurso (*EndStop*) para cada um dos veículos contidos na simulação.

A partir do conjunto de veículos (*VehicleFleet*) é possível saber a situação atual da simulação uma vez que cada objeto veículo *(Vehicle*) irá conter as informações relativas ao estado atual do mesmo e do seu serviço como paragens a ser visitadas, a paragem atual em que o mesmo se encontra, clientes dentro do mesmo, clientes a serem "recolhidos", etc.

É ainda importante salientar que para o problema em questão foi considerada a mesma paragem de começo e fim de percurso para cada um dos veículos, sendo esta a paragem *Depot* contida no contexto da simulação. Assim, todos os veículos, inicialmente partem da paragem *Depot* e retornam à mesma no final do seu percurso. No entanto, poderia ter sido considerada uma outra abordagem na qual as paragens de começo poderiam ser distintas, ou na qual as mesmas pudessem variar de veículo para veículo.

4.1.1.5 Parâmetros de simulação

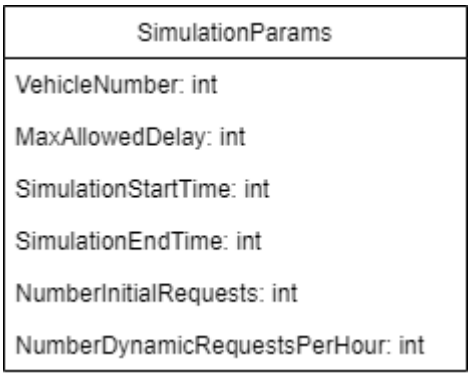

Figura 6 - Objeto de dados dos parâmetros da Simulação (SimulationParams)

Este objeto é responsável por conter algumas informações relativas à configuração da simulação, incluindo:

- **VehicleNumber** – número de veículos que serão utilizados na simulação;
- **MaxAllowedDelay** – tempo de atraso máximo que poderá ser utilizado pelo *Solver* caso não seja possível encontrar uma solução, de forma a relaxar os intervalos de tempo do problema e de modo a tornar a resolução do mesmo possível. Note-se que este tempo será abordado em mais detalhe posteriormente quando forem abordados aspetos relevantes acerca do *Solver*;
- **MaxRelativeCustomerRideTime** – tempo de viagem máximo relativo dos clientes, que será utilizado pelo *Solver* na imposição da restrição de tempo de viagem dos clientes. Este tempo também será abordado em mais detalhe posteriormente.;
- **SimulationStartTime** – tempo de começo da simulação;
- **SimulationEndTime** – tempo de fim da simulação;
- **NumberInitialRequests –** número de pedidos estáticos que serão gerados e conhecidos no início da simulação;
- **NumberDynamicRequestsPerHour** – Número de pedidos dinâmicos que serão gerados para cada hora de simulação.

4.1.2 Eventos da simulação

Agora será abordado em mais detalhe a modelação e implementação do tratamento cada um dos diferentes eventos discretos modelados para a simulação.

4.1.2.1 Evento de entrada de um cliente num veículo

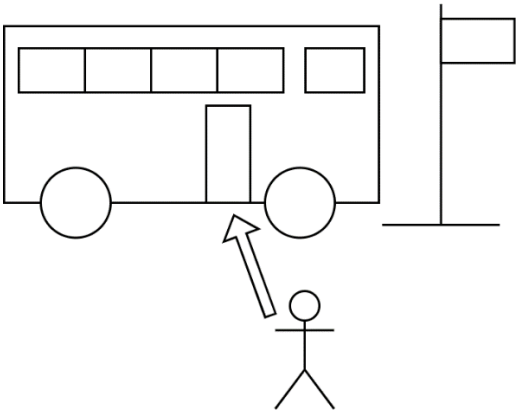

Figura 7 – Representação do evento de entrada de um cliente num veículo

Estes são os eventos que permitem simular a entrada de um cliente num veículo e numa determinada paragem. Este evento é constituído pelas seguintes informações adicionais:

- **Cliente (*Customer*)** – identifica o objeto cliente que entra no veículo;
- **Veículo (*Vehicle*)** – identifica o objeto veículo no qual o cliente entra;
- **Paragem (*Stop*)** – identifica o objeto paragem no qual este evento ocorre.

O tratamento de um evento de entrada de um cliente num veículo, passa por verificar se existe espaço suficiente no veículo para o mesmo entrar. Caso isto se verifique, o cliente será inserido à lista de clientes do veículo, no momento estabelecido pelo tempo do evento, sendo assim atualizada a quantidade atual de clientes (*CurrentLoad*) que o mesmo transporta de momento (aumentando-a por 1). Se o veículo estiver cheio o cliente será recusado e consequentemente não entrará no mesmo, ou seja, este não será inserido à lista de clientes que se encontram veículo (*CustomersInsideVehicle*). Quando um cliente entra num veículo é ainda registado o momento da simulação na qual o cliente entrou no veículo, sendo assim atualizada a informação do tempo real de entrada (*RealPickupTime*) contida no objeto cliente (*Customer*), para o tempo do evento de entrada que está a ser tratado atualmente.

4.1.2.2 Evento de saída de um cliente de um veículo

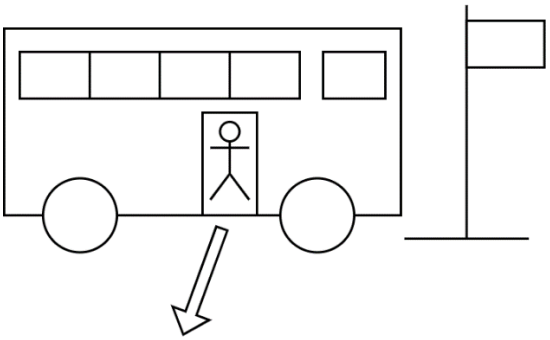

Figura 8 - Representação do evento de saída de um cliente de um veículo

Como o seu nome indica, estes são os eventos que simulam a saída de um cliente de um determinado veículo ocorrendo numa determinada paragem. As informações adicionais que este implica são muito semelhantes aos eventos de entrada de um cliente num veículo, nas quais estas incluem:

- **Cliente (*Customer*)** – identifica o objeto cliente que saí do veículo;
- **Veículo (*Vehicle*)** – identifica o objeto veículo do qual o cliente irá sair;
- **Paragem (*Stop*)** – identifica o objeto paragem no qual este evento ocorre.

O tratamento deste tipo de evento consiste em remover o cliente da lista de clientes que se encontram no veículo (*CustomersInside*) no momento indicado pelo tempo do evento, sendo então atualizada a carga atual (*CurrentLoad*) do veículo (diminuindo-a por 1) de forma a expressar a saída do mesmo. O estado do serviço do cliente é também atualizado de forma a indicar que o mesmo já foi servido (*HasBeenServed* = verdadeiro) e ainda é guardado o tempo de ocorrência deste evento como o tempo real de entrega do cliente (*RealDeliveryTime*).

É ainda importante referir que um evento de saída de um cliente c de um dado veículo v, só poderá ocorrer se já tiver ocorrido anteriormente um evento de entrada de c em v numa paragem anteriormente visitada pelo mesmo, isto porque não faz muito sentido um cliente sair de um certo veículo se este não entrou no mesmo anteriormente. Assim sendo, a geração dos eventos de entrada e saída de clientes deve conter mecanismos que assegurem esta dependência, na qual a entrada de um cliente numa dada paragem deve ocorrer primeiro que as saídas desse mesmo cliente numa outra paragem, sendo esta visitada posteriormente à paragem de entrada do mesmo. Note-se que, esta dependência será garantida na geração dos eventos de saída e entrada de clientes, os quais serão gerados no tratamento dos eventos de chegada de um veículo a uma paragem, que serão abordados de seguida.

4.1.2.3 Evento de chegada de um veículo a uma paragem

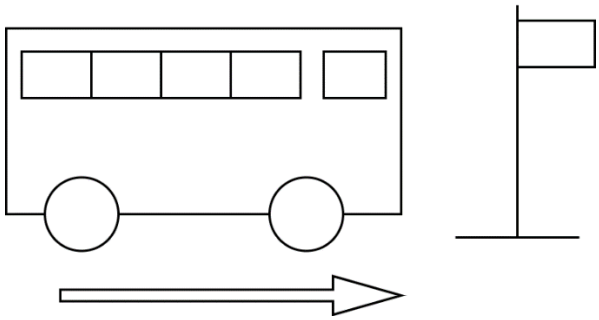

Figura 9 – Representação do evento de chegada de um veículo a uma paragem

Os eventos de chegada de um veículo a uma paragem, como o nome indica, são eventos que ocorrem de forma a simular a chegada de um veículo a uma determinada paragem. Para além das informações base que os eventos contêm já referidas este tipo de evento contém ainda um conjunto de informações adicionais relativas ao mesmo:

- **Veículo (*Vehicle*)** – identifica o objeto veículo que chega a uma dada paragem;
- **Paragem (*Stop*)** – identifica o objeto paragem no qual o veículo chega;

O tratamento deste tipo de evento consiste em atualizar o estado atual do veículo, indicando que o mesmo se encontra no estado "parado", ou seja, *IsIdle* é passado para verdadeiro. Depois é verificado se existem clientes para sair na paragem atual, ou seja, é verificado se existem clientes na lista de clientes dentro do veículo (*CustomersInside*) com a paragem atual como paragem de "entrega".

Caso existam clientes para sair na paragem atual do veículo em questão, serão então gerados os eventos que reflitam essa saída (evento de saída de um cliente de um veículo) para cada um desses clientes, no qual o tempo que será atribuído a esses eventos (de saída) terá que ser calculado. Primeiro, para cada cliente é obtido um tempo de saída, gerado aleatoriamente dentro de um intervalo de tempo (e.g. gerar um valor aleatório entre 1 e 5 segundos), de forma a simular o tempo que o mesmo demora a sair do veículo. Depois, é calculado o valor do tempo máximo atribuído, sendo este obtido ao calcular o tempo máximo entre todos os eventos já gerados para o veículo e paragem do evento atual em questão, isto é, para o veículo e paragem atual do evento de chegada que está a ser tratado atualmente. Note-se que, o tempo máximo atribuído inicialmente, quando ainda não foi gerado nenhum evento para além do atual que está a ser tratado (de chegada de um veículo a uma paragem) será o tempo deste mesmo evento, ou seja, o tempo de chegada do veículo à paragem atual. O cálculo do tempo máximo atribuído pode ser resumido na equação:

$$t_{ma}(v,i) = \max\{T_e(v,i)\}, \qquad (v,i) \in \mathbb{N}, T_e(v,i) = \{t_1, t_2, \dots t_n\}, t \in T_e : t \geq t_c(v,i)$$

Equação 1 - Tempo máximo atribuído aos eventos gerados para um veículo v numa paragem i

Onde $t_{ma}(v,i)$ é o tempo máximo atribuído para uma paragem i e um veículo v, max é a função máximo que obtém o valor máximo entre um conjunto de valores, $T_e(v,i)$ contém o conjunto de todos os tempos de ocorrência dos eventos que foram gerados até ao momento para o veículo v e que ocorrem na paragem atual i. $t_c(v,i)$ tempo do evento de chegada do veículo v à paragem i, que está a ser tratado atualmente.

Após saber o tempo máximo atribuído torna-se então possível, calcular o tempo que será atribuído ao evento de saída para um dado cliente, no qual este será a soma entre o tempo de saída atribuído ao cliente e o tempo máximo atribuído aos eventos do veículo e paragem atual. É possível ainda perceber que o tempo que será atribuído a esse evento de saída de um cliente será sempre superior em relação ao tempo do evento atual de chegada de um veículo a uma paragem que está a ser tratado. A Equação 2 demonstra o cálculo do tempo que será atribuído a um evento de saída de um cliente que será gerado:

$$t_s(v,i,j) = t_{ma}(v,i) + t_{saida}(j), \qquad (v,i,j) \in \mathbb{N}, t_{ma}(v,i) \geq t_c(v,i)$$

Equação 2 - Tempo atribuído a um evento de saída de um cliente j do veículo v, na paragem i

Onde $t_s(v,i,j)$ é o tempo atribuído ao evento de saída do cliente j do veículo v na paragem i, $t_{ma}(v,i)$ é o tempo máximo atribuído para a paragem i e o veículo v obtido através da Equação 1 e $t_{saida}(j)$ é o tempo que o cliente j demora a sair do veículo. $t_c(v,i)$ é o tempo do evento atual que está a ser tratado, de chegada do veículo v à paragem i.

Obviamente, caso não existam clientes na lista de clientes que se encontram dentro do veículo (*CustomersInside*) com a paragem atual como paragem "entrega", não serão gerados eventos de saída de clientes do veículo atual e na paragem atual.

Após isto, é ainda verificado se existem clientes na lista de clientes que serão servidos pelo veículo (*CustomersToBeServed*), para entrarem na paragem atual, ou seja, se existem clientes cuja paragem de "recolha" é a paragem indicada no evento de chegada de um veículo a uma paragem, que está a ser tratado. Caso existam, serão então gerados os eventos que reflitam essa entrada (evento de entrada de um cliente num veículo), sendo também necessário calcular o tempo que será atribuído ao evento de entrada para cada um dos clientes. Este tempo será a soma entre o tempo de entrada do cliente para o qual o evento está a ser gerado e o tempo máximo atribuído a todos os eventos para a paragem e veículo do evento atual. Relativamente ao tempo de entrada de um cliente, este é semelhante ao tempo de saída de um cliente já falado anteriormente, sendo este também um valor aleatoriamente gerado dentro de um intervalo, de forma a simular o tempo que um dado cliente demora a entrar num veículo. No que toca ao tempo máximo atribuído, este é também calculado da mesma forma que na geração dos eventos de saída de clientes, utilizando a Equação 1. Assim, a Equação 3 demonstra a obtenção do tempo de um evento de entrada de um cliente j num dado veículo v e numa determinada paragem i:

$$t_e(v,i,j) = t_{ma}(v,i) + t_{entrada}(j), \qquad (v,i,j) \in \mathbb{N}, t_{ma}(v,i) \geq t_c(v,i)$$

Equação 3 - Tempo atribuído a um evento de entrada de um cliente j no veículo v, na paragem i

Onde $t_e(v,i,j)$ é o tempo atribuído ao evento de entrada do cliente j do veículo v na paragem i, $t_{ma}(v,i)$ é o tempo máximo atribuído para a paragem i e o veículo v calculado usando a Equação 1, $t_{entrada}(j)$ é o tempo que o cliente j demora a entrar no veículo e $t_c(v,i)$ é o tempo do evento atual que está a ser tratado, de chegada do veículo v à paragem i.

No que diz respeito ao caso de não existirem clientes para entrar na paragem atual, obviamente que não serão gerados eventos de entrada de clientes para a paragem atual.

Por fim, é ainda verificado, se existem, mais paragens a ser visitadas pelo veículo do evento que está a ser tratado, ou seja se existe *NextStop*, caso isto seja verificado será então

gerado o evento de partida do veículo da paragem atual. Sendo o tempo atribuído para a ocorrência deste evento, sempre superior ao tempo do evento de chegada do veículo atual à paragem atual e ainda superior a qualquer um dos tempos de entrada e saída de clientes na paragem atual, isto, caso estes tenham sido gerados nos passos anteriores, de forma a garantir coerência temporal na ocorrência dos diferentes eventos. O cálculo do tempo que será atribuído a este evento será a soma entre o tempo máximo atribuído para a paragem e veículo atual usando a Equação 1, e um tempo de atraso, que é o tempo que o veículo ficará à espera na paragem, após a ocorrência do último evento dessa paragem presente na lista de eventos. A Equação 4 reflete o cálculo do tempo do evento de partida da paragem que será gerado:

$$t_p(v,i) = \mathrm{t_{ma}}(v,i) + t_a(v,i), \qquad (v,i) \in \mathbb{N}, t_{ma} \geq t_c(v,i)$$

Equação 4 - Tempo atribuído a um evento de partida de um veículo v da paragem i

Onde $t_p(v,i)$ é o tempo que será atribuído ao evento que será gerado de partida do veículo v da paragem i, $t_{ma}(v,i)$ é o tempo máximo atribuído para a paragem i e o veículo v utilizando a Equação 1, $t_a(v,i)$ é o tempo de atraso que o veículo v ficará á espera na paragem i e $t_c(v,i)$ é o tempo do evento atual que está a ser tratado, de chegada do veículo v à paragem i.

Assim com tudo o que foi aqui descrito é possível perceber que a ordem de ocorrência de eventos de um veículo v que chega a uma dada paragem i (evento de chegada de um veículo a uma paragem) será a seguinte:

1. Ocorrência de um evento de chegada do veículo v à paragem i;
2. Ocorrência de zero ou mais eventos de saída de clientes do veículo v na paragem i (conforme o número de clientes existentes no veículo v para saírem na paragem i);
3. Ocorrência de zero ou mais eventos de entrada de clientes no veículo v na paragem i (conforme o número de clientes existentes na paragem i para entrarem no veículo v);
4. Ocorrência de um ou zero eventos de partida do veículo v da paragem i (conforme se existe próxima paragem, seguinte a i, a ser visitada pelo veículo v ou não, respetivamente).

Com a descrição do tratamento dos eventos de chegada de um veículo a uma paragem, é possível perceber que os eventos de saída de um cliente de um veículo têm prioridade sobre os eventos de entrada no mesmo, nos quais se ambos existirem para uma dada paragem e um dado veículo, os de saída irão sempre ocorrer primeiro que os de entrada, pois o tempo de ocorrência gerado dos eventos de saída será sempre inferior aos eventos de entrada para um dado veículo e numa dada paragem. É importante referir que esta medida foi adotada de forma a facilitar o controlo da capacidade dos veículos de forma a permitir maximizar o número de clientes servidos.

Para melhor compreender o porquê da utilização desta abordagem, consideremos o seguinte caso onde os eventos de entrada são tratados primeiro ou alternadamente com os de saída. Isto pode levar a que ocorram determinadas situações, em que eventos de entrada de certos clientes sejam recusados, pelo facto de o veículo estar cheio atualmente e ainda não ter sido tratado o evento de saída de um dos clientes que irá sair na paragem atual e ter sido primeiro tratado o evento de entrada de um cliente nessa mesma paragem. No entanto, é possível resolver esta situação, ao serem atribuídos tempos de ocorrência aos eventos de saída

inferiores aos tempos de eventos de entrada para um dado veículo e numa dada paragem, levando a que os mesmos sejam tratados primeiros e permitindo assim aumentar o número de clientes servidos. Note-se que a abordagem aqui tomada não é única, pelo que podem existir outras abordagens e possivelmente melhores para a resolução deste mesmo problema.

4.1.2.4 Evento de partida de um veículo de uma paragem

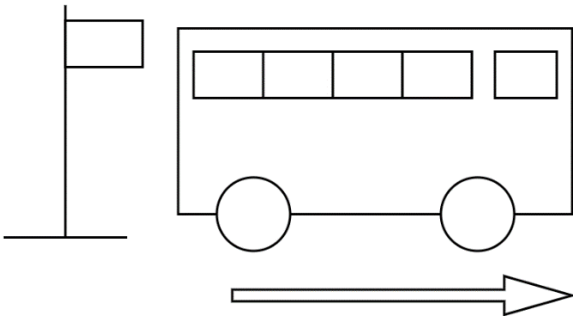

Figura 10 – Representação do evento de partida de um veículo de uma paragem

Como o seu nome indica, os eventos de partida de um veículo de uma paragem são eventos que ocorrem de forma a simular a partida de um veículo de uma determinada paragem. Semelhante ao evento anterior, este tipo de evento, também contêm as seguintes informações adicionais:

- **Veículo (*Vehicle*)** – identifica o objeto veículo que parte de uma paragem;
- **Paragem (*Stop*)** – identifica o objeto paragem do qual o veículo parte;

O tratamento deste veículo resume-se a atualizar o estado atual do veículo de forma a que reflita que este está em movimento (*IsIdle* é alterado para falso) e atualizar a paragem atual (*CurrentStop)* que será a próxima paragem a ser visitada do percurso (*Route*) do mesmo, sendo isto efetuado no momento estipulado pelo tempo do evento. É também no tratamento deste tipo de evento que será gerado o próximo evento de chegada à próxima paragem a ser visitada pelo veículo, no qual será necessário efetuar um cálculo do tempo de chegada à próxima paragem de forma a obter o tempo para esse evento que será gerado.

Este tempo de chegada é calculado tendo em conta distância entre a paragem do qual o veículo parte e a próxima paragem a ser visitada pelo mesmo utilizando a Equação 10, e sendo esta depois convertida para tempo de deslocamento utilizando essa distância e a velocidade média (*AverageSpeed*) a que o veículo se desloca através da Equação 11. Obtido o tempo de deslocamento, basta somá-lo ao tempo base atual do evento de partida da paragem atual.

$$t_c(v,i,j) = t_p(v,i) + td(i,j)\,, \qquad v,i,j \in \mathbb{N}$$

Equação 5 - Tempo de chegada à próxima paragem j tendo em conta o evento atual de partida da paragem i

Onde $t_c(v,i,j)$ é o tempo de chegada do veículo v à paragem j vindo da paragem i, $t_p(v,i)$ é o tempo de partida do veículo v da paragem i anterior a j do evento (de partida de um veículo de uma paragem) e $td(i,j)$ é o tempo de deslocamento entre a paragem i e a paragem j obtido através da Equação 11.

Note-se que, para existir a ocorrência de um evento de partida de um veículo de uma dada paragem, é primeiro necessário ocorrer um evento de chegada desse mesmo veículo a essa mesma paragem, existindo, então uma dependência entre estes dois tipos de eventos. Isto

porque de forma geral, não faz grande sentido um veículo partir de uma dada paragem antes de este ter se quer chegado à mesma, anteriormente. No entanto, isto será sempre assegurado pela forma como são tratados os eventos de chegada de um veículo a uma paragem, nos quais os tempos de ocorrência dos eventos de partida serão sempre superiores em relação ao tempo de ocorrência do evento de chegada desse veículo a essa mesma paragem.

4.1.2.5 Eventos de surgimento de novo pedido de serviço

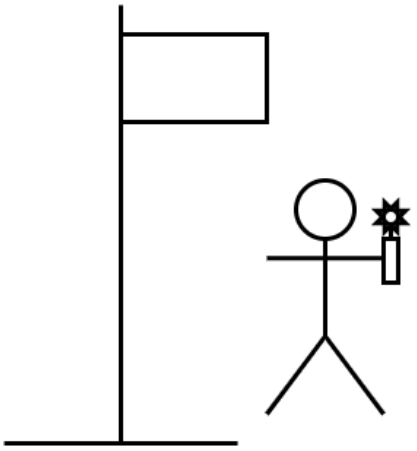

Figura 11 – Representação do evento de surgimento de novo pedido de serviço

Como o seu nome indica estes são os eventos que ocorrem de forma a simular a ocorrência ou surgimento de um novo pedido (dinâmico) de serviço. Este evento é caracterizado por conter a seguinte informação adicional: Cliente (*Customer*) – objeto que identifica o cliente que efetua o pedido de serviço, contendo as informações relativas ao mesmo.

No tratamento deste evento, é necessário obter o contexto atual da simulação para que possa ser utilizado o *Solver* para resolver o problema atual de *routing* de forma a compreender, se é possível atribuir esse novo pedido de serviço a algum dos veículos da simulação. Assim, são obtidos todos os veículos considerados para a simulação, isto é, todos os veículos contidos na lista de veículos (*VehicleFleet*) do contexto da simulação, sendo que estes podem ou não estar a realizar um percurso (serviço) atualmente, conforme, respetivamente, se os mesmos têm paragens que necessitam de ser visitadas no seu percurso atual ou se estão parados atualmente na sua paragem de partida inicial (*StartStop*). Para cada veículo são obtidos todos os clientes atribuídos ao mesmo, isto é, todos os clientes que se encontram dentro do veículo (*CustomersInside*), como também todos os clientes que foram atribuídos ao mesmo, mas que ainda não se encontram dentro do veículo (*CustomersToBeServed*). Para além disto, é também obtido ainda, para cada veículo, o tempo de chegada à paragem atual. Caso o veículo já esteja na paragem atual, este tempo será o tempo do evento atual de surgimento de novo pedido; caso o veículo esteja a se deslocar para a paragem atual, será o tempo previsto de chegada à paragem para a qual o mesmo se desloca.

Depois de conhecidos todos os clientes para cada veículo, estes são agrupados numa única lista (conjunto de todos os clientes atuais da simulação), juntamente com o novo cliente que surgiu com o evento atual (de surgimento de novo pedido). Assim, a partir do que foi aqui

descrito é agora possível obter todas as informações iniciais necessárias para construção do modelo de dados que será utilizado pelo *Solver* para resolver o problema de *routing*, de forma servir o novo pedido e os pedidos anteriormente aceites. Assim, é passado então ao modelo de dados do *Solver* o conjunto de veículos contidos no contexto atual da simulação (*VehicleFleet*), os tempos de chegada à paragem atual para cada um dos veículos, como também todos os clientes a serem considerados para o serviço, isto é, os clientes já previamente atribuídos aos diferentes veículos, como também o novo cliente e, por fim, o parâmetro de simulação do tempo de viagem relativo de um cliente (*MaxRelativeCustomerRideTime*). Note-se que a construção do modelo de dados do *Solver*, e as informações que o mesmo requer, estarão descritas em mais detalhe na secção 4.2.1.

Depois da construção do modelo de dados a partir das informações referidas, o *Solver* irá tentar resolver o problema de *routing* contido no mesmo (que reflete o contexto atual da simulação) dentro de um tempo limite (e.g. 60 segundos). Este limite é imposto de forma estabelecer um tempo máximo de procura para o *Solver* encontrar soluções, que não seja muito demorosa para o cliente (simulando um contexto real). Caso não seja possível resolver o problema, o pedido correspondente ao evento atual, será recusado, sendo mantidos os percursos atuais dos veículos da simulação. Na situação em que o *Solver* encontra uma solução, no contexto atual da simulação, este cliente será obviamente aceite, tornando-se então necessário atualizar os percursos dos veículos considerados para o problema. A partir da solução obtida pelo *Solver*, é atualizado (se necessário) para cada veículo o seu conjunto de paragens que constituem o seu percurso (*Route*), como também os intervalos de tempo previstos para cada uma dessas paragens (*ExpectedTimeWindows*). Para além disto, são também atualizadas as informações dos clientes a serem servidos (*CustomersToBeServed*), de forma a refletir a inclusão do novo pedido. Saliente-se ainda que, ambos estes conjuntos são atualizados a partir da paragem atual em que o veículo se encontra.

É importante referir que com a ocorrência deste tipo de evento (de surgimento de novo pedido), podem por vezes existir veículos (contidos no contexto da simulação) que não estejam a ser utilizados atualmente, mas que com a geração da nova solução (percurso), tendo em conta esse novo pedido, seja necessário que os mesmos entrem em serviço. Para estes veículos, torna-se necessário inicializar o primeiro evento de chegada desse veículo a uma paragem, na qual essa primeira paragem será a paragem inicial do percurso atribuído ao mesmo e o tempo de chegada será o tempo de chegada a essa mesma paragem inicial contido na lista de intervalos de tempos esperados (*ExpectedTimeWindows*) desse veículo. Isto é efetuado de forma a inicializar a cadeia de eventos para um dado veículo que se encontre inativo (ou fora de serviço), uma vez que a partir da ocorrência do primeiro evento de chegada de um veículo será possível ir gerando os eventos seguintes tendo em conta o evento que está a ser tratado atualmente.

Note-se que todos os eventos deste tipo (de surgimento de novo pedido de serviço) serão gerados no início da simulação, sendo que este processo estará melhor descrito no próximo subcapítulo.

4.1.3 Inicio da simulação

Com o início da simulação são construídos e inicializados todos os objetos necessários, tais como, os parâmetros de simulação e os objetos contidos no contexto da simulação (conjunto de veículos (*VehicleFleet*) e o conjunto de clientes conhecidos inicialmente). Será

também nesta altura que serão também gerados alguns eventos iniciais de chegada de um veículo a uma paragem, de forma a inicializar os primeiros eventos a serem tratados para cada veículo. Além disso, são também gerados todos os eventos de surgimento de novo pedido serviço.

No que toca ao conjunto de veículos (*VehicleFleet*), são construídos os objetos de todos os veículos (*Vehicles*) que serão utilizados na simulação, sendo que o seu número é dado pelo parâmetro da simulação *VehicleNumber*. Para cada veículo é atribuído a sua paragem de começo (*StartStop*) e fim de percurso (*EndStop*), sendo esta a paragem *Depot* contida no objeto do contexto da simulação. Para além disto, é ainda atribuído a cada veículo uma hora de começo para o seu serviço, este tempo serve para indicar que o momento em que o veículo estará disponível para partir da sua paragem inicial.

Relativamente ao conjunto de pedidos, serão gerados todos os clientes cujos pedidos são conhecidos inicialmente (pedidos estáticos). O número destes clientes será dado pelo parâmetro da simulação *NumberInitialRequests*. Para cada um destes clientes, o tempo do pedido (*RequestTime*) corresponde ao tempo inicial da simulação. No que respeita às paragens de "recolha" e "entrega", estas são escolhidas aleatoriamente dentro do conjunto de paragens (*Stops*) existentes no contexto da simulação. Os tempos de "recolha" e "entrega" são também, gerados aleatoriamente. Note-se que, as paragens de "recolha" e "entrega" para um dado cliente nunca podem ser iguais, pois não faz sentido um cliente requisitar serviço para ser "entregue" na paragem em que foi "recolhido". Relativamente ao tempo de "recolha" de um dado cliente, este estará dentro do intervalo de tempo do período da simulação, isto é, entre o tempo estipulado pelo parâmetro de simulação *SimulationStart* e o tempo estipulado pelo parâmetro de simulação *SimulationEnd*, sendo que a atribuição deste tempo pode ser resumida na equação:

$$t_r(j) = random(SimulationStart, SimulationEnd)$$

Equação 6 - Cálculo do tempo de "recolha" um dado cliente j

Onde $t_r(j)$ é o tempo de "recolha" do cliente j, $random$ é uma função que escolhe um valor aleatoriamente dentro de um intervalo, $SimulationStart$ é o tempo de começo da simulação, $SimulationEnd$ é o tempo de fim da simulação.

O tempo de "entrega" de um dado cliente terá um valor sempre superior ao tempo de "recolha" desse mesmo cliente. Os tempos de "entrega" são geralmente atribuídos aleatoriamente dentro de um intervalo de tempo após o tempo de "recolha" atribuído (e.g. tempo gerado aleatoriamente entre 10 a 60 minutos após o tempo de "recolha" atribuído). E esta atribuição pode ser resumida na Equação 7:

$$t_e(j) = random(t_r(j) + t_{min}, t_r(j) + t_{max})$$

Equação 7 - Cálculo do tempo de "entrega" de um dado cliente j

Onde $t_e(j)$ é o tempo de "entrega" do cliente j, $random$ é uma função que escolhe um valor aleatoriamente dentro de um intervalo, $t_r(j)$ é o tempo de "recolha" obtido anteriormente do cliente j, t_{min} é um valor de tempo mínimo em relação ao tempo de "recolha" e t_{max} é um valor de tempo máximo em relação ao tempo de "recolha". Note-se que, $t_{min} < t_{max}$ e $t_{min} > 0$.

Após obter os clientes e veículos já referidos, é então possível construir o modelo de dados, que reflete o contexto inicial da simulação, a ser resolvido pelo *Solver*. Este modelo é contruído ao ser passado ao mesmo, o conjunto de veículos contidos no contexto atual da simulação (*VehicleFleet*), os tempos de chegada à paragem atual para cada um dos veículos (geralmente será 0 inicialmente), o conjunto de todos os clientes iniciais da simulação e ainda, por fim, o parâmetro de simulação do tempo máximo de viagem relativo de um cliente. Como já referido anteriormente a construção deste modelo de dados (do *Solver*) estará descrita em mais detalhe posteriormente.

Depois de construído o modelo de dados, é então possível que o *Solver* da *Google* procure e encontre soluções que resolvam o problema de *routing* que reflete o contexto inicial da simulação. Esta procura geralmente decorre durante um determinado tempo (e.g. 2 minutos), de forma a tentar encontrar melhores soluções. Após a obtenção de uma solução, serão atribuídos aos veículos indicados pelo *Solver* (que tenham percursos atribuídos), um conjunto de paragens (*Route*), como também um conjunto de intervalos de tempo para estar em cada uma dessas paragens (*ExpectedTimeWindows*) e ainda os clientes que serão servidos por cada um desses veículos (*CustomersToBeServed*). Para cada um desses veículos, a sua paragem atual (*CurrentStop*) será a paragem inicial de cada uma dessas listas de paragens (*Route*), ou seja, a paragem *Depot* contida no contexto da simulação, pois como já referido, todos os veículos começam os seus percursos inicialmente nessa paragem.

O passo seguinte para a inicialização da simulação passa por gerar o primeiro evento de chegada a uma paragem para cada um dos veículos que tiverem um percurso atribuído pelo *Solver*. Para cada veículo, a paragem atual do veículo será a registada no parâmetro *Depot*, e o tempo do evento será o valor presente no tempo de chegada essa mesma paragem presente na lista *ExpectedTimeWindows* desse veículo.

Para além disto, ainda são gerados todos os eventos de surgimento de novo pedido de serviços que irão ocorrer na simulação. Para cada hora de simulação o número de clientes (dinâmicos) que serão gerados será igual ao parâmetro de simulação *NumberDynamicRequestsPerHour*. A cada um destes clientes é atribuída uma paragem de "entrega" e "recolha", sendo estas paragens escolhidas aleatoriamente dentro do conjunto de paragens (*Stops*) do contexto da simulação. Semelhante às paragens atribuídas aos clientes conhecidos inicialmente (estáticos), para um dado cliente, estas paragens também têm que ser distintas, isto é, a paragem de "recolha" de um dado cliente não pode ser a mesma paragem de "entrega" para esse mesmo cliente. Para além disto, é ainda atribuído a cada cliente um tempo de efetuação do pedido (*RequestTime*), este tempo serve para indicar quando é o evento de surgimento de novo pedido a ser tratado. O seu valor será gerado aleatoriamente dentro de um intervalo de tempo que reflete o início e fim da hora para o qual o cliente está a ser gerado. A obtenção do tempo do pedido para um dado cliente pode ser resumida na Equação 8:

$$t_{request}(j) = random(HourStartTime, HourEndTime)$$

Equação 8 - Tempo de efetuação pedido para um cliente j para uma da hora de simulação

Onde $t_{request}(j)$ é o tempo de efetuação do pedido do cliente j, $random$ é uma função que escolhe um valor aleatoriamente dentro de um intervalo, $HourStartTime$ é o tempo mínimo do intervalo de tempo que indica o inicio da hora de simulação a ser considerada e

$HourEndTime$ é o tempo máximo do intervalo de tempo que indica o fim da hora de simulação a ser considerada.

O tempo de "recolha" de um dado cliente será atribuído aleatoriamente dentro de um intervalo de tempo que reflete o tempo de efetuação do pedido previamente atribuído e o tempo de efetuação desse mesmo pedido somado com um valor de tempo máximo (e.g. 15mins), como demonstrado na Equação 9:

$$t_r(j) = random\big(t_{request}(j), t_{request}(j) + t_{max}\big), \qquad t_{max} \geq 0$$

Equação 9 – Cálculo do tempo de "recolha" de um cliente j para uma dada hora da simulação

Onde $t_r(j)$ é o tempo de "recolha" do cliente j gerado para uma dada hora, $random$ é uma função que escolhe um valor aleatoriamente dentro de um intervalo, $t_{request}(j)$ é o tempo de efetuação do pedido do cliente j calculado usando a Equação 8, t_{max} é o valor do tempo máximo em relação ao tempo do pedido.

O tempo de "entrega" é aleatoriamente atribuído dentro de um intervalo de tempo após o tempo de "recolha" atribuído anteriormente através da Equação 9, sendo esta atribuição efetuada do mesmo modo que o tempo de "entrega" para os clientes estáticos (Equação 7).

Note-se que, o valor do tempo máximo em relação ao tempo de efetuação do pedido foi utilizado de forma a permitir simular diferentes situações reais. Isto é, um cliente pode requisitar um certo pedido de serviço num determinado momento, mas o tempo desejado para ser "recolhido" poderá nem sempre ser o mesmo em que o pedido é efetuado. Caso tal fosse possível, o cliente que requisita o serviço teria de estar na paragem de "recolha" e pronto para ser recolhido no momento em que foi efetuado o pedido de serviço. Isto num contexto real poderá não ser sempre verdade, podendo existir situações nas quais, por vezes, um serviço possa ser requisitado com alguma antecedência (e.g. 5 ou 10 minutos antes do tempo de "recolha" pretendido pelo cliente) devido à necessidade do cliente se deslocar para o ponto de "recolha" ou por qualquer outro fator que influencie o tempo no qual o mesmo consiga estar presente na paragem de "recolha". Assim, o tempo máximo em relação ao tempo de efetuação do pedido é utilizado de forma a ser possível também representar essas mesmas situações, permitindo assim melhor aproximar a simulação à realidade e dando assim uma certa flexibilidade à mesma.

Após obter cada cliente (dinâmico) e as respetivas informações, referidas anteriormente, é possível então gerar cada evento de surgimento de novo pedido de serviço para cada um desses clientes. O tempo de ocorrência para cada um desses eventos (de surgimento de novo pedido dinâmico) será o tempo de efetuação do pedido (*RequestTime*) do cliente a que esse pedido se refere.

Após tudo isto é possível então começar a simulação uma vez que já foram gerados os eventos iniciais de chegada de um veículo a uma paragem como também todos os eventos de surgimento de novo pedido dinâmico que terão que ser tratados ao longo da simulação.

4.2 Google OR-Tools

4.2.1 Modelo de Dados do *Solver*

De forma a conseguir utilizar a biblioteca *OR-Tools* para resolver o problema do DARP especificado em 3.1, é primeiro necessário construir um modelo de dados (do *Solver*), que conterá um conjunto de informações necessárias para definir o problema de *routing* a ser tratado (resolvido) pelo *Solver*. Este modelo irá permitir refletir o contexto do problema atual que é necessário resolver (e.g. o contexto atual da simulação), permitindo que o *Solver* procure soluções para o mesmo. O processo para a construção das estruturas de dados constituintes deste modelo será agora referido.

4.2.1.1 IndexManager

O primeiro passo para a construção de um modelo de dados que reflete o contexto atual da simulação é a construção de um objeto chamado de *IndexManager*. Este objeto será responsável por mapear os diferentes objetos (veículo, cliente e paragens) da simulação, para índices que os identificam nas estruturas de dados consideradas. Estes índices irão permitir representar os diferentes objetos da simulação, tanto no modelo de dados, como também no *Solver*, uma vez que o modelo de dados será passado a este. Para além disto, este objeto também irá permitir mapear cada índice que representa um dado objeto para o objeto real que o mesmo representa. É importante referir que a implementação deste objeto deve-se ao tipo de estruturas manipuladas pelo *Solver* da *Google*, que implica que a representação das diferentes entidades constituintes de um problema (clientes, veículos e paragens) contidas no modelo de dados deve ser efetuada utilizando índices (números inteiros).

O *IndexManager* necessita de receber um conjunto de dados, sendo que estes geralmente irão advir do contexto atual da simulação, nos quais os mesmos incluem as seguintes informações:

- **O conjunto de veículos (*Vehicles*)** – uma lista que contém todos os objetos veículo (*Vehicle*), não repetidos e disponíveis atualmente para realizar serviço, e informações referentes aos mesmos, que poderão ser utilizados para efetuar a realização do conjunto serviços que, posteriormente, serão delegados pelo *Solver*. Note-se que, na simulação, o conjunto que será passado como esta informação será o conjunto de veículos (*VehicleFleet*) contidos no contexto da mesma;
- **O conjunto que contém as horas de começo de percurso para cada veículo (*StartDepotArrivalTimes*)** – lista que contém a hora de começo de serviço para cada um dos presentes no conjunto de veículos. Note-se que o índice desta lista permite identificar a hora de começo para o veículo que tenha esse mesmo índice na lista de veículos (*Vehicles*). Na simulação, para um dado veículo, é obtido o seu tempo de começo de serviço, caso este se encontre parado, será o momento em que é requisitada a otimização dos percursos dos veículos que está a ser atualmente tratada. Caso o veículo se encontre em movimento, este tempo será o momento previsto para o veículo chegar à próxima paragem;
- **O conjunto dos clientes a serem considerados (*Customers*)** – uma lista que contém todos os objetos cliente (*Customer*), não repetidos, que serão considerados para o problema de *routing* atual, podendo estes estarem em dois estados distintos: “dentro

do veículo", se o cliente já estiver a ser servido, ou "fora do veículo", se o cliente ainda não tiver sido "recolhido" por nenhum veículo. Na simulação este conjunto é obtido ao agrupar todos os clientes que estão ou serão servidos por cada um dos veículos contidos no contexto, como também ao adicionar os novos clientes que surgem, por exemplo, ao ocorrer um evento de surgimento de novo pedido;

- **Tempo máximo de viagem relativo de um cliente** – constante que define o tempo máximo relativo de um cliente (e.g. 30min), na qual esta será utilizada na imposição da restrição do tempo máximo de viagem de um cliente. Na simulação esta constante é geralmente obtida através do parâmetro de simulação do tempo de atraso máximo (*MaxAllowedDelay*).

O *IndexManager* será constituído por três listas principais que irão permitir obter os índices que representam cada objeto como também obter o objeto que cada índice representa:

- **Lista de Veículos (*Vehicles*)** – contem o conjunto de todos veículos considerados para um dado problema de *routing.* Esta lista nunca conterá veículos repetidos*;*
- **Lista de Cliente (*Customers*)** – contem o conjunto de todos clientes considerados para um dado problema de *routing*. Esta lista nunca conterá clientes repetidos;
- **Lista de Paragens (*Stops*)** – contem o conjunto de todas as paragens consideradas para um dado problema de *routing*. É construída tendo em conta as paragens de começo e fim de percurso de cada veículo como também as paragens de "entrega" e "recolha" de cada cliente presente no problema. Note-se que esta lista pode conter paragens repetidas;

Será agora descrito como é que é efetuado o mapeamento entre os conjuntos dos diferentes objetos da simulação (veículo, cliente e paragem) para índices que os representam.

O mapeamento entre a lista de veículos (*Vehicles*) e clientes (*Customers*) recebidos como entrada para listas de índices, que permitem representar esses objetos, é efetuado utilizando indexação única, na qual cada um dos objetos é representado por um único número inteiro (índice). O índice que representa um dado cliente ou veículo, será o índice que esse objeto ocupa na sua respetiva lista recebida inicialmente (*Customers* ou *Vehicles*).

A Figura 12, permite melhor compreender o mapeamento efetuado entre os clientes e veículos contidos na lista de clientes (*Customers*) e veículos (*Vehicles*), respetivamente, para índices que os representam, para um caso no qual são considerados três clientes e três veículos.

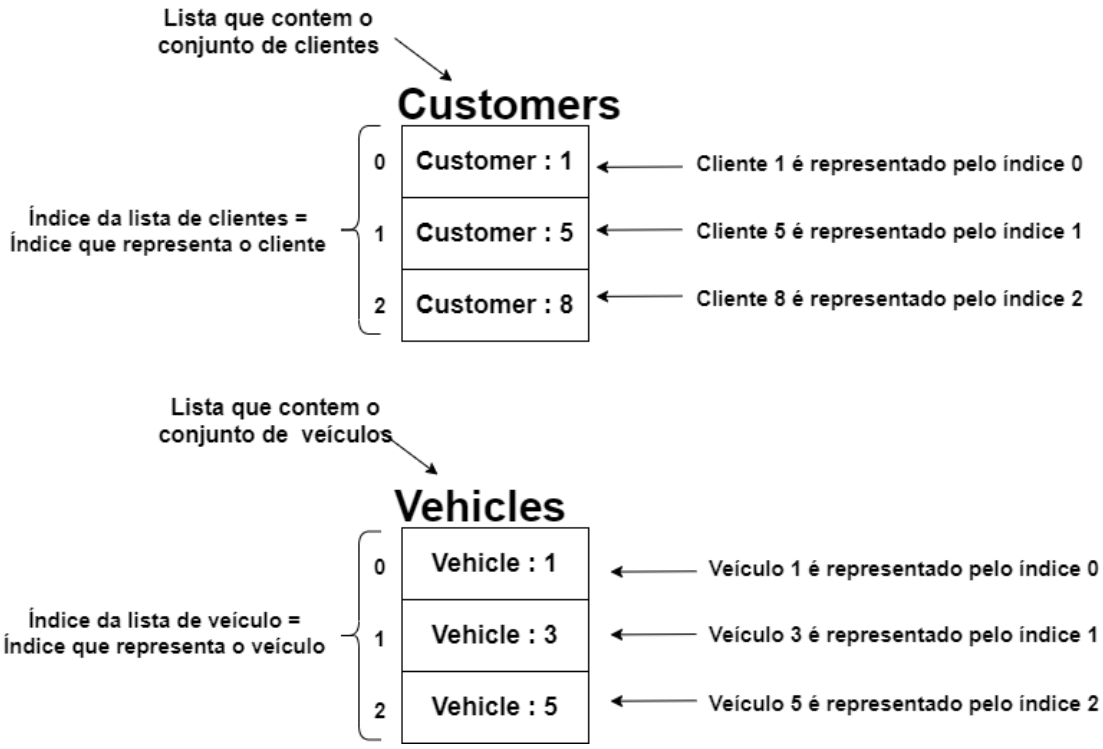

Figura 12 - Mapeamento entre lista de objetos dos clientes e veículos para lista que contem os índices que representam esses objetos, para três veículos e três clientes utilizando a indexação única

Por exemplo, se observarmos a lista de clientes (*Customers*) é possível perceber que o cliente 1 (*Customer 1*) ocupa o índice 0. Semelhantemente, os índices que representam os clientes 5 e 8, são os índices 1 e 2 respetivamente, sendo estes as posições que ambos ocupam na lista de clientes (*Customers*). O mesmo pode também ser dito para a representação dos veículos na qual foi tomada esta mesma abordagem.

Com estes exemplos, é possível verificar que o índice que irá representar cada objeto veículo ou cliente presente no *IndexManager*, será o respetivo índice que o mesmo ocupa na lista de veículos (*Vehicles*) ou clientes (*Customers*), respetivamente.

Embora já tenhamos as listas de clientes, ainda não conhecemos todas as paragens que serão necessárias visitar, existindo então a necessidade de construir uma lista de paragens (*Stops*) que as englobe. O primeiro passo para construir a lista de paragens, passa por criar duas listas: uma que irá conter as paragens de início de percurso (*StartsList*) de onde cada veículo irá partir no início do seu percurso, e uma outra lista que conterá as paragens de fim de percurso (*EndsList*) para onde cada veículo irá retornar no final do seu percurso. Podendo cada uma destas paragens estar associada a um ou mais veículos.

Para um dado veículo, a sua informação da paragem de começo do percurso atual será inserida na lista das paragens de início (*StartsList*), na posição em que esse veículo ocupa na lista de veículos recebida inicialmente. É importante referir que a paragem de começo de percurso deum dado veículo será a paragem atual (*CurrentStop*), para um dado contexto da simulação. Note-se que, foi utilizada a paragem atual (*CurrentStop*) em que cada veículo se encontra e não

a paragem de partida de cada veículo (*StartStop*), de forma a ser possível resolver tanto os casos de *routing* estático, como é o caso do contexto inicial da simulação, no qual a paragem atual para um dado veículo será a paragem de inicio de percurso (*StartStop*) contida no objeto do mesmo, como também para os casos dinâmicos, no qual é necessário efetuar reotimização de um dado percurso a partir da paragem atual em que esse veículo se encontra, como é o caso de quando surge um novo pedido dinâmico.

Semelhante à inserção das paragens iniciais, a informação da paragem de término de percurso para cada um dos veículos considerados, será então adicionada à lista de paragens de termino (*EndsList*), na posição em que esse veículo ocupa na lista de veículos recebida inicialmente. Note-se que, a paragem de fim de percurso de um dado veículo será a informação contida no mesmo da paragem de fim de percurso (*EndStop*).

A Figura 13 permite melhor compreender a construção de ambas estas listas aqui referidas (das paragens de começo e fim de percurso), para um exemplo em que são considerados dois veículos (veículo 1 e veículo 3). Note-se que as cores representam de que veículo é que vem a informação.

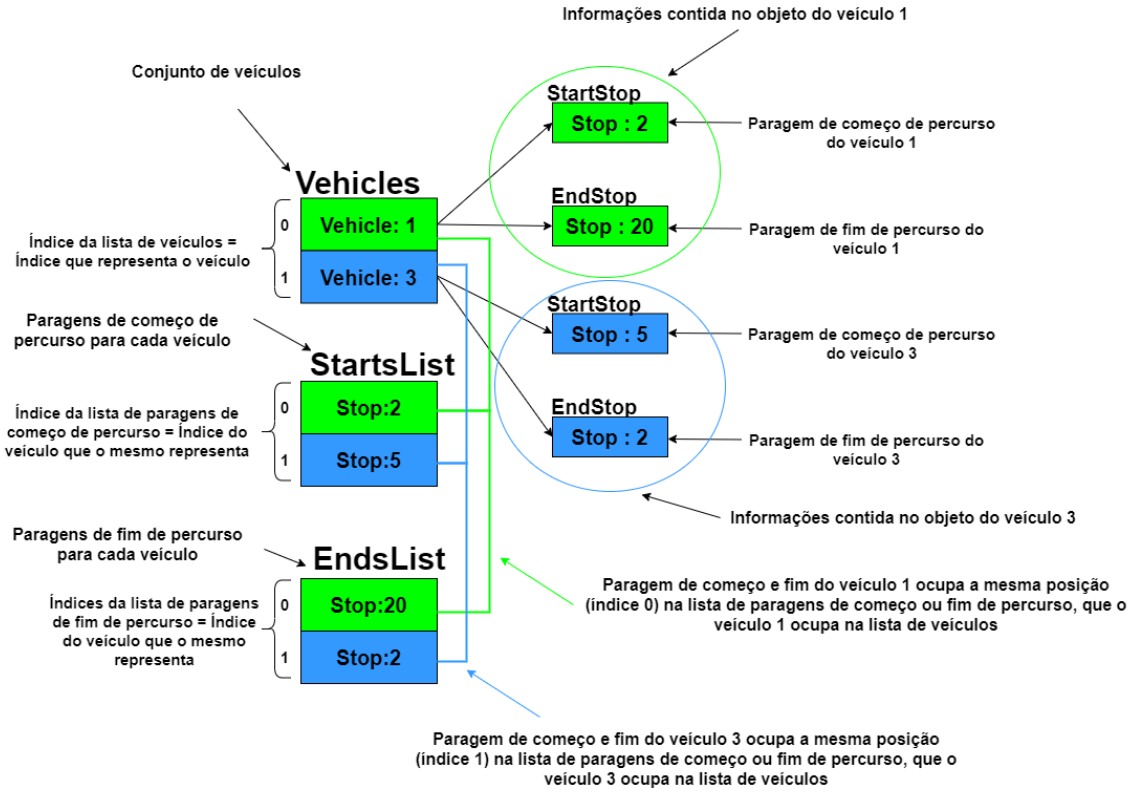

Figura 13 - Mapeamento entre o conjunto de veículos e suas paragens de começo e fim de percurso para a lista de paragens de começo e fim de percurso

Com este exemplo é possível constatar que a posição que cada paragem inicial ou de fim de percurso de cada veículo ocupará na sua respetiva lista, será o índice que o veículo ocupa na lista de veículos, como já referido anteriormente. Por exemplo, o veículo 3 ocupa a posição 1 da lista de veículos (*Vehicles*) e sendo este representado pela cor azul, assim, a paragem de começo ou fim de percurso referente ao mesmo irá também ocupar essa mesma posição (1) na sua respetiva lista, como é possível constatar pela cor azul presente nessa posição em ambas as listas das paragens de começo (*StartsList*) e fim de percurso (*StopsList*).

Depois de obtidas as paragens de começo e fim de percurso, é depois construída a lista *Stops*, que irá conter conjunto de todas as paragens consideradas para o problema de *routing* em questão. Esta lista é construída sequencialmente sendo primeiro adicionadas as já faladas paragens de início de percurso e, posteriormente, adicionadas as paragens de fim de percurso para cada veículo considerado e terminando com a adição das paragens de "entrega" e "recolha" para cada um dos clientes considerados. Como já referido, um cliente pode se encontrar em dois estados distintos, dependendo do momento em que é necessário construir uma nova solução para o problema de *routing* em questão, sendo que este pode estar no estado "dentro do veículo" (a ser servido) ou "fora do veículo" (à espera para ser servido). Para o primeiro caso, quando existe um cliente no estado "dentro do veículo", torna-se apenas necessário adicionar a paragem de "entrega" do mesmo à lista de paragens (*Stops*), retirando assim, a necessidade de adicionar a paragem de "recolha" do mesmo. Isto devesse ao facto de a paragem de "recolha" já ter sido visitada anteriormente, pois o cliente já foi "recolhido" anteriormente, encontrando-se assim dentro de um dado veículo. No segundo caso, quando um cliente encontra-se no estado "fora do veículo", ou seja à espera para ser servido, são adicionadas ambas as paragens (de "recolha" e "entrega") do respetivo cliente à lista de paragens (*Stops*), pois ambas terão que ser visitadas.

É ainda importante referir que mesmo que existam clientes ou veículos que partilhem paragens repetidas, estas serão na mesma adicionadas à lista de paragens (*Stops*). Esta medida foi tomada de forma a permitir utilizar a múltipla indexação, na qual cada paragem de começo ou fim de percurso de cada veículo como também de "entrega" e "recolha" de cada cliente irá ser representada por um único índice distinto que a permite identificar. Este método para representação das paragens (múltipla indexação), faz com que nenhum cliente ou veículo partilhe o mesmo índice para as suas paragens consideradas, sendo então possível representar cada uma das paragens associadas aos mesmos por um único índice que a identifica e que esse índice, por sua vez, também permite de certa forma relacionar essa paragem com o veículo ou cliente que a mesma está associada. Isto também leva a que uma paragem real (objeto) possa ser representada por mais do que um índice distinto, isto se existirem veículos ou clientes que partilhem de alguma forma a mesma paragem real. Notemos ainda, que o índice que representa cada uma das paragens presentes no *IndexManager* será o índice que essa paragem ocupa na lista de paragens (*Stops*) construída.

A Figura 14 demonstra um exemplo de construção da lista de paragens (*Stops*) como também mapeamento que é efetuado entre essa lista e o índice que representa essa paragem para uma situação na qual é considerada um único veículo e dois clientes para um problema de *routing* estático. As cores permitem melhor compreender de que objeto e informação relativa ao mesmo, cada paragem presente na lista de paragens origina.

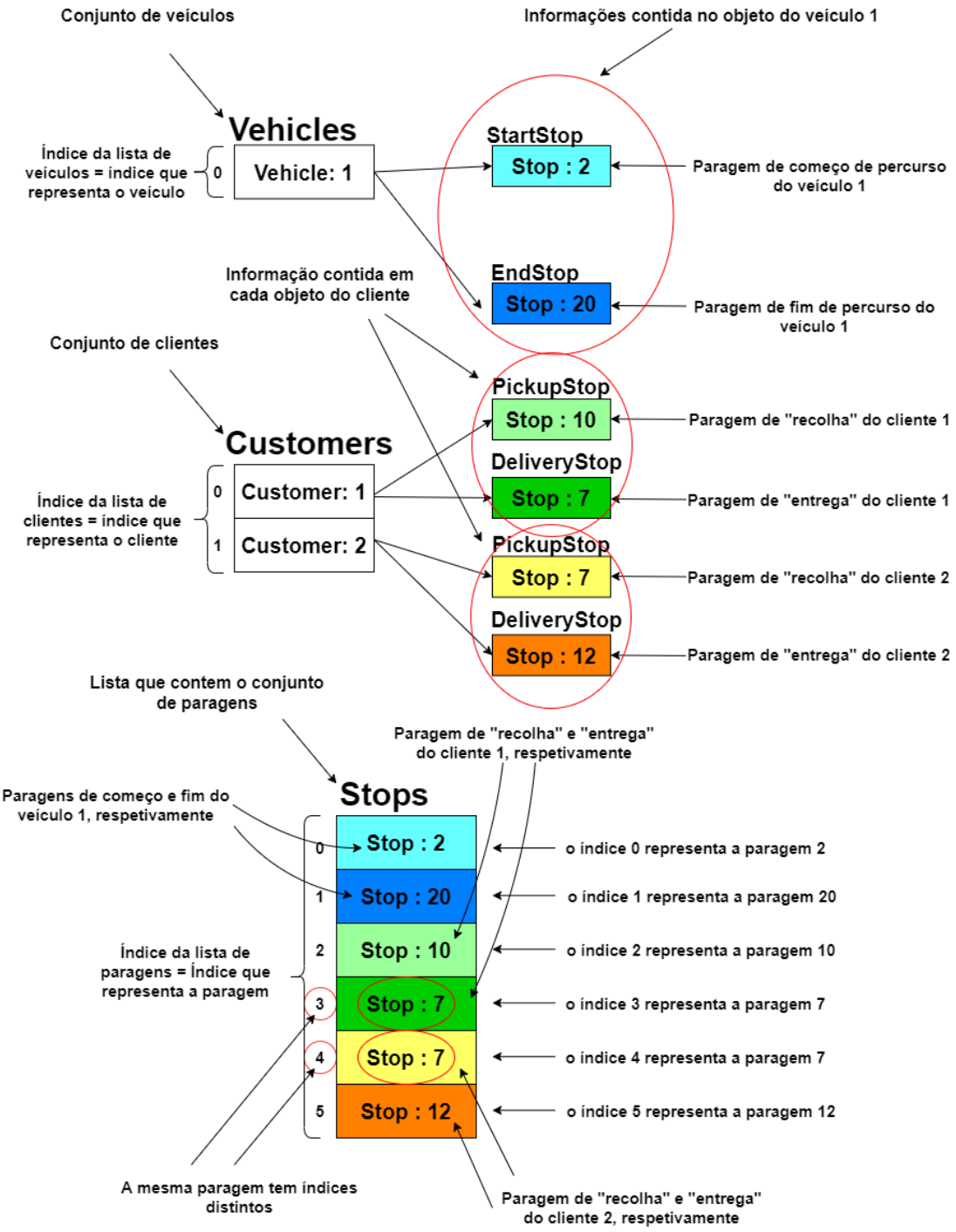

Figura 14 - Exemplo de construção da lista de paragens e mapeamento da mesma para a lista que contem os índices das paragens, considerando um único veículo e dois clientes para um problema de routing estático

Com este exemplo é possível verificar que a lista de paragens (*Stops*) é construída primeiro adicionando as paragens de começo de percurso dos veículos, sendo posteriormente adicionadas as paragens fim de percurso dos veículos e por fim sendo inseridas as paragens de "entrega" e "recolha" dos clientes. Também é possível ver que o índice que representa uma dada paragem será o índice que a mesma ocupa na lista de paragens (*Stops*).

É ainda possível ver que o cliente 1 e o cliente 2 partilham a mesma paragem, a paragem 7, na qual esta é a paragem de "entrega" para o cliente 1 e a paragem de "recolha" para o cliente 2. Ao olharmos para os índices da lista de paragens é possível perceber que esta paragem é

representada por dois índices distintos, pois a lista pode conter paragens repetidas, sendo estes índices o índice 3 (para o cliente 1) e índice 4 (para o cliente 2), sendo assim evidente a utilização da múltipla indexação e estando também cada um desses índices, associado a um dos clientes, como é possível ver pelas cores utilizadas para representação das informações.

Por fim, é importante referir que a razão da utilização da múltipla indexação na lista das paragens deve-se de forma a ultrapassar alguns problemas encontrados na representação das paragens utilizada pelo *Solver* da *Google*. Nomeadamente, a utilização da indexação única levava a que por vezes não fosse possível resolver certos problemas de *routing*, nos quais existem vários clientes que partilhem paragens que contenham o mesmo índice. Para além disto, a utilização da indexação única, pode levar a que em certos casos, sejam de certa forma, "perdidas" algumas informações que constituem o modelo de dados para o problema. Sendo que um exemplo destes casos será demonstrado em mais detalhe, quando for falado do vetor de procura (Exemplo 1 e Exemplo 2).

Os próximos subcapítulos irão demonstrar como é que são construídas as diferentes estruturas de dados que compõe o modelo de dados que será utilizado pelo *Solver*. É importante referir que o *IndexManager* irá ter um papel bastante ativo na construção destas estruturas de dados, pois estas serão construídas utilizando os índices que representam os diferentes objetos contidos neste mesmo objeto.

4.2.1.2 Paragens de começo, término e capacidade dos veículos

Após obter o *IndexManager* são construídos dois vetores, um que contém os índices das paragens iniciais (*Starts*) e outro que contém os índices das paragens de termino (*Ends*) do percurso de cada veículo. Cada célula destes vetores contém o índice que representa a paragem inicial ou de termino em questão, respetivamente, e no qual o índice da posição do vetor irá representar o índice do veículo em questão. O acesso a ambos estes vetores são semelhantes à lista de paragens de começo (*StartsList*) e fim de percurso (*EndList*) contida no *IndexManager*, sendo a grande diferença o facto de uma dada posição do vetor que representa a paragem de começo ou fim de um dado veículo, conter o índice que representa essa paragem, em vez de conter o objeto paragem.

Depois, é ainda construído um outro vetor que contém as capacidades dos veículos chamado *de VehicleCapacities* no qual cada célula do mesmo contém o valor da capacidade de clientes máxima suportada pelo veículo representado pelo índice dessa posição. Note-se que a capacidade de cada veículo é obtida no seu respetivo objeto (*Vehicle*) no qual o mesmo tem um atributo (*Capacity*) que a indica.

Com isto, é possível compreender que o acesso a qualquer um destes vetores é semelhante, no sentido em que o índice de cada posição do vetor representa o índice do veículo que o mesmo representa e o valor contido na célula dessa posição representa o respetivo valor para o vetor correspondente, no qual é seguido o mesmo tipo de indexação do *IndexManager.* Consequentemente, estes três vetores (*Starts, Ends* e *VehicleCapacities*) também terão o mesmo tamanho, no qual este será o número de veículos distintos considerados.

A Figura 15, permite melhor compreender a utilização destes três vetores para uma situação em que são considerados quatro veículos:

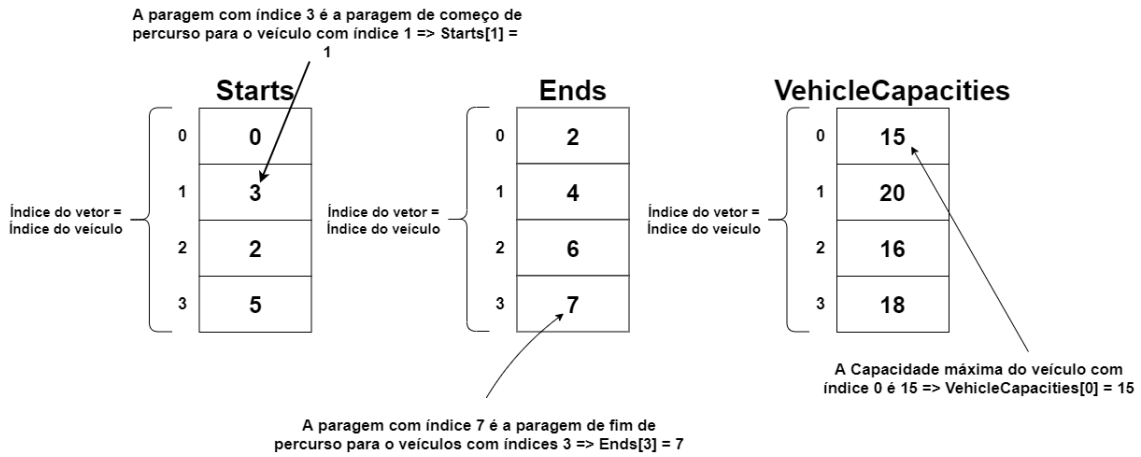

Figura 15 - Exemplo de utilização para os vetores: Starts, Ends, VehicleCapacities para quatro veículos

Como podemos ver para acedermos à informação relativa a qualquer um dos veículos basta aceder ao índice do vetor correspondente que o representa. Por exemplo, caso queiramos saber a capacidade do veículo com índice 2, esta será 16 ($VehicleCapacities[2] = 16$). A paragem de começo para esse mesmo veículo será a paragem com índice 2 ($Starts[2] = 2$) e a paragem de fim, a paragem com índice 0 ($Ends[2] = 6$).

4.2.1.3 Tempos de deslocamento entre cada paragem

A matriz de tempos de deslocamento (*TavelTimes*), contém o tempo de deslocamento (em segundos) entre todas as paragens consideradas para o problema. Esta matriz é construída tendo como base a lista de paragens (*Stops*) contida no *IndexManager* e seguindo também o mesmo tipo de indexação. Consequentemente, o tamanho desta matriz será o tamanho dessa mesma lista de paragens (*Stops*) pelo tamanho da mesma. O primeiro passo para construir esta matriz passa por calcular a distância mínima entre todas as paragens contidas na lista *Stops*, incluindo as próprias (que será 0).

É importante referir que a distância mínima entre as diferentes paragens foi calculada no sistema de medição métrico, utilizando a fórmula de *Haversine* e utilizando as coordenadas de latitude e longitude das paragens consideradas, presentes nos dados disponibilizados pelos Horários do Funchal, permitindo então, representar cada paragem como pontos numa superfície esférica. A Equação 10 representa a fórmula de *Haversine*:

$$d = 2r \arcsin\left(\sqrt{sin^2\left(\frac{latXr - latYr}{2}\right) + \cos(latYr) * cost(latXr) * sin^2\left(\frac{lonXr - lonYr}{2}\right)}\right)$$

Equação 10 - Distância de Haversine

Onde d é a distância em metros, r é o raio, sendo neste caso o raio da terra (r = 6371000 metros), $latYr$ e $latXr$ é a latitude em radianos para as paragens Y e X, respetivamente, e $lonYr$ e $lonXr$ a longitude em radianos para as paragens Y e X, respetivamente.

Esta fórmula permite calcular a distância mínima entre dois pontos numa superfície esférica, isto é, calcula a distância como se o deslocamento fosse efetuado em arco sobre uma superfície esférica. Logo, esta fórmula não considera assim então as curvas e/ou diferenças na altitude que possam existir nas estradas reais, dando-nos então apenas uma estimativa da distância entre os diferentes pontos, na qual esta será geralmente menor que a distância real. Consequentemente, esta fórmula também não considera o sentido da distância entre dois pontos, ou seja, a distância entre um ponto A e um ponto B será igual à distância entre um ponto B e um ponto A. Todavia, isto acabará por não ter grandes impactos na avaliação tanto da simulação como também da biblioteca da *Google*, uma vez que as distâncias são utilizadas como inputs para a geração da solução para o problema de *routing*, permitindo então a que estes inputs sejam facilmente modificados e adaptados.

Obviamente que no caso de o sistema ser implementado na realidade, seria importante utilizar as distâncias reais, que permitiria resolver alguns dos problemas já referidos, relacionados com a Equação 10. Estas distâncias poderiam ser obtidas por exemplo utilizando uma API como a da matriz de distâncias da Google[2]. Contudo, notemos que esta API não foi utilizada, devido ao facto de esta ter políticas de faturação, após exceder um período limitado gratuito de teste da mesma.

As distâncias obtidas foram depois convertidas para tempo de deslocamento (em segundos) sendo a matriz de tempos de deslocamento (*TravelTimes*) então atualizada de forma a conter o tempo deslocamento em cada célula para cada distância anteriormente calculada através da Equação 10 e a velocidade média a que o veículo se desloca. A fórmula para a transformação da distância para tempo de deslocamento utilizada é a seguinte:

$$t = \frac{d}{v}$$

Equação 11 - Tempo de deslocamento

Onde t é o tempo de deslocamento em segundos, d a distância entre os dois pontos em metros e v a velocidade média a que o veículo se desloca em metros por segundo.

É importante notar que o valor considerado da velocidade média de deslocamento a que cada veículo se desloca foi o mesmo para todos os veículos (e.g. 40 km/h convertido para metros por segundo). No entanto, poderiam ter sidos consideradas diferentes velocidades deslocamento para cada um dos diferentes veículos. Para este caso, teria também que ser feita uma pequena alteração, na qual em vez de existir uma única matriz de tempos de deslocamento, teriam que existir várias matrizes de tempos de deslocamento, sendo então utilizada uma matriz para cada um dos diferentes veículos.

[2] https://developers.google.com/maps/documentation/distance-matrix/start

A Figura 16 demonstra um exemplo de utilização da matriz de tempos de deslocamento utilizando o segundo como unidade temporal.

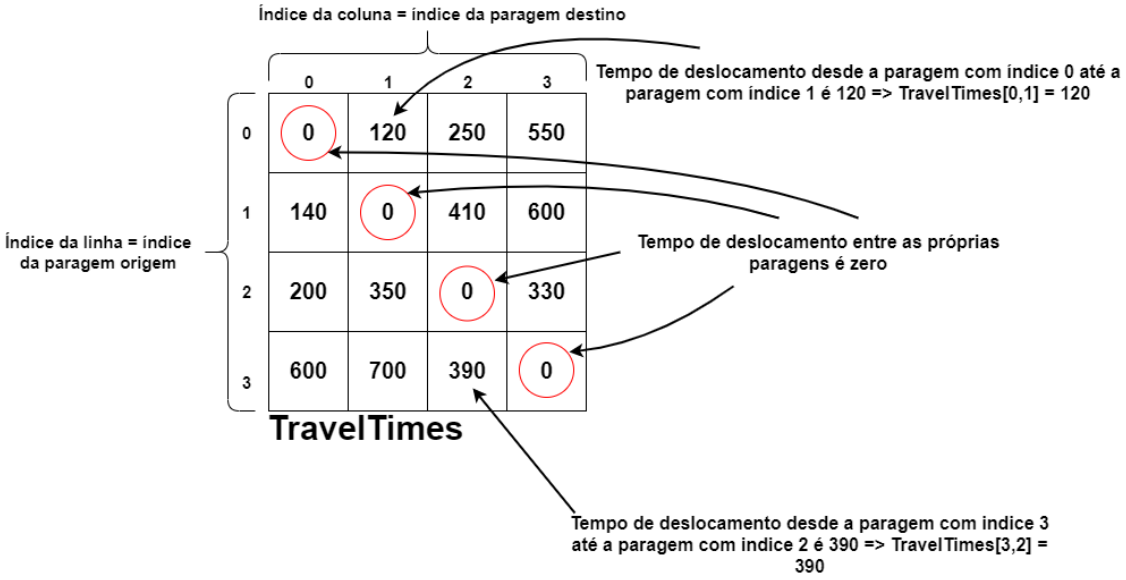

Figura 16 - Exemplo de utilização para a matriz de Tempos de Deslocamento com quatro paragens consideradas

Por exemplo caso queiramos saber o tempo de deslocamento entre a paragem com índice 2 (origem) e a paragem com índice 1 (destino), este será o valor presente na posição da matriz com a linha 2 e a coluna 1, ou seja, 350 segundos ($TravelTimes(2,1) = 350$). Assim, para saber a distância entre duas paragens basta aceder à posição da matriz que tenha a linha indicada pelo índice da paragem origem e a coluna indicada pelo índice da paragem destino.

4.2.1.4 Intervalos de tempo de cada paragem

A matriz de intervalos de tempo (*TimeWindows*) como o nome indica, conterá os intervalos de tempo que devem ser cumpridos para cada paragem, que estarão contidos num intervalo $[TimeWindows_{min}(i), TimeWindows_{max}(i)]$ para cada paragem i, onde $TimeWindows_{min}(i)$ representa o tempo de chegada à paragem i, ou seja, tempo mínimo para estar na paragem i e $TimeWindows_{max}(i)$ o tempo de partida da paragem i, ou seja tempo máximo para partir da paragem i. O número de linhas desta matriz será o número de paragens consideradas na lista de paragens (*Stops*) do *IndexManager*, sendo que cada índice da linha irá representar o índice que da respetiva paragem contida no *IndexManager*. No que toca ao número de colunas, esta terá duas colunas para cada paragem, na qual a primeira coluna irá conter o tempo mínimo (do intervalo) para estar nessa paragem (tempo de chegada) e a segunda coluna conterá o tempo máximo (do intervalo) para partir dessa paragem (tempo de partida), dando-nos então o intervalo de tempo que deverá ser cumprido para a paragem em questão. Esta matriz é inicialmente inicializada a zero para os tempos mínimos de chegada a cada paragem e sendo o tempo máximo de partida de cada paragem inicializado com um valor bastante elevado que não será ultrapassado (e.g. 24 horas convertidas para segundos). Depois para cada cliente será encontrado o índice que representa a paragem de "recolha" e de "entrega".

Após saber o índice da paragem de "recolha" do cliente em questão o valor mínimo ou tempo de chegada a essa essa paragem será o máximo entre o valor atualmente presente no índice dessa paragem e o valor do tempo de "recolha" desejado para esse cliente, como se encontra representado na Equação 12.

$$TimeWindows(i,0) = max\{TimeWindows(i,0), DesiredPickupTime(j)\}, \qquad (i,j) \in \mathbb{N}$$

Equação 12 - Cálculo do tempo de chegada para a paragem com índice i

Onde $TimeWindows(i,0)$ é a o valor tempo de chegada à paragem i na matriz de intervalos de tempo, $DesiredPickupTime(j)$ representa o tempo desejado para ser "recolhido" por um dado cliente j na paragem i e max é o valor máximo entre ambos os valores.

Como os valores mínimos (tempo de chegada) dos intervalos de tempo das paragens são inicializados a 0 este valor será geralmente o valor do tempo desejado de "recolha" para o cliente que está a ser considerado. Para o caso de não existir nenhum cliente com esta paragem como paragem de "recolha" será o valor inicial, ou seja 0.

No que toca aos tempos de "entrega", após saber o índice da paragem de "entrega" do cliente em questão o valor máximo para essa paragem será o mínimo entre o valor atualmente presente nesse mesmo índice e o valor de tempo de "entrega" desejado pelo cliente em questão, como se encontra representado na Equação 13.

$$TimeWindows(i,1) = \min\{TimeWindows(i,1), DesiredDeliveryTime(j)\}, \qquad i \in \mathbb{N}$$

Equação 13 - Cálculo do tempo de partida da paragem com índice i

Onde $TimeWindows(i,1)$ é o valor do tempo de partida da paragem i na matriz de intervalos de tempo, $DesiredDeliveryTime(j)$ representa o tempo desejado para ser "entregue" pelo cliente j, na paragem i e min representa o mínimo entre ambos estes valores.

Como os valores máximos (tempo de partida) dos intervalos de tempo das paragens são inicializados com um valor bastante elevado, os limites máximos dos intervalos de tempo serão geralmente o tempo de "entrega" desejado pelo cliente que está a ser considerado. Para o caso de não existir nenhum cliente com esta paragem como paragem de "entrega", este valor será então o valor definido inicialmente.

A Figura 17 exemplifica a utilização da matriz de intervalos de tempo para quatro paragens.

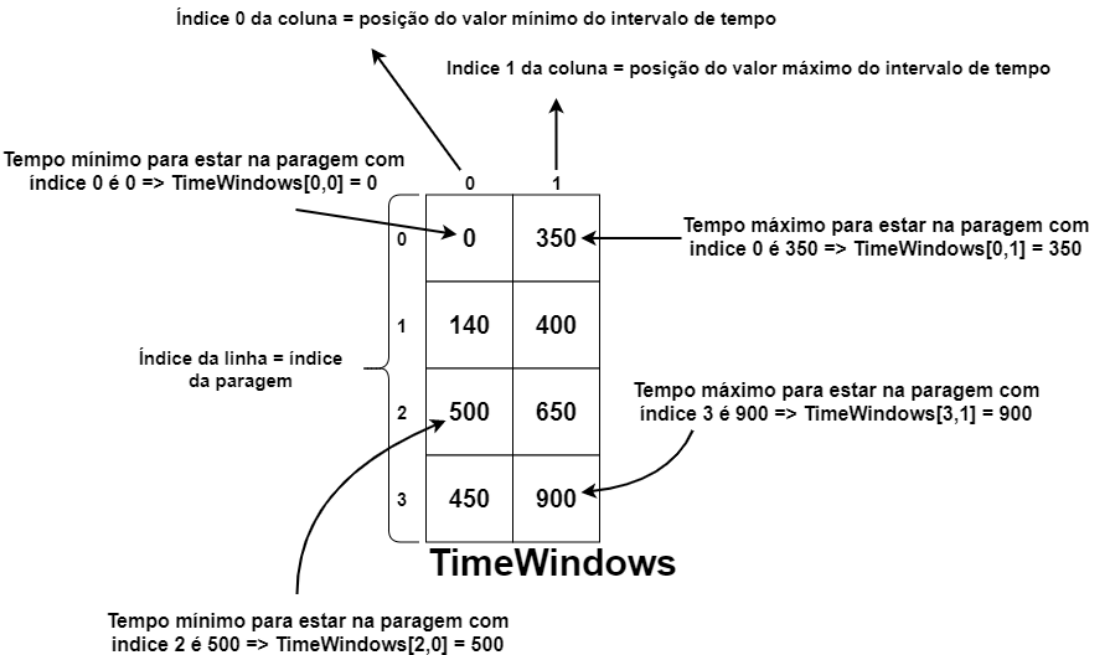

Figura 17 - Exemplo para a matriz de intervalos de tempo constituída por quatro paragens

Com este exemplo é possível constatar que se quisermos saber o tempo mínimo do intervalo de tempo de uma paragem representada por um dado índice, basta acedermos a linha indicada por esse índice e coluna 0 da matriz de intervalos de tempo. Relativamente ao tempo máximo do intervalo de tempo basta acedermos à posição contida na linha indicada pelo índice que representa essa paragem, e coluna 1 da matriz de intervalos de tempo.

4.2.1.5 Pares "recolha" e "entrega" dos clientes

Para além destas matrizes, é ainda necessário construir a matriz de "recolha" e "entrega" (*PickupsDeliveries*), que irá conter os pares de paragens ("recolha" e "entrega") que devem ser visitadas para cada um dos clientes considerados. O tamanho desta matriz será o número de clientes contidos na lista de clientes (*Customers*) do *IndexManager* para as linhas, tendo esta duas colunas. A primeira coluna contém o índice que representa a paragem de "recolha" para o cliente que é representado pelo índice da linha e a segunda contém o índice que representa a paragem de "entrega" para esse mesmo cliente. Para o caso de existir clientes no estado "dentro de um veículo", o índice da paragem de "entrega" é inserido com o valor -1 de modo a indicar que não será necessário passar na paragem de "recolha", uma vez que esta já foi visitada anteriormente. Esta matriz servirá para impor restrições de precedência na qual a paragem de "recolha" deverá ser visitada antes da paragem de "entrega" e ainda restrições do mesmo veículo, no qual o veículo que irá servir a paragem de "recolha" também deverá ser o mesmo que servirá a paragem de "entrega".

A Figura 18 exemplifica a utilização da matriz de “recolha” e “entrega”, para uma situação onde são considerados três clientes:

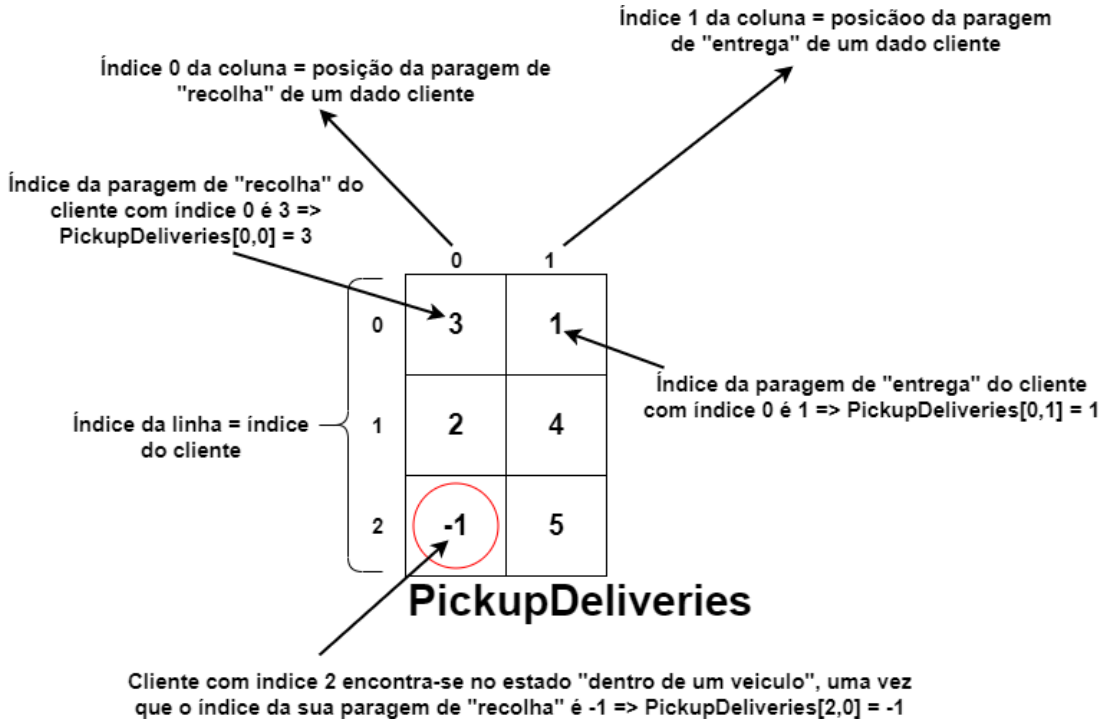

Figura 18 - Exemplo da utilização da matriz de "recolha" e "entrega" para três clientes

Com este exemplo, é possível verificar que para sabermos a paragem de “recolha” de um dado cliente, basta aceder a linha que representa o índice desse cliente e a coluna 0 da matriz de “recolha” e “entrega”. No que toca à paragem de “entrega”, é obtida ao aceder à linha que representa o índice do cliente considerado e a coluna 1 da matriz de “recolha” e “entrega”.

4.2.1.6 Clientes presentes em cada veículo

Existe ainda a necessidade de calcular o vetor dos clientes presentes em cada veículo (*CustomersVehicle*). Tal serve para indicar qual o veículo que deve servir um dado cliente. Neste vetor, cada índice do vetor indica o índice que representa o cliente, sendo que célula do mesmo indica o índice que representa o veículo que irá servir esse cliente, sendo também seguido o mesmo tipo de indexação que o *IndexManager*. Relativamente ao tamanho deste vetor, este será o número de clientes considerados.

Para cada um dos clientes existentes na lista de clientes do *IndexManager* é verificado, se este se encontra no estado “dentro de um veículo”, caso afirmativo, o valor presente na célula do vetor com o índice desse cliente, será então o índice que representa o veículo que o contém. Para a situação em que cliente não esteja no estado “dentro de um veículo” a célula que contém o índice do veículo para esse cliente será -1, indicando que o mesmo não se encontra dentro de um veículo atualmente.

Com isto, é possível compreender que este vetor, fundamentalmente, permite indicar o estado atual de um cliente relativamente a um veículo, indicando se um dado cliente já se encontra dentro de um dado veículo, ou se este ainda não foi “recolhido” por nenhum veículo. Este vetor será utilizado de forma a impor a restrição de que um cliente esteja no estado “dentro do veículo”, terá que ser servido pelo mesmo veículo que o “recolheu” inicialmente.

É ainda importante realçar que para problemas de *routing* estáticos as células deste vetor terão todas o valor -1, pois não existem clientes no estado "dentro de um veículo", porque a geração do percurso é efetuada numa situação inicial onde não existem veículos em serviço. No entanto para problemas de *routing* dinâmicos, as células deste vetor poderão então conter valores distintos de -1, isto se existirem clientes dentro de algum dos veículos considerados no momento em que é efetuada reotimização dos percursos atuais dos mesmos.

A Figura 19 exemplifica a utilização deste vetor para um caso em que existem cinco clientes no total, existindo três deles no estado "dentro do veículo" e dois deles no estado "fora do veículo".

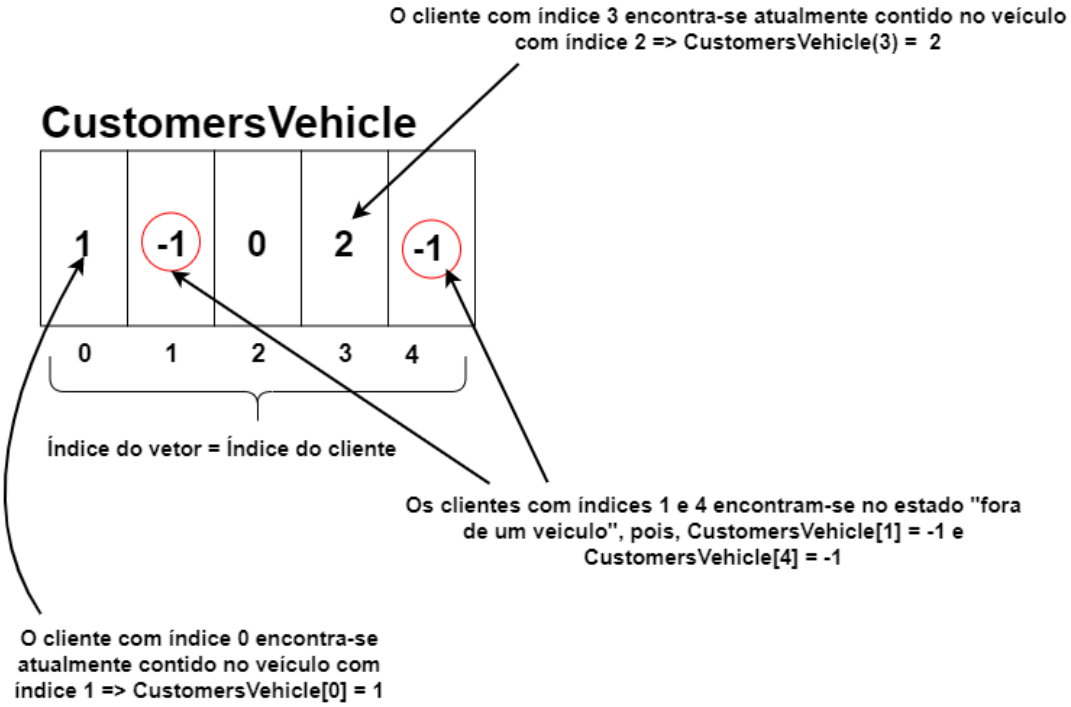

Figura 19 – Exemplo da utilização do vetor dos clientes presentes em cada veículo para cinco clientes, existindo dois clientes no estado "fora do veículo" e três clientes no estado "dentro do veículo"

Como é possível verificar, para sabermos o estado de um dado cliente relativo a um veículo, basta acedermos ao índice que representa esse cliente no vetor. Se o valor for -1 o cliente não se encontra em nenhum veículo. Caso este valor seja diferente de -1, este valor indicará o índice que representa o veículo, no qual esse cliente está contido.

4.2.1.7 Tempos de viagem atuais dos clientes

O vetor de tempos de viagens atuais dos clientes (*CustomerRideTimes*) serve para representar o tempo de viagem atual de cada cliente que se encontre no estado "dentro de um veículo". O tamanho deste vetor será o número de clientes contidos na lista de clientes (*Customers*) do *IndexManager*, no qual cada índice deste vetor designará o índice do cliente em questão, sendo que o valor presente em cada célula indicará o tempo de viagem do cliente indicado por esse índice. Se o cliente estiver no estado "fora de um veículo" o seu tempo de viagem será 0, pois este ainda não entrou em nenhum veículo e, consequentemente, não começou ainda a ser servido. Para problemas de *routing* estáticos, o tempo de viagem de qualquer cliente será 0, uma vez que a solução para um dado problema estático é construída

quando ainda não existem clientes dentro do veículo, levando então a que não existem tempos de viagem para cada cliente (tempo de viagem é zero).

No entanto, para casos de *routing* dinâmico, em que podem existir clientes no estado "dentro de um veículo", o tempo de viagem atual desses clientes terá que ser calculado. Sendo estes tempos obtido através da diferença entre o momento em que um dado cliente entrou no veículo e o momento atual, em que está a ser efetuado a reotimização dos percursos atuais. A seguinte equação reflete o cálculo do tempo de viagem atual de um dado cliente j que esteja no estado "dentro de um veículo":

$$CurrentRideTime(j) = CurrentTime - RealPickupTime(j), \qquad j \in \mathbb{N}$$

Equação 14 - Tempo de viagem atual de um cliente j no estado "dentro de um veículo"

Onde $CurrentRideTime(j)$ é o tempo de viagem atual de um cliente j no estado "dentro de um veículo", $CurrentTime$ é o tempo atual, no qual é efetuada a reotimização do percurso e $RealPickupTime(j)$ é o tempo (real) de entrada do cliente j no estado "dentro de um veículo".

É importante salientar que este vetor será utilizado de forma a auxiliar na imposição das restrições dos tempos máximos de viagem para os clientes que já se encontram no estado "dentro de um veículo".

A Figura 20 demonstra um exemplo de utilização do vetor de tempo de viagem para uma situação na qual são considerados cinco clientes:

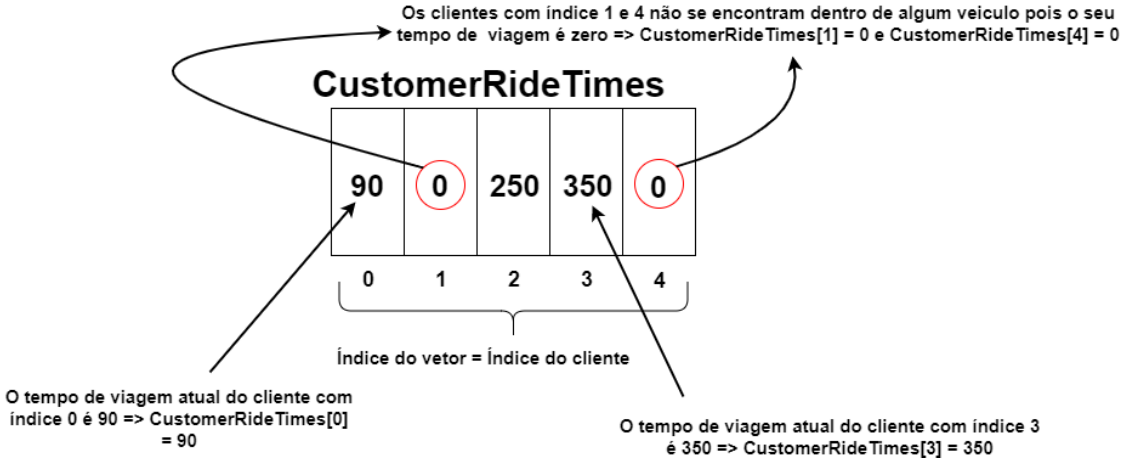

Figura 20 - Exemplo de utilização do vetor do tempo de viagem atual dos clientes

Caso queiramos saber o tempo de viagem para um dado cliente representado por um dado índice, basta acedermos à posição indicado por esse índice no vetor dos tempos de viagens atuais dos clientes.

4.2.1.8 Procura em cada paragem

Por fim, é necessário determinar o vetor procura (*Demands*) que irá conter a procura em cada paragem, sendo que cada índice deste vetor representa o índice da respetiva paragem na lista de paragens (*Stops*) do *IndexManager.* Todas as posições deste vetor são inicialmente inicializadas a 0. Depois, para cada cliente, são obtidos os seus índices para a sua paragem de "recolha" e para a sua paragem de "entrega". Para as paragens de "recolha" é adicionado 1 ao

valor atualmente presente na matriz de procura (*Demands*). No que toca às paragens de "entrega", é subtraído 1 ao valor atualmente presente no vetor de procura. Sendo este processo repetido até terem sido percorridos todos os clientes "dentro do veículo" como também os que se encontram "fora do veículo", permitindo então obter a matriz de procura final, onde cada célula indicará o número de pessoas que entram ou saem do veículo. A equação que reflete este comportamento encontra-se abaixo representada:

$$Demands(i) = \begin{cases} Demands(i) + 1, & sse\ i\ \text{é}\ paragem\ de\ \text{"recolha"}\ de\ um\ dado\ cliente \\ Demands(i) - 1, & sse\ i\ \text{é}\ paragem\ de\ \text{"entrega"}\ de\ um\ dado\ cliente \end{cases}$$

Equação 15 - Cálculo da Procura para um dado cliente na paragem i

Onde $Demands(i)$ é o valor atualmente presente no vetor de procura para a paragem i, e i é uma paragem de "recolha" ou "entrega" de um cliente.

A Figura 21 exemplifica a utilização deste vetor para um conjunto de sete paragens consideradas:

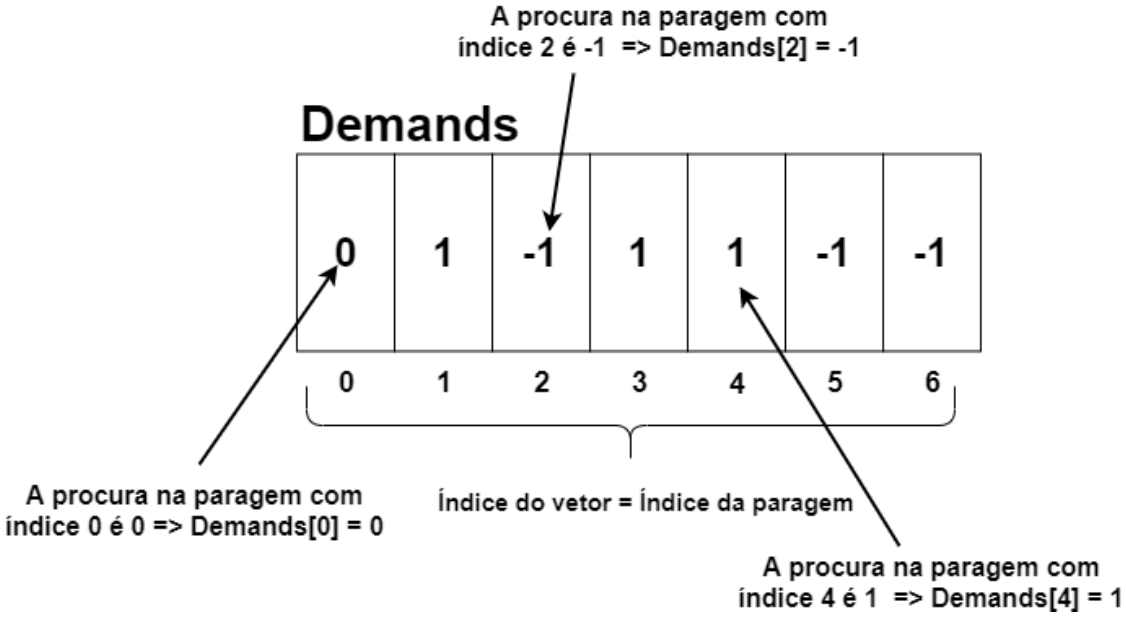

Figura 21 - Exemplo de utilização do vetor de Procura para sete paragens consideradas

Para sabermos a procura numa paragem representada por um dado índice, basta acedermos a essa posição no vetor de procura (*Demands*). Se o valor for negativo, esse valor indica o número de clientes que saem nessa paragem. Caso seja 0 indica que nenhum cliente entrou ou saiu na paragem considerada. Por fim, caso seja positivo indica o número de clientes que entraram no veículo na paragem indicada pelo índice da posição do vetor.

É importante referir que este vetor é também uma das razões para a utilização da múltipla indexação nas paragens. Consideremos o seguinte exemplo utilizando indexação única: temos um cliente c_1 com a paragem A com índice i como paragem de "entrega" e um cliente c_2 com a mesma paragem A com índice i como paragem de "recolha". Este problema pode ser resumido da seguinte forma:

$$PD(c_1, 1) = i \text{ e } PD(c_2, 0) = i$$

$$\text{Onde } A = i$$

Se calcularmos a procura para essa paragem A com índice i utilizando a Equação 15 obteremos:

$$Demands(i) = (-1) + 1$$

$$\Leftrightarrow Demands(i) = 0$$

Exemplo 1 – Cálculo da procura na paragem A com dois clientes que saem e entram na mesma respetivamente utilizando indexação única

Como podemos ver a procura será 0, indicando que não entra nem sai nenhum cliente nessa paragem, tendo sido então a informação de certa forma "perdida". No entanto isto pode ser ultrapassado ao ser utilizada a múltipla indexação para as paragens.

Consideremos agora a mesma situação, mas utilizando múltipla indexação. Onde o cliente c_1 tem a paragem A com índice i como paragem de "entrega" e c_2 tem a mesma paragem A com índice j como paragem "recolha", ou seja:

$$PD(c_1, 1) = i \text{ e } PD(c_2, 0) = j$$

$$\text{Onde } A = \{i, j\}$$

Ao utilizarmos índices distintos (múltipla indexação) que representam a mesma paragem real A, iremos obter a seguinte procura utilizando a Equação 15:

$$Demands(i) = -1$$

$$Demands(j) = 1$$

Exemplo 2 - Cálculo da procura na paragem A com dois clientes que saem e entram na mesma respetivamente utilizando múltipla indexação

Assim, ao separarmos os índices é possível perceber que um cliente irá sair na paragem A e um outro cliente irá entrar nessa mesma paragem, permitindo assim resolver o problema encontrado anteriormente da única indexação. Note-se que este mesmo problema, encontrado com a indexação única foi um dos principais fatores que contribuíram para a utilização da múltipla indexação na lista de paragens (*Stops*) contida no *IndexManager*.

4.2.2 Solver

Após construir as estruturas de dados já referidas, que constituem o modelo de dados, será então possível solucionar o problema imposto pelo mesmo utilizando o *Solver* da *Google*. Note-se que o *Solver,* será a entidade responsável por resolver o problema de *routing*, imposto pelo modelo de dados (do *Solver)*. É no Solver que serão impostas todas as restrições e nuances do problema, nas quais estas se encontram de alguma forma representadas no modelo de dados. Existe um conjunto de identidades essenciais que permitem ao *Solver* resolver um problema de *routing*:

- **Modelo de dados**: o modelo de dados anteriormente referido que contém as informações necessárias que irão permitir definir as restrições e características do problema no *Solver*;

- ***RoutingIndexManager***: entidade responsável por efetuar o mapeamento de nós para índices e de índices para nós para ser utilizado pelo *RoutingModel* (sendo este semelhante ao já falado *IndexManager* utilizado no modelo de dados);
- ***RoutingDimension***: entidade contida no *RoutingModel*, responsável por seguir e armazenar todas quantidades que se acumulam ao longo de um percurso para um conjunto de veículos (e.g. tempos de deslocamento, distâncias percorridas, capacidades, entre outras) para mais informações visitar: https://developers.google.com/optimization/routing/dimensions
- ***RoutingModel***: representação do modelo de *routing* que conterá todas as restrições e características do problema como também o *Solver* que irá possibilitar a resolução do mesmo.

O *RoutingIndexManager* começa por ser inicializado com as paragens iniciais e finais (*Starts* e *Ends*), como também o número total de paragens a ser tidas em conta e ainda o número total de veículos a ser utilizados, sendo que todas estas informações advêm do modelo de dados. Seguidamente, é construído o *RoutingModel* inicial utilizando as informações contidas no *RoutingIndexManager* e sendo a este atribuído a função de custo principal e a matriz de tempos de deslocamento (*TravelTimes*) presente no modelo de dados. O objetivo da função de custo será minimizar o custo temporal enquanto são satisfeitas todas as restrições impostas pelo modelo de dados, nas quais estas serão referidas mais à frente. Uma vez que o objetivo é a minimização dos tempos de deslocamento, é importante referir que por sua vez, a minimização da distância total percorrida também estará indiretamente presente nesta função objetivo, uma vez que os tempos de deslocamento são calculados a partir da distância entre cada paragem e a velocidade média a que o veículo se desloca.

4.2.2.1 Dimensões

Como já referido, uma dimensão (*RoutingDimension*) é responsável por seguir e guardar todas as quantidades que se irão acumular ao longo de um percurso para o conjunto de veículos considerados. De forma a obter estas quantidades, as dimensões utilizam três tipos distintos de variáveis: *Cumul*, *Slack* e *Transit* [25]. Sendo que as variáveis *Cumul* representam as quantidades acumuladas para uma determinada paragem de um percurso. O aumento ou redução das quantidades entre paragens é representado pelas variáveis *Transit*. Por fim, as variáveis *Slack* representam algum tipo de "folga" que pode ocorrer numa determinada paragem de um percurso, e.g., tempo de espera em cada nó para dimensões temporais. Estas variáveis relacionam-se da seguinte forma:

$$Cumul_d(j) = Cumul_d(i) + Transit_d(i) + Slack_d(i), \qquad (i,j) \in \mathbb{N}$$

Equação 16 - Valor acumulado na paragem com índice j

Onde j representa uma paragem j seguinte à paragem i, d representa uma dimensão (*RoutingDimension*). $Cumul_d(j)$ representa o valor acumulado para a dimensão d na paragem j seguinte à paragem i, $Cumul_d(i)$ o valor acumulado para a dimensão d para uma dada paragem i, $Transit_d(i)$ o valor da variável $Transit$ na dimensão d para uma dada paragem i e $Slack_d(i)$ representa o valor da variável $Slack$ para a dimensão d numa determinada paragem i.

Com a Equação 16 é possível ver que para uma determinada *RoutingDimension* d, a quantidade acumulada para a paragem j, seguinte à paragem i, será a soma entre as variáveis

Cumul, *Transit* e *Slack* da paragem i. É ainda importante referir que qualquer uma destas variáveis, por vezes podem tomar a forma de intervalos, ou seja cada uma delas pode conter um valor mínimo e máximo.

A primeira dimensão que foi necessário criar foi a *TimeDimension*, que estará responsável por seguir e guardar todas quantidades temporais que se vão armazenando ao longo dos percursos de cada veículo, como tempos de deslocamento e tempos de espera em cada paragem que irão permitir obter os intervalos de tempo da solução final para estar em cada paragem. Sendo esta dimensão inicializada indicando a matriz que permitirá obter os valores da variável de *Transit,* o tempo máximo de cada percurso gerado, e ainda indicando o valor máximo das variáveis de *Slack.* A variável *Cumul* para uma determinada paragem irá conter o tempo total acumulado para um dado veículo estar nessa mesma paragem.

As variáveis de *Transit* podem ser vistas como a as quantidades temporais que serão adicionadas à variável *Cumul* e *Slack,* da paragem atual considerada, de forma a obter o valor da variável *Cumul* da paragem seguinte. Por outras palavras, as variáveis *Transit* permitem obter o custo (temporal) de deslocamento desde a paragem atual para a paragem seguinte. Dito isto, a matriz de tempos de deslocamento (*TravelTimes*) presente no modelo de dados, é então utilizada como forma de representação das variáveis *Transit*, permitindo então refletir os tempos de deslocamento que serão adicionadas ao longo do percurso de um dado veículo. A relação entre a variável *Transit* e a matriz de tempos de deslocamento (*TravelTimes*) reflete-se na seguinte igualdade:

$$Transit_t(i) = TravelTimes(i, j), \qquad i \in \mathbb{N}$$

Equação 17 - Igualdade entre Transit da dimensão temporal e matriz de deslocamento

Onde *i* é uma paragem anterior a *j*, $Transit_t(i)$ é o valor da variável *Transit* para dimensão temporal *t* na paragem *i* e $TravelTimes(i, j)$ é o tempo de deslocamento presente na matriz de tempos de deslocamento desde a paragem *i* e para a paragem *j*.

Relativamente ao tempo máximo de cada percurso, a este é atribuído um valor bastante elevado de forma a não restringir a duração total dos percursos de cada veículo. Como já referido a variável de *Slack* para uma dimensão temporal, pode ser vista como tempos de espera em cada paragem, o valor máximo atribuído a estas variáveis de "folga" em cada paragem foi também um valor bastante elevado de forma a permitir aos veículos ficarem nas paragens à espera o tempo que fosse necessário.

Para exemplificação das variáveis *Slack*, *Cumul* e *Transit*, consideremos o seguinte percurso $1 \rightarrow 3$, onde 1 é a primeira paragem a ser visitada e 3 a paragem visitada depois de 1, depois consideremos as seguintes variáveis para uma dimensão temporal *t*: $Cumul_t(1) = 5, Transit_t(1) = 3, Slack_t(1) = 2$, seguindo a Equação 16, $Cumul_t(3)$ será:

$$Cumul_t(3) = Cumul_t(1) + Transit_t(1) + Slack_t(1)$$

$$\Leftrightarrow Cumul_t(3) = 5 + 3 + 2$$

$$\Leftrightarrow Cumul_t(3) = 10$$

Ou seja, o tempo para chegar à paragem seguinte *3* $Cumul(3)$ será a soma entre o tempo para chegar à paragem anterior *1* $Cumul(1)$, mais o tempo de deslocamento entre a paragem 1 e a paragem seguinte 3, ou seja, $Transit(1) = TravelTimes(1,3)$ usando Equação

17 e ainda adicionando o tempo de espera na paragem 1 $Slack(1)$. Uma vez que o tempo de espera na paragem 1, é 2 $Slack_t(1) = 2$, o intervalo de tempo que o veículo estará na paragem 1 será [5,7], ou seja $Cumul_t(1) + Slack_t(1)$, logo podemos deduzir que para um dada paragem *i* o intervalo de tempo que um veículo estará nessa paragem será:

$$\Delta Cumul_t(i) = [Cumul_t(i), Cumul_t(i) + Slack_t(i)], \qquad i \in \mathbb{N}$$

Equação 18 - Intervalo de tempo que um veículo estará numa dada paragem

Foi ainda criada uma outra dimensão, a *CapacityDimension*, que estará responsável por rastrear e armazenar todas as quantidades relativas à capacidade do veículo que se vão acumulando ao longo dos percursos. Servirá também para impor a restrição da capacidade máxima imposta a cada veículo. Esta dimensão é inicializada utilizando o vetor *VehicleCapacities* que indica a capacidade total para cada um dos veículos considerados e ainda com o vetor de procura (*Demands*) que será a base para as variáveis de *Transit* da mesma. As variáveis *Transit* para esta dimensão irão permitir saber a quantidade de clientes que entram ou saem do veículo da paragem atualmente considerada, podendo então estes valores serem positivos ou negativos. O vetor de procura (*Demands*) e as variáveis *Transit*, relacionam-se da seguinte forma para uma dimensão de capacidades *c*:

$$Transit_c(i) = Demands(i), \ i \in \mathbb{N}$$

Equação 19 – Igualdade entre Transit da dimensão de capacidades e o vetor de procura

Onde $Transit_c(i)$ representa o valor da variável *Transit* da dimensão de capacidade *c* na paragem *i* e $Demands(i)$ representa o valor presente no vetor de procura para a paragem *i*.

A variável *Cumul* para uma dada paragem dará o número acumulado de clientes que se encontram no veículo para essa mesma paragem, sendo que este número não pode ultrapassar a capacidade máxima do veículo. É importante ainda referir que, para esta dimensão não existirá variável *Slack* uma vez que esta não faz muito sentido quando falamos de uma dimensão que reflete a quantidade atual acumulada de clientes existentes num determinado veículo.

Consideremos agora o exemplo anterior do percurso $1 \rightarrow 3$, onde 1 é a primeira paragem a ser visitada e 3 a seguinte paragem a ser visitada, $Transit_c(1) = -1$, e $Cumul_c(1) = 3$ o valor acumulado do número de clientes quando o veículo chega à paragem seguinte 3 $Cumul_c(3)$ será:

$$Cumul_c(3) = Cumul_c(1) + Transit_c(1) + Slack_c(1)$$

$$\Leftrightarrow Cumul(3) = 3 + (-1) + 0$$

$$\Leftrightarrow Cumul(3) = 2$$

Ou seja, o número acumulado de clientes presentes quando o veículo chega à paragem *3* será a soma entre o número de clientes acumulados na paragem *1* $Cumul(1)$ e o número de clientes que irão sair na paragem *1* $Transit(1)$. Com este exemplo podemos perceber que o número de clientes acumulados presentes no veículo quando este chega à paragem 1 é 3 ($Cumul_c(1) = 3$) e quando este chega à paragem seguinte (paragem com índice 3) será 2 ($Cumul_c(3) = 2$), pois ocorreu a saída de um cliente na paragem 1, sendo isto indicado pela variável *Transit* da paragem 1 ($Transit_c(1) = -1$).

4.2.2.2 Imposição das restrições

4.2.2.2.1 Intervalos de tempo

As restrições de intervalos de tempos são impostas pela matriz de intervalos de tempo (*TimeWindows*), na qual para cada paragem indicada pelo índice da linha será indicado o intervalo de tempo que deverá ser cumprido. Ou seja, o valor acumulado de tempo para uma dada paragem deverá de estar dentro do intervalo $[TimeWindows(i,0), TimeWindows(i,1)]$ contido na matriz de intervalos de tempo para essa mesma paragem como é indicado em R 1.

$$TimeWindows(i,0) \leq Cumul_t(i) \leq TimeWindows(i,1), \forall\, i \in \mathbb{N}$$

R 1 - Restrição de intervalos de tempo

Onde $TimeWindows(i,0)$ representa o tempo mínimo de chegada à paragem i, e $TimeWindows(i,1)$ representa o tempo máximo para partir da paragem i, $Cumul_t(i)$ é o tempo acumulado da *TimeDimension* para a paragem i.

4.2.2.2.2 Precedência

A restrição de precedência é imposta pela matriz de "recolha" e "entrega" (*PickupsDeliveries*) presente no modelo de dados, de forma a indicar ao *Solver* que a paragem de "recolha" deverá ser primeiro visitada antes de visitar a paragem de "entrega" para os clientes que se encontram no estado "fora do veículo". Para os clientes que se encontram no estado "dentro do veículo" não é preciso fazer nada, uma vez que a sua paragem de "recolha" já terá sido visitada anteriormente, pois o cliente já se encontra dentro de um dado veículo, levando a que esta restrição esteja, de certa forma, inerentemente respeitada. Para que esta restrição seja respeitada, o tempo acumulado de partida da paragem de "recolha" deverá ser inferior ao tempo acumulado de chegada à paragem de "entrega", ou por outras palavras caso a seguinte desigualdade seja verificada:

$$Cumul_t(r) \leq Cumul_t(e), \qquad (r,e) \in \mathbb{N}$$

R 2 - Restrição de Precedência

Onde $Cumul_t(r)$ é o intervalo de tempo acumulado da *TimeDimension* da paragem de "recolha" r e $Cumul_t(e)$ é o intervalo de tempo acumulado da *TimeDimension* para a paragem de "entrega" e.

4.2.2.2.3 Mesmo veículo

A restrição do mesmo veículo é também imposta pela matriz de "recolha" e "entrega" do modelo de dados para os clientes que se encontram "à espera para ser servidos", no qual para um par "recolha" e "entrega" é indicada ao *Solver* que o veículo que efetua a "recolha" também deverá ser o mesmo que efetua a "entrega". A igualdade que representa esta restrição encontra-se abaixo representada:

$$VehicleVar(r) = VehicleVar(e), \qquad (r,e) \in \mathbb{N}$$

R 3 - Restrição do mesmo veículo

Onde $VehicleVar(r)$ é o veículo que serve a paragem de "recolha" r de um cliente j e $VehicleVar(e)$ é o veículo que serve a paragem de entrega e de um cliente j.

Note-se que, para os casos em que existem clientes no estado "dentro do veículo" é ainda necessário utilizar o vetor dos clientes dentro cada veículo (*CustomersVehicle*) para impor esta restrição, indicando que o cliente deve ser servido pelo veículo indicado pela célula do mesmo, isto se o valor do índice do veículo presente nessa célula for diferente de -1. Após sabido o índice do cliente no estado "dentro do veículo", é encontrado o seu índice da paragem de "entrega" na matriz de "recolha" e "entrega" (*PickupsDeliveries*) e é ainda encontrado o índice da paragem atual desse veículo, que será a paragem de começo do mesmo (*Starts*), uma vez que o percurso será construído a partir da paragem atual em que o veículo se encontra. Após sabida a paragem de começo do veículo e a paragem de "entrega" do cliente é então definida a restrição indicando que o veículo que serve a paragem de começo do mesmo deve ser o mesmo que serve a paragem de "entrega" do cliente em questão.

4.2.2.2.4 Tempo máximo de viagem de um cliente

A restrição de máximo tempo de viagem é imposta pelo valor da constante do tempo máximo de viagem de um cliente, presente no modelo de dados, e pelo tempo direto de viagem de um cliente. Sendo para cada par "recolha" e "entrega" calculado o tempo direto de viagem desde a paragem "recolha" para a paragem "entrega", ou seja, o tempo mínimo que demora se a viagem fosse efetuada diretamente entre a paragem "recolha" e "entrega". Para tal, é utilizada a matriz de tempos de deslocamento (*TravelTimes*) da seguinte forma:

$$DirectRideTime(r, e) = TravelTimes(r, e), \qquad (r, e) \in \mathbb{N}$$

Equação 20 - Tempo direto de viagem de um cliente

Onde $DirectRideTime(r, e)$ o tempo direto de viagem desde a paragem de "recolha" r do cliente j até a paragem de "entrega" e desse mesmo cliente j e $TravelTimes(r, e)$ o tempo de deslocamento presente na matriz de tempo de deslocamento entre a paragem r e a paragem e.

O valor máximo do tempo de viagem de um cliente é finalmente obtido ao somar o tempo direto de viagem desse cliente com o valor da constante do tempo máximo de viagem relativo de um cliente presente no modelo de dados como indica a seguinte equação:

$$MaxRideTime(j) = DirectRideTime(r, e) + MaxRelativeRideTime, \qquad (r, e, j) \in \mathbb{N}$$

Equação 21 - Tempo máximo de viagem de um cliente

Onde $MaxRideTime(j)$ é o tempo máximo de viagem do cliente j, $DirectRideTime(r, e)$ o tempo direto de viagem desde a paragem de "recolha" r do cliente j até a paragem de "entrega" do cliente j e e e $MaxRelativeRideTime$ o tempo máximo de viagem relativo.

Depois de sabido o tempo máximo de viagem para esse dado cliente, é necessário calcular o tempo de viagem real para esse cliente. Este tempo é calculado ao somar o tempo de viagem atual do cliente presente no vetor *CustomerRideTimes* do modelo de dados, com a diferença entre o tempo acumulado para a paragem de entrega do cliente e o tempo acumulado para a paragem de recolha, isto é:

$$RideTime(j) = CustomerRideTimes(j) + \big(Cumul_t(e) - Cumul_t(r)\big), \qquad (r, e, j) \in \mathbb{N}$$

Equação 22 - Tempo de viagem (real) de um cliente j

Onde $RideTime(j)$ é o tempo de viagem (real) do cliente j, $CustomerRideTimes(j)$ é o tempo de viagem atual do cliente j presente no vetor *CustomerRideTimes* do modelo de dados e $Cumul_t(r)$ é o tempo acumulado da *TimeDimension* da paragem de "recolha" r do cliente j e $Cumul_t(e)$ é o tempo acumulado da *TimeDimension* para a paragem de "entrega" e do cliente j.

A restrição do tempo máximo de viagem será satisfeita se o tempo de viagem (real) do cliente calculado através da Equação 22, não ultrapassar o valor máximo de viagem do cliente calculado através da Equação 21, conforme a desigualdade abaixo apresentada:

$$RideTime(j) < MaxRideTime(j), \qquad (j) \in \mathbb{N}$$

R 4 - Restrição do tempo máximo de viagem de um cliente

Onde $RideTime(j)$ é o tempo de viagem (final) do cliente j e $MaxRideTime(j)$ é o tempo máximo de viagem para o cliente j.

4.2.2.2.5 Número máximo de veículos utilizados

A restrição do número máximo de veículos utilizados é inerentemente imposta pelo *Solver* inicialmente quando é criado o *RoutingIndexManager*, no qual este recebe o número de veículos que serão considerados como argumento, sendo que esta restrição pode ser formulada da seguinte forma:

$$NumVehiclesUsed \leq NumVehicles$$

R 5 - Restrição do número de veículos utilizados

Onde $NumVehiclesUsed$ é o número de veículos utilizados para gerar a solução atual, e $NumVehicles$ é o número total de veículos considerados. Note-se que $NumVehicles$ terá o mesmo tamanho que os vetores $Starts$, $Ends$ ou $VehicleCapacities$, uma vez que qualquer um destes vetores contém informações relativas a qualquer um dos veículos considerados em que cada índice do vetor representa um dos veículos.

4.2.2.2.6 Capacidade dos veículos

A Restrição de capacidade dos veículos é imposta através da dimensão de capacidades (*CapacityDimension*), sendo isto possível através do vetor de capacidades dos veículos (*VehicleCapacities*) que irá indicar a capacidade máxima de clientes suportada por um determinado veículo e pelo vetor de procura (*Demands*), permitindo saber a quantidade de clientes que entram ou saem do veículo em cada paragem indicada pelo índice do vetor. Esta restrição é satisfeita caso a quantidade ou carga acumulada de um veículo numa determinada

paragem seja menor ou igual à capacidade máxima desse veículo, ou por outras palavras, se a seguinte condição for verificada:

$$Cumul_c(i) \leq VehicleCapacities(v), \qquad (i, v) \in \mathbb{N}$$

R 6 - Restrição do número atual de clientes presentes num veículo numa dada paragem

Onde $Cumul_c(i)$ é a quantidade atual acumulada da *CapacityDimension* de clientes que se encontram dentro de um dado veículo v numa determinada paragem i e $VehicleCapacity(v)$ é a capacidade máxima do veículo com índice v.

4.2.2.2.7 Paragens de começo e fim de percurso

A restrição das paragens de começo e fim são impostas pelo vetor das paragens de começo (*Starts*) e pelo vetor das paragens de término ou fim de serviço (*Ends*), respetivamente. Cada posição destes vetores contém a respetiva paragem de começo e fim para o veículo indicado pelo índice da mesma. Esta restrição é utilizada de forma a impor que um veículo deve começar o seu percurso numa dada paragem dada pelo vetor *Starts* e deve terminar o mesmo numa outra paragem dada pelo vetor *Ends*. Note-se que esta restrição já é inerentemente imposta ao *Solver*, quando é criado o *RoutingModel*, ao o inicializar com o *RoutingIndexManager* que já as contém. A restrição do começo de paragem pode ser formulada da seguinte forma:

$$Start(v) = Starts(v), \qquad v \in \mathbb{N}$$

R 7- Restrição para as paragens de começo de cada veículo

Onde $Start(v)$ é a paragem de começo para um dado veículo v e $Starts(v)$ é a posição do vetor das paragens de começo com índice v.

De igual modo a restrição das paragens de termino de serviço pode ser representada pela seguinte igualdade:

$$End(v) = Ends(v), \qquad v \in \mathbb{N}$$

R 8 - Restrição para as paragens de término de cada veículo

Onde $End(v)$ é a paragem de término para um dado veículo v e $Ends(v)$ é a posição vetor das paragens de término com índice v.

4.2.2.3 Procura de Soluções

Após a construção do modelo de dados e imposição das restrições já referidas, torna-se então possível a geração de soluções que permitam então resolver o Dial-a-Ride-Problem formulado. A biblioteca da *Google OR-Tools* contém já um conjunto de algoritmos implementados que possibilitam o encontro de soluções viáveis, sendo que estes algoritmos podem ser separados em dois tipos: algoritmos de encontro de solução inicial e algoritmos de procura de soluções. Sendo que os algoritmos do primeiro tipo, como o nome indica estão responsáveis por gerar uma solução inicial viável e possível. Os segundos algoritmos ficam responsáveis por melhorar as soluções iniciais de forma a encontra uma melhor solução utilizando algum tipo de meta-heurística que o permita efetuar. Para mais informações acerca destes algoritmos consultar a documentação sobre as opções de *routing*[3]. É ainda importante

[3] https://developers.google.com/optimization/routing/routing_options

referir que para qualquer um destes algoritmos existe ainda uma opção que permite ao próprio *solver* selecionar a estratégia a ser utilizada automaticamente tendo em conta o problema que está a ser resolvido.

No que toca aos algoritmos de encontro da primeira solução, foi optado por utilizar sempre a mesma estratégia, a *Parallel Cheapest Insertion*. Sendo que esta resume-se: a construir iterativamente a solução inserindo os nós (que representam as paragens) com menos custo nas posições com menos custo, sendo o custo da inserção baseado na função de custo de principal, ou seja, para este caso em particular, baseada na matriz de tempos de deslocamento (*TravelTimes*). Note-se que, a escolha da utilização deste algoritmo de primeira solução, deve-se pelo facto do mesmo ser o algoritmo que é automaticamente selecionado pelo *solver* para a resolução de problemas de *routing* de veículos que tenham restrições de "recolha" e "entrega" como é possível constatar em [26].

Relativamente aos algoritmos de pesquisa de soluções, esta biblioteca contém cinco algoritmos no total, estando estes representados na Tabela 7.

Algoritmo	**Descrição**
Greedy Descent	Melhora a vizinhança da pesquisa local, reduzindo os custos até que um mínimo local seja atingindo.
Guided Local Search	Utiliza pesquisa local guiada (G*uided Local Search)* de forma a escapar mínimas locais. https://en.wikipedia.org/wiki/Guided_Local_Search
Simulated Annealing	*Utiliza Simulated Annealing* de forma a escapar mínima locais. https://en.wikipedia.org/wiki/Simulated_annealing
Tabu Search	Utiliza a pesquisa tabu (*Tabu Search*) de forma a escapar mínimas locais. https://en.wikipedia.org/wiki/Tabu_search
Objective Tabu Search	Utiliza a pesquisa tabu (*Tabu Search*) no valor da função objetivo da solução, de forma a escapar mínimas locais.

Tabela 7 - Algoritmos de pesquisa de soluções presentes na biblioteca da Google OR-Tools

Estes algoritmos de pesquisa estão geralmente limitados temporalmente ou pelo número de soluções encontradas, de forma a que a pesquisa não ocorra infinitamente para situações em que o espaço de pesquisa de soluções seja bastante elevada. Assim sendo, as soluções geradas também por vezes poderão não ser as ótimas. No entanto, são soluções viáveis e possíveis de ser efetuadas tendo em conta as restrições presentes no problema modelado. Se considerarmos contextos bastante dinâmicos (e.g. DARP dinâmico) em que os tempos de resposta devem de ser bastante rápidos, estas soluções são muitas vezes aceitáveis, pelo facto de serem uma solução possível de concretizar, embora não ótima e na qual a mesma foi obtida dentro de um intervalo de tempo estipulado.

Todos os algoritmos de pesquisa acima referidos foram testados e serão então apresentados os resultados dos mesmos na secção 5.1 (Testes dos algoritmos de procura de soluções), de forma tentar compreender se existe algum algoritmo que se demonstre superior para a resolução do problema de *routing* formulado, tendo em conta o valor obtido da função objetivo e diferentes configurações de testes.

4.2.2.4 Encontro de soluções "impossíveis"

Como já referido, por vezes poderá não ser possível encontrar soluções ótimas tendo em conta um intervalo de tempo de procura das mesmas. Poderão ainda ocorrer situações em que não seja possível a que o *Solver* encontre uma solução para o modelo de dados que se está a tentar solucionar. A impossibilidade de solução de um dado problema, geralmente advém da violação de alguma ou algumas das restrições impostas pelo mesmo. Note-se que basta que uma das restrições seja violada para que não seja possível a que o *Solver* da *Google OR-Tools* encontre uma solução. No entanto, existem medidas que podem ser tomadas de forma a "relaxar" o problema, tornando-o solucionável.

Uma destas medidas poderia ser o aumento do número de veículos existentes para a realização do serviço, isto leva a que as restrições temporais sejam mais facilmente cumpridas, pois simplesmente existem mais veículos para servirem os diferentes pedidos. No entanto, esta estratégia foi descartada, pois na realidade nem sempre é possível aumentar a oferta, uma vez que os fornecedores de transporte se encontram limitados por um determinado número de veículos que compõem a sua frota. Embora esta primeira hipótese tenha sido descartada, foram selecionadas duas outras abordagens que permitem de certa forma "relaxar" o problema de *routing* presente no modelo de dados de modo a torná-lo solucionável.

A primeira abordagem poderia ser: permitir a que *Solver* desista de certos pedidos de serviço, ou seja, pares de paragens "entrega" e "recolha", sendo adicionado um custo (penalidade) à função objetivo sempre que tal acontecesse. O valor deste custo, deve ser geralmente bastante elevado, de modo a que seja minimizado o número de pedidos recusados e consequentemente maximizado o número de pedidos satisfeitos. Assim, o *Solver* fica responsável por "relaxar" o problema tornando a sua resolução possível, ao avaliar todos os pares de paragens "recolha" e "entrega" existentes (que refletem os pedidos de serviço), sendo então servidos todos pedidos cujo seu serviço seja possível e, por sua vez, descartados os pedidos que não sejam possíveis servir. Para mais informações acerca da desistência nós e utilização de penalidades[4].

Esta abordagem funciona bem para um problema estático do DARP, onde todos os pedidos já são conhecidos inicialmente e existe tempo suficiente para notificar todos os clientes acerca da aceitação ou recusa do serviço para os mesmos. Contudo, para um sistema real de DARP dinâmico esta abordagem apresenta uma clara desvantagem, dada a possibilidade de recusar o serviço a clientes já foram anteriormente notificados da aceitação do serviço. Numa situação real não é aceitável pelo facto de esses clientes já terem sido notificados que seriam servidos.

[4] https://developers.google.com/optimization/routing/penalties

A outra medida que poderia também ser tomada é permitir o "relaxamento" dos intervalos de tempo em cada paragem, nos quais estes são "relaxados", sendo estes de certa forma alargados, ao adicionarmos um atraso ao valor máximo desse intervalo de tempo para uma dada paragem como é indicado pela Equação 23.

$$TimeWindows(i) = [TimeWindows(i,0), TimeWindows(i,1) + delay], \qquad (i) \in \mathbb{N}$$

Equação 23 - Relaxamento dos intervalos de tempo

Onde $TimeWindows(i)$ é a linha da matriz de intervalos de tempo para a paragem i, $TimeWindows(i,0)$ é o tempo mínimo para chegar na paragem i e $TimeWindows(i,1)$ é o tempo máximo para partir da paragem i e $delay$ é o atraso que será adicionado ao tempo máximo de partida.

Assim, uma vez que os intervalos de tempo mínimos e máximos em cada paragem, advém dos tempos de "recolha" e "entrega" dos diferentes clientes, o "relaxamento" dos intervalos de tempo em cada paragem, permite no fundo de certa forma "relaxar" o problema de *routing* ao serem aumentados os tempos de "entrega" de cada cliente por um determinado valor (atraso).

Inicialmente este atraso seria inicializado a zero, no caso de não ser possível formular uma solução para o modelo de dados atual, este atraso é "relaxado" sendo-lhe atribuído o valor de 1 minuto e sendo então atualizada a matriz de intervalos de tempos com esse novo atraso. Caso não seja possível resolver o problema com o atraso de 1 minuto a este atraso será adicionado mais 1 minutos, sendo isto efetuado até que seja possível resolver o problema ou até que seja atingido um atraso máximo de considerado (e.g. 30 minutos).

Para além deste atraso, é ainda, para cada paragem adicionada uma restrição "suave" que contém o tempo máximo original de partida dessa paragem, ou seja, o tempo antes de ser somado esse atraso. Esta restrição "suave" serve para indicar ao *Solver* para que garanta essa restrição numa dada paragem sempre que possível. Assim quando esta é garantida, o intervalo de tempo numa dada paragem i será o seu intervalo de tempo original, isto é, antes de ser somado o atraso ao tempo máximo desse intervalo, estando este então contido em $[TimeWindows_{min}(i), TimeWindows_{max}(i)]$. Caso essa restrição seja violada para uma dada paragem i, será adicionado um custo (penalidade) à função objetivo de forma a refletir a violação da mesma, sendo que o intervalo de tempo para essa mesma paragem considerará o atraso, no qual estará contido em $[TimeWindows_{min}(i), TimeWindows_{max}(i) + delay]$ como indicado pela Equação 23. Assim, o relaxamento do tempo máximo de partida das paragens, torna possível transformar um problema que inicialmente não era possível de resolver, num problema com solução, isto, caso o valor do atraso não exceda um dado limite definido (e.g. 30 minutos).

No entanto é notório que esta abordagem apresenta uma clara desvantagem, tendo esta um impacto na qualidade do serviço de um cliente ao serem alargados os intervalos de tempo das paragens. Para exemplificação, consideremos, um cliente cuja paragem de "entrega" teve o seu tempo máximo aumentado por 15 minutos, isto leva a que a "entrega" desse cliente possa no máximo chegar 15 minutos, depois do tempo desejado pelo mesmo para chegar a essa paragem destino, existindo então um visível impacto na qualidade do serviço. Embora esta seja evidentemente uma desvantagem, também é possível compreender que esta também trará

vantagens para a qualidade do serviço do cliente, uma vez que foi possível garantir o serviço do mesmo, algo que não seria possível se esta medida não fosse adotada. Ao considerarmos um sistema real, algum tipo de atraso irá sempre existir, sendo que ao utilizarmos esta medida, após a construção de uma solução de um percurso que serve um conjunto de clientes, os clientes que fossem aceites, poderiam ser notificados que o serviço poder-se-á atrasar em relação aos tempos estipulados pelos mesmo e conforme essa informação, estes aceitariam se queriam avançar com a requisição do seu serviço ou não.

4.2.2.5 Interpretando uma solução

Quando é possível resolver um dado problema de *routing* é possível então obter uma solução para o mesmo. A solução obtida através do *Solver* da *Google* irá apresentar os índices que representam as paragens a ser visitadas por cada veículo, como também as quantidades acumuladas (*Cumul*) para as diferentes dimensões (*RoutingDimensions*) ao longo de um percurso para um dado veículo.

A Figura 22 demonstra um exemplo de um problema de *routing* para dois clientes e um veículo no qual a unidade temporal considerada é o segundo.

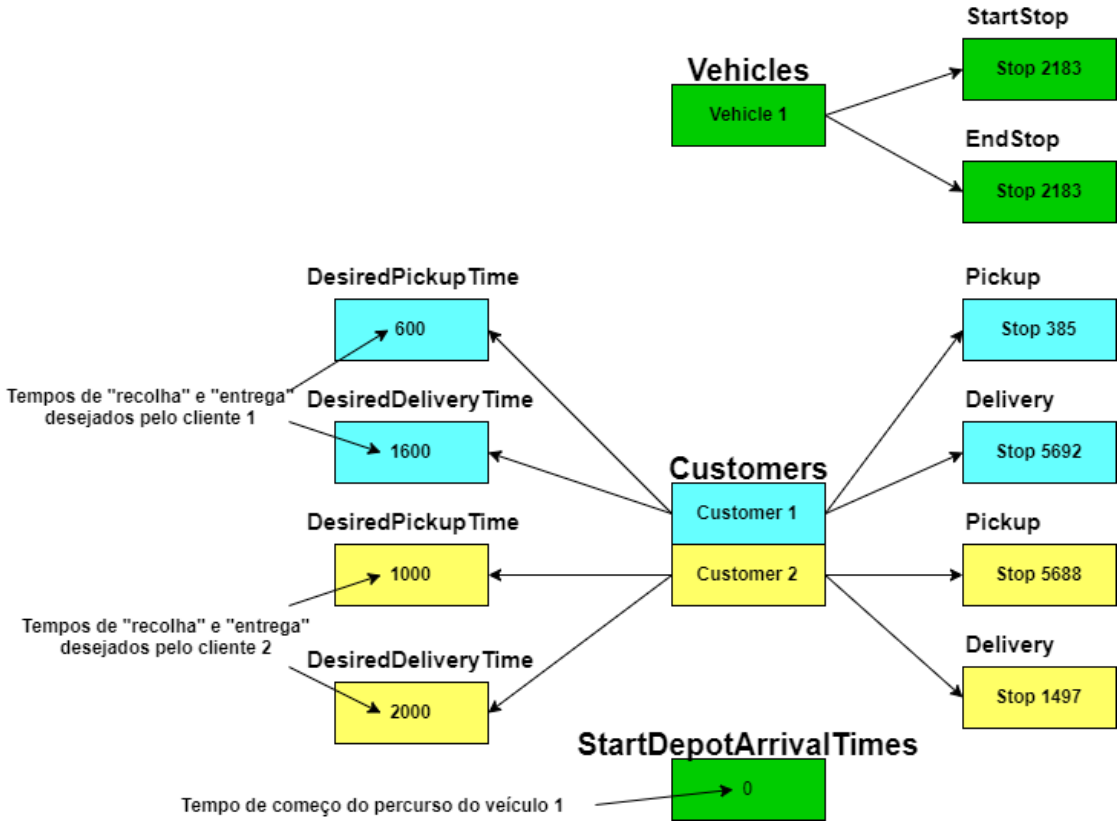

Figura 22 - Exemplo para um DARP no qual são considerados dois clientes e um único veículo

A partir do exemplo na Figura 22, é possível obter as seguintes listas (Figura 23) que estarão contidas no *IndexManager*, que por sua vez irá estar contido no modelo de dados do *Solver*:

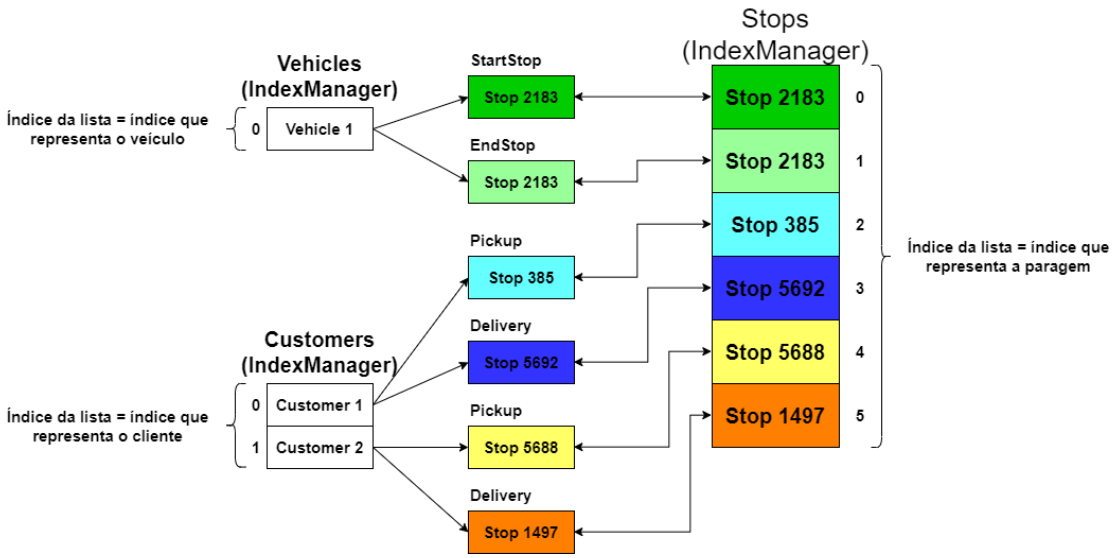

Figura 23 - Mapeamento entre os diferentes objetos (veículo, cliente e paragem) para índices que os representam

É importante notar que as soluções do *Solver* irão apresentar índices que representam as paragens, assim, de forma a podermos saber a paragem real que representa qualquer um desses índices, basta procurarmos o valor presente na posição indicada por esse índice na lista de paragens (*Stops*).

A Figura 24 representa a matriz de tempos de deslocamento (utilizando o segundo como unidade temporal), que será obtida para este problema que estará presente no modelo de dados.

Ambos os índices 0 e 1 representam a mesma paragem pois os tempos de deslocamento entre as diferentes paragens são os mesmos para ambos os casos

	0	1	2	3	4	5
0	0	0	268	551	297	132
1	0	0	268	551	297	132
2	268	268	0	560	425	299
3	551	551	560	0	287	681
4	297	297	425	287	0	429
5	132	132	299	681	429	0

TravelTimes (DataModel)

Figura 24 - Matriz de tempos de deslocamento para o problema em questão

A Figura 25 representa um exemplo de uma solução encontrada pelo *Solver* para este problema:

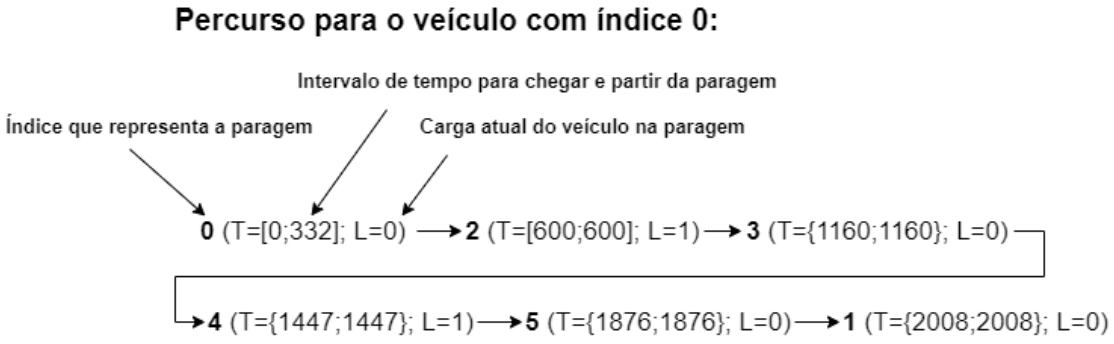

Figura 25 - Exemplo de solução encontrada pelo Solver para o problema em questão

Notemos que os intervalos de tempo T em cada paragem acima apresentados, podem ser vistos como a os valores acumulados temporais para estar e partir de cada paragem ou seja, a já referida (em 4.2.2.1) variável *Cumul* da dimensão temporal (*TimeDimension*). Relativamente a L, este representa basicamente a variável acumulada da capacidade atual do veículo numa dada paragem, ou seja, a variável *Cumul* para uma dada paragem da dimensão de capacidade (*CapacityDimension*) também já referida em 4.2.2.1.

Assim, a partir da Figura 25, é possível compreender que o veículo 1 irá começar o seu percurso na paragem com índice 0, ou seja, a paragem 2183, no qual permanecerá na mesma desde o tempo 0 até o tempo 332. É também possível verificar que a ocupação atual do veículo nesta paragem é 0 ($L = 0$). A partir do momento 332 o veículo parte da paragem 2183 e depois chega à paragem com índice 2 (paragem 385) no momento 600. É possível ver que nesta paragem (paragem 385) irá entrar um cliente no veículo, uma vez que a ocupação do veículo atual passa para 1 ($L = 1$). Se olharmos para a Figura 22, é possível perceber que o cliente que irá entrar será o cliente 1, que tem a paragem 385 como paragem de "recolha" e ainda o tempo mínimo desejado de "recolha" é 600, logo no momento em que o veículo chega a esta paragem, o cliente já se encontra na paragem e pode então ser "recolhido", pois o tempo de chegada do veículo à paragem é maior ou igual ao tempo desejado pelo cliente para ser "recolhido". Após a entrada do cliente, o veículo parte desta mesma paragem dirigindo-se então para a paragem com índice 3 (paragem 5692) e chegando à mesma no momento 1160. Quando este chega é possível ver que o cliente que estava anteriormente no veículo (cliente 1), irá sair nesta paragem, pois a ocupação atual do veículo passa para 0 ($L = 0$), e ainda é notório que o seu tempo desejado de "entrega" foi respeitado, pois o tempo de chegada a esta paragem (1160) é inferior ao tempo de "entrega" desejado pelo cliente em questão (1600). Após a saída do cliente 1, o veículo irá partir da paragem com índice 3 e tendo como destino a paragem com índice 4 (paragem 5688), no qual chegará no momento 1447. Nesta paragem o veículo irá "recolher" um cliente como é notório pelo aumento na ocupação atual do veículo ($L = 1$). Ao olharmos para a Figura 22 é possível perceber que o cliente que entra no veículo é o cliente 2, pois este tem a paragem atual (paragem 5688) como paragem de "recolha" e ainda o seu tempo mínimo desejado para ser "recolhido" é 1000, ou seja, inferior em relação ao tempo atual em que o veículo chega a esta paragem, logo o cliente pode ser "recolhido", pois já se encontra na paragem. De seguida o veículo parte da paragem com índice 4 e dirige-se então para a paragem de "entrega" do cliente 2, isto é, a paragem com índice 5 (paragem 1497). O veículo chega a esta

paragem no tempo 1876, na qual será então entregue o cliente 2, como é possível verificar pela diminuição na ocupação atual do veículo em relação à paragem anterior ($L = 0$). É ainda possível ver que o tempo de "entrega" desejado pelo cliente 2 será respeitado, uma vez que o tempo de chegada do veículo à paragem 1497 (1876) é inferior ao tempo de "entrega" desejado pelo cliente (2000). Por fim, o veículo depois parte para a sua paragem final, com índice 1 (2183) chegando à mesma no momento 2008 e terminando assim o seu percurso.

Como já referido os tempos de partida e chegada às diferentes paragens são obtidos pelo Solver ao utilizar a matriz de tempos de deslocamento (*TravelTimes*). De forma a constatar isto, consideremos por exemplo, o deslocamento entre a paragem com índice 0 e a paragem com índice 2. Se subtrairmos o tempo de chegada à paragem com índice 2 com o tempo de partida paragem com índice 0, obtemos então o tempo de deslocamento entre estas duas paragens, que será: $600 - 332 = 268$. Ao olharmos para o índice da linha 0 e índice da coluna 2 da matriz de tempos de deslocamento é possível ver que o tempo de deslocamento presente na mesma é também 268, sendo assim então respeitado o tempo de deslocamento entre ambas as paragens. Se olharmos para qualquer um dos outros deslocamentos entre as diferentes paragens o mesmo é também verificável. Com isto, é então possível compreender que esta matriz foi utilizada corretamente pelo *Solver* para o cálculo dos tempos de chegada e partida entre as diferentes paragens.

Note-se ainda, que estas soluções são geralmente consideradas numa situação perfeita no qual não existem atrasos e os quais os tempos de deslocamento dos veículos não variam, pois é considerado que os veículos se deslocam a uma velocidade média constante e não são considerados atrasos em relação à chegada e partida das diferentes paragens. Ao olharmos para a paragem com índice 2 por exemplo, é possível ver que um cliente entra no veículo e que o momento de chegada e partida da paragem do veículo a essa paragem é o mesmo, não sendo então tido em conta os tempos inerentes à entrada de um dado cliente num veículo. Com a utilização da simulação será possível testar estas soluções utilizando um contexto mais real e no qual poderão existir atrasos em relação aos tempos de chegada e partida das diferentes paragens.

4.2.2.6 Tradução de índices da solução para os objetos da simulação

Uma vez que a solução obtida pelo *Solver* apenas apresenta os índices dos diferentes objetos como foi possível constatar no subcapítulo anterior (4.2.2.5), torna-se necessário então converter esses índices para os objetos reais que os mesmos representam. Assim, houve a necessidade de criar o *RoutingSolutionObject*, que é um objeto que pega nos dados de uma dada solução construída pelo *Solver*, e os transforma num conjunto de dados que possam ser utilizados na simulação. Neste objeto todos os índices presentes na solução do *Solver* são traduzidos para os respetivos objetos que os mesmos representam, através do *IndexManager* presente no modelo de dados, que reflete o problema que foi solucionado. Este objeto é depois responsável por construir um dicionário chave-valor, no qual a chave será o objeto de cada veículo considerado no problema e o seu valor será o seguinte conjunto de informações referentes a cada um desses veículos considerados, baseando-se na solução encontrada pelo *Solver*. Para um dado veículo considerado presente numa solução providenciada pelo *Solver* são contidas as seguintes informações:

- **O conjunto de paragens (VehicleStops)** – lista que contém o conjunto de paragens (objetos) que terão que ser visitadas pelo veículo considerado. Esta lista é construída a partir da solução gerada ao traduzir cada índice da paragem para o objeto paragem que o mesmo representa e sendo estas inseridas sequencialmente, conforme aparecem na solução do percurso de um dado veículo;
- **O conjunto de intervalos de tempo (VehicleExpectedTimeWindows)** – lista que contém o conjunto de todos os intervalos de tempo para cada paragem contida no percurso do veículo considerado. É construída ao inserir sequencialmente cada par intervalo de tempo contido na solução dada pelo *Solver*. Note-se que, para um mesmo veículo esta lista terá o mesmo tamanho que a lista do conjunto de paragens, na qual cada índice desta mesma lista representa a respetiva paragem no índice da lista do conjunto de paragens;
- **O conjunto de clientes (VehicleCustomersToBeServed)** – lista que contém o conjunto de cliente que terão que ser servidos pelo veículo considerado. De forma a construir este conjunto é necessário ser verificado se o percurso gerado (solução) para um dado veículo contém o par de paragens "entrega" e "recolha" de um dado cliente presente no modelo de dados e se a paragem de "recolha" é visitada anteriormente à paragem "entrega" (restrição de precedência) e ainda se o intervalo de tempo mínimo para "recolher" um cliente é respeitado. Caso isto se verifique para um dado cliente, esse cliente será então adicionado a lista do veículo em questão.

Estas informações são depois utilizadas na simulação, ao atualizar (se necessário) o percurso (*Route*), intervalos de tempo esperados (*ExpectedTimeWindows*) e clientes a serem servidos (*CustomersToBeServed*) para cada um dos veículos presentes no contexto (atual) da simulação.

5 Resultados

5.1 Testes dos algoritmos de procura de soluções

Os já referidos algoritmos de pesquisa presentes na biblioteca *OR-Tools*, foram testados de forma a poder avaliá-los, compara-los e também compreender de que modo é que estes são influenciados por certas características inerentes a diferentes instâncias para o mesmo tipo de problemas de *routing*.

Os testes foram efetuados utilizando a rede de transporte público da cidade do Funchal para uma perspetiva estática. Foram gerados um conjunto de pedidos iniciais e utilizado o *Solver* para encontrar e otimizar uma solução, para cada um dos diferentes algoritmos de procura de soluções considerados, dentro de um certo intervalo de tempo definido. Estes testes foram efetuados, utilizando as mesmas circunstâncias e configurações utilizadas no início da simulação, de forma a que os resultados obtidos se aproximem o melhor possível em relação ao contexto em que o *Solver* será utilizado.

Relativamente aos modelos de dados utilizados nos testes, que serão resolvidos pelo *Solver*, as configurações dos mesmos eram as seguintes:

- **Número de clientes (pedidos) iniciais** – o número de clientes foi variável, de forma a compreender o impacto que os mesmos têm nas soluções obtidas, tendo sido efetuados testes para instâncias de 25, 50, 75 e 100 clientes;
- **Intervalo de tempo do problema de *routing*** – foi considerada uma duração fixa de 4 horas para cada problema de *routing*;
- **Número de veículos** – o número de veículos considerados para o problema foi fixo, tendo sido considerado que existem 20 veículos disponíveis para a resolução dos diferentes problemas de *routing*. Note-se que este número era mais que suficiente para o número de clientes considerados para a maioria dos casos;
- **Capacidade dos veículos** – foi considerada uma frota homogénea, na qual todos os veículos tinham a capacidade máxima de 20 clientes;
- **Velocidade de deslocamento dos veículos** – foi considerado que todos os veículos de deslocam à mesma velocidade média de deslocamento de 40 km/h;
- **Paragem de começo e termino** – todos os veículos começavam e terminavam o seu percurso na mesma paragem (*Depot*).

Para cada teste efetuado foram utilizadas as seguintes configurações no *Solver*:

- **Algoritmo de encontro primeira solução** – foi sempre utilizado o algoritmo *Parallel Cheapest Insertion* para a geração da solução inicial, tendo esta solução geralmente um custo relativamente alto e não otimizada;
- **Algoritmo de procura de soluções** – foram variados os diferentes algoritmos de procura de soluções, contidos na biblioteca *OR-Tools*, de forma a tentar compreender a influência que cada um têm nas soluções geradas em um dado intervalo de tempo definido. Como já referido, os algoritmos de pesquisa contidos nesta biblioteca incluem: *Greedy Descent*, *Guided Local Search*, *Simulated Annealing*, *Tabu Search* e *Generic Tabu Search.* Note-se que enquanto o algoritmo de primeira solução, é responsável por gerar uma solução inicial, o algoritmo de procura de soluções é responsável por otimizar essa

solução, ou seja, este é responsável por procurar novas soluções com custos inferiores ao da inicial dentro de um dado intervalo de tempo;

- **Tempo de procura de soluções –** o tempo de procura de soluções foi ainda variado de forma a compreender a influência que este tem nas soluções encontradas para cada um dos algoritmos considerados. Os tempos de procura considerados iam desde os 5 segundos até os 60 segundos para as instâncias de teste de 25 até 75 clientes e até 90 segundos para as instâncias de teste de 100 clientes;
- **Tempo máximo de viagem relativo dos clientes** – o tempo máximo de viagem relativo de cada um dos clientes considerados para cada um dos problemas de *routing* foi fixo, sendo este valor 30 minutos. Ou seja, o tempo de viagem de cada cliente não poderia ultrapassar a soma de 30 minutos com tempo de viagem direto referente ao mesmo, isto é, tempo mínimo de viagem se a "entrega" do cliente fosse efetuada diretamente desde a paragem "recolha" até à paragem "entrega";
- **"Relaxamento" dos intervalos de tempo** - foi definido o tempo máximo de 30 minutos de atraso aceitável em relação ao tempo de "entrega" (na paragem destino), originalmente desejado pelo cliente, de forma a "relaxar" certos problemas tornando-os possíveis de resolver;

Para cada número de clientes considerados (25, 50, 75 e 100) foram gerados 10 modelos de dados distintos, nos quais eram considerados o já referido número fixo de 20 veículos, a duração fixa (do problema) de 4 horas e o tempo máximo de viagem relativo dos clientes de 30 minutos. Com isto, foram então obtidos no total, 40 modelos de dados distintos (10 modelo de dados distintos para cada um dos diferentes números de clientes considerados).

Note-se que a construção de cada um destes modelos de dados (e informações referentes aos mesmos), seguiu a mesma abordagem que a seguida no início da simulação, na qual para cada cliente considerado, tanto a sua paragem de "entrega" e "recolha" como também os tempos desejados para estarem ou partirem das mesmas, foram gerados aleatoriamente e dentro de um intervalo de tempo considerado (intervalo de tempo do problema). É importante ainda referir que cada um destes modelos de dados obtidos, irá ser distinto em relação aos outros todos, mesmo quando o número de clientes considerado seja o mesmo, pois a localização das paragens de "entrega" e "recolha" dos diferentes clientes irão variar, como também os seus tempos desejados para serem "entregues" ou "recolhidos" nas mesmas, permitindo assim refletir diferentes situações, geradas aleatoriamente, para um mesmo tipo de problema. Para além disto, é ainda importante referir que cada um dos modelos de dados gerados para cada uma das diferentes configurações, eram modelos que refletiam um problema de *routing* válido e possível de resolver pelo *Solver*, isto é, era possível obter pelo menos uma solução para cada um dos problemas existentes em cada modelo de dados, e sendo então possível servir todos os clientes considerados.

Após a obtenção dos já referidos 40 modelos de dados (distintos), cada um destes foi testado, utilizando cada um dos diferentes algoritmos de procura de soluções considerados e, também, para cada um dos diferentes tempos de procura considerados, sendo então guardados os resultados referentes às soluções obtidas para cada um desses testes. Os testes foram efetuados utilizando um computador equipado com um processador Intel core i5 4210U e 4GB de memória RAM.

A métrica escolhida de forma a poder comparar os resultados dos diferentes algoritmos de pesquisa foi o valor da função objetivo, que permite representar o custo total acumulado da solução gerada pelo *Solver*, permitindo assim de certa forma medir as diferentes soluções obtidas. Como já referido, o objetivo do *Solver* é encontrar uma solução viável (em um dado intervalo de tempo) e possível de forma a minimizar o custo, utilizando algum dos algoritmos de pesquisa existentes para otimização dessa solução. Assim, esta métrica irá permitir medir as soluções obtidas por cada algoritmo de pesquisa considerado, possibilitando assim compará-los. De notar que o objetivo dos algoritmos de pesquisa é ir de encontro a melhores soluções, de forma a minimizar os custos totais acumulados (valor da função objetivo) de uma dada solução.

5.1.1 Tempo de procura de soluções

De forma a compreender o impacto que o tempo de procura de soluções tem em cada um dos algoritmos considerados, serão brevemente apresentados os resultados médios obtidos dos testes para cada um dos diferentes números de clientes considerados, isto é, para 25, 50, 75 e 100 clientes. A tabela com os dados dos resultados obtidos para cada um dos números de clientes considerados encontra-se na secção dos anexos (Anexo 1 - Tabela com os dados do valor médio da função objetivo para os testes efetuados para 25, 50, 75 e 100 clientes).

A Figura 26 demonstra um gráfico que representa o valor da função objetivo para cada algoritmo em função do tempo de procura de soluções dos resultados obtidos para as instâncias de testes de 25 clientes.

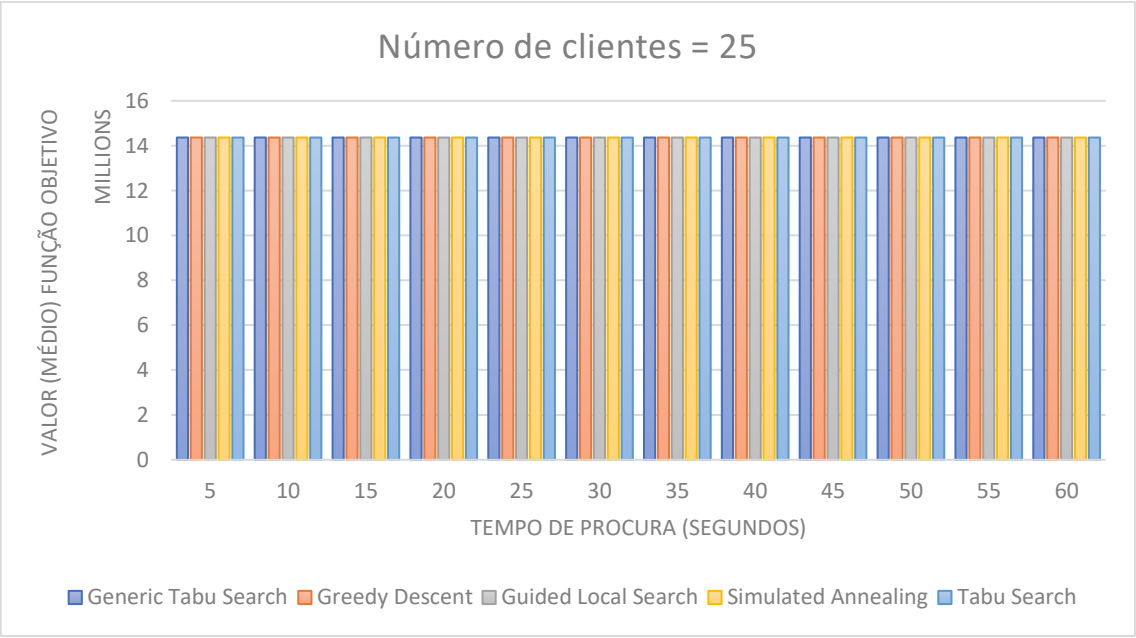

Figura 26 - Valor (médio) da função objetivo para cada algoritmo em função do tempo de procura para 25 clientes

A partir da Figura 26 é possível compreender que para problemas de *routing* de 25 clientes, os algoritmos de procura de soluções, de modo geral, comportam-se de forma semelhante. Os valores médios da função objetivo são relativamente aproximados para qualquer um dos tempos de procura considerados e não existe variação visível no valor médio

da função objetivo ao longo do tempo. Assim, acreditamos que para problemas de 25 clientes, a escolha do algoritmo utilizado acabará por não ter grande impacto na solução gerada, uma vez que as soluções geradas por cada um dos diferentes algoritmos são relativamente semelhantes.

Para além disto, é ainda possível verificar, que para esta mesma situação, o tempo de procura mínimo de 5 segundos é suficiente para resolver o problema, uma vez que foi observado que o valor médio da função objetivo se mantém relativamente semelhante e constante ao longo do tempo para qualquer um dos algoritmos considerados. Assim, não existe um ganho na qualidade da solução gerada se utilizarmos tempos de procura superiores.

A Figura 27 reflete a representação gráfica dos resultados obtidos para os valores médios da função objetivo para cada algoritmo em função do tempo de procura, quando são consideradas as instâncias de teste para 50 clientes.

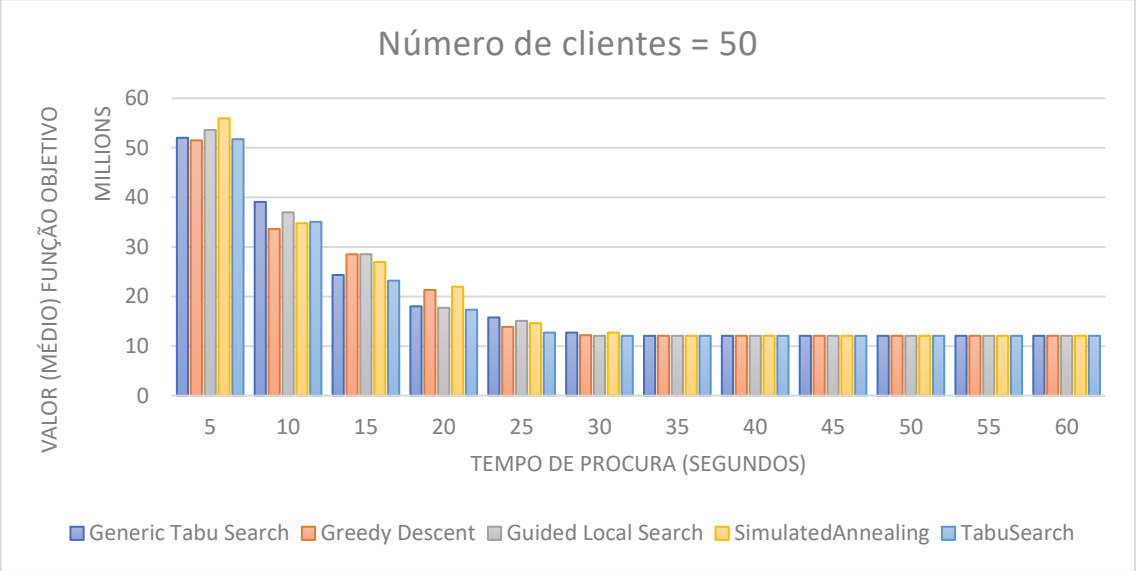

Figura 27 - Valor (médio) da função objetivo para cada algoritmo, em função do tempo de procura para 50 clientes

Quando consideramos os testes efetuados para 50 clientes, ao olharmos para a Figura 27 é notório que o aumento do tempo de procura possibilita reduzir o valor (médio) da função objetivo para qualquer um dos algoritmos considerados, permitindo então a obtenção de melhores soluções. No entanto é também observável que existe um dado momento, no qual já não é possível melhorar mais uma dada solução, uma vez que a partir desse momento o valor médio da função objetivo estabiliza, isto é, se mantém aproximadamente constante.

É importante notar que embora o valor médio da função objetivo se mantenha constante (estabilize), isto não quer dizer necessariamente que o *Solver* encontrou a solução ótima para o problema atual (mínima possível). Isto é, apenas indicativo que não é possível aos algoritmos de pesquisa melhorar as soluções obtidas, atendendo ao tempo máximo de procura considerado para estes testes.

A Tabela 8 - Percentagem em relação ao valor médio da função objetivo obtido inicialmente (em t = 5 segundos) para os diferentes algoritmos, tempos de procura e para 50 clientes permite comparar a qualidade das diferentes soluções para 50 clientes, utilizando os

diferentes tempos de procura e algoritmos, em relação à função objetivo obtida inicialmente, quando o tempo de procura é 5 segundos (para efeitos de comparação, considera-se que o custo máximo é obtido para t = 5s). Note-se que para os testes efetuados, foi considerado que a função objetivo estabiliza a partir do momento em que a sua variação em relação ao valor obtido para o tempo de procura posterior não seja superior a 1%, durante pelo menos três tempos de procura consecutivos (a contar com o próprio). O valor da função objetivo a partir do qual é considerado que a função a objetivo estabiliza encontra-se representado a negrito na seguinte tabela para cada um dos diferentes algoritmos:

Percentagem em relação ao valor médio da função objetivo inicial (para t = 5 s) quando são considerados 50 clientes					
			Algoritmo		
Tempo de procura (segundos)	**Generic Tabu Search**	**Greedy Descent**	**Guided Local Search**	**Simulated Annealing**	**Tabu Search**
5	100%	100%	100%	100%	100%
10	75%	65%	69%	62%	68%
15	47%	55%	53%	48%	45%
20	35%	41%	33%	39%	33%
25	30%	27%	28%	26%	25%
30	**24%**	**24%**	**23%**	**23%**	**23%**
35	23%	23%	23%	22%	23%
40	23%	23%	23%	22%	23%
45	23%	23%	23%	22%	23%
50	23%	23%	23%	22%	23%
55	23%	23%	23%	22%	23%
60	23%	23%	23%	22%	23%

Tabela 8 - Percentagem em relação ao valor médio da função objetivo obtido inicialmente (em t = 5 segundos) para os diferentes algoritmos, tempos de procura e para 50 clientes

A partir da Tabela 8, é possível compreender em que momento é que cada um dos algoritmos estabiliza o seu valor médio da função objetivo. Dito isto, é observável que quando são considerados 50 clientes, quase todos os algoritmos começam a estabilizar relativamente no mesmo momento, isto é, quando o tempo de procura é 30 segundos. A partir deste momento é possível perceber que se aumentarmos o tempo de procura, não existe grande variação no valor encontrado por cada um dos algoritmos. Para todos os algoritmos, a partir dos 35 segundos, os valores obtidos da função objetivo não sofrem qualquer alteração (percentual) notória em relação a qualquer tempo de procura posterior ao mesmo. Estas observações indicam então que não existem grandes ganhos na qualidade das soluções obtidas se utilizarmos um tempo de procura para além de 30 segundos, para a situação aqui considerada.

Relativamente às soluções obtidas por cada algoritmo é possível perceber que a partir do momento em que cada um estabiliza, as soluções encontradas são relativamente

semelhantes, pois o valor da função objetivo é relativamente aproximado, indicando assim que a qualidade das diferentes soluções encontradas a partir do momento em que cada algoritmo estabiliza, serão semelhantes (para 50 clientes). Um outro aspeto que é ainda possível observar, é que a partir do momento em que existe a referida estabilização, em qualquer um dos algoritmos considerados, o valor inicial da função objetivo é otimizado em média em pelo menos 75%, existindo assim uma melhoria significativa em relação à solução encontrada inicialmente.

Assim com base nos resultados aqui apresentados e discutidos acreditamos que qualquer um dos algoritmos considerados, demonstram-se bons candidatos para a resolução de problemas de *routing* com 50 clientes, pelo facto de todos estes apresentarem um desempenho semelhante, e estabilizando aproximadamente no mesmo momento. Relativamente ao tempo de procura ótimo, com base nestes resultados este deve estar entre os 30 e 35 segundos para qualquer um dos algoritmos considerados. Para além destes tempos, os ganhos a nível da qualidade da solução são mínimos ou quase nulos.

A Figura 28 apresenta o gráfico que reflete os resultados obtidos dos valores médios das funções para cada algoritmo considerado em função do tempo de procura, para as instâncias de teste de 75 clientes.

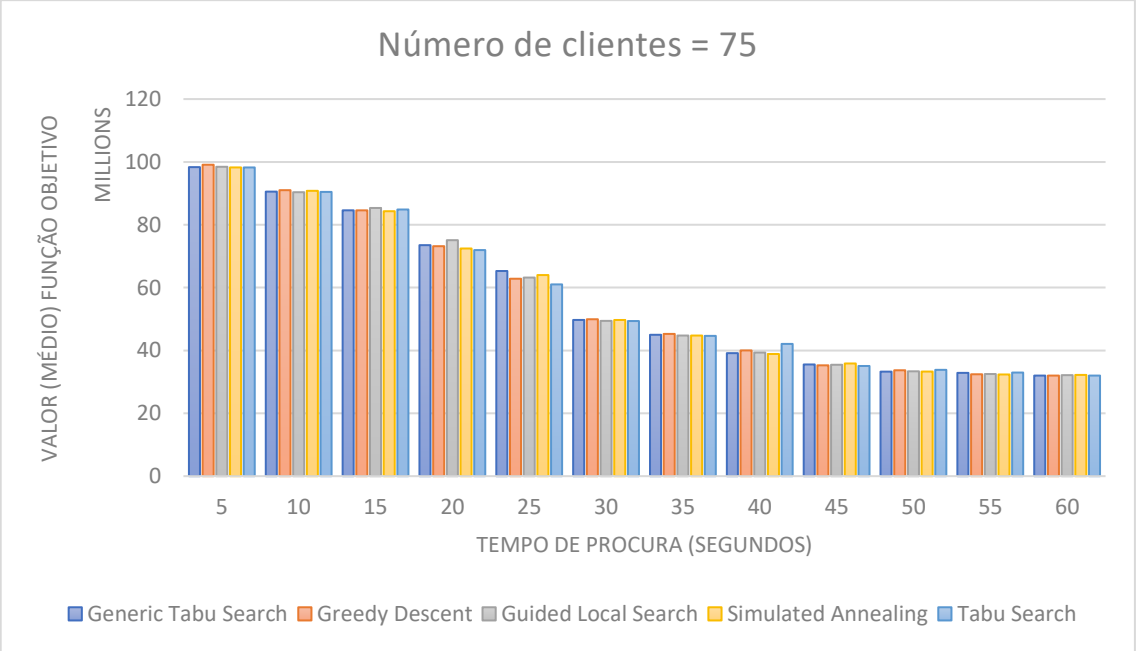

Figura 28 - Valor (médio) da função objetivo para cada algoritmo em função do tempo de procura para 75 clientes

A partir do gráfico presente na Figura 28 que retrata os resultados obtidos para os testes efetuados para 75 clientes, é possível também constatar que o valor da função objetivo vai sendo otimizado ao longo do tempo, decrescendo então com o aumento do tempo de procura de soluções e chegando a uma dada altura que o valor médio da função objetivo começa então a estabilizar, existindo então cada vez menos diferença em relação aos valores da função objeto para o tempo de procura anterior em relação ao atualmente considerado.

A Tabela 9 demonstra a qualidade das diferentes soluções obtidas em relação ao valor (médio) da função objetivo obtido inicialmente (quando o tempo de procura é 5 segundos), nas instâncias de teste de 75 clientes, para cada um dos diferentes algoritmos e tempos de procura considerados. Note-se que o valor da função objetivo a partir do qual é considerado que a função a objetivo estabiliza encontra-se representado a negrito.

Percentagem em relação ao valor médio da função objetivo inicial (para t = 5 s) quando são considerados 75 clientes					
	Algoritmo				
Tempo de procura (segundos)	**Generic Tabu Search**	**Greedy Descent**	**Guided Local Search**	**Simulated Annealing**	**Tabu Search**
5	100%	100%	100%	100%	100%
10	92%	92%	92%	92%	92%
15	86%	85%	87%	86%	86%
20	75%	74%	76%	74%	73%
25	66%	63%	64%	65%	62%
30	51%	50%	50%	51%	50%
35	46%	46%	45%	46%	45%
40	40%	40%	40%	40%	43%
45	36%	36%	36%	36%	36%
50	**34%**	**34%**	**34%**	**34%**	**34%**
55	33%	33%	33%	33%	34%
60	33%	32%	33%	33%	33%

Tabela 9 - Percentagem em relação ao valor médio da função objetivo obtido inicialmente (em t = 5 segundos) para os diferentes algoritmos e tempos de procura e para 75 clientes

Ao observarmos a Tabela 9 é possível também verificar que todos os algoritmos começam a estabilizar mais ou menos no mesmo momento, sendo isto semelhante ao que acontecia para 50 clientes. No entanto, o momento considerado ótimo no qual estes parecem estabilizar é distinto em relação ao verificado anteriormente (para 50 clientes), sendo este superior e cerca de 50 segundos. Note-se que embora o valor da função objetivo varie um pouco passado os 50 segundos, o ganho na qualidade da solução é relativamente baixo (no máximo 1%), sendo esta variação relativamente mínima, em relação ao valor da função objetivo anterior, daí ter sido considerado este tempo como ótimo. No entanto, seria interessante ter sido aumentado o tempo de procura considerado para estes testes de forma a tentar encontrar um momento a partir do qual não haja mais esta variação a nível percentual no valor da função objetivo, como foi possível observar nos testes efetuados para 25 e 50 clientes.

Relativamente à qualidade das soluções obtidas, no momento a partir do qual os algoritmos estabilizam, existe uma melhoria em cerca de 65% em relação à solução inicial (em t = 5 segundos).

Assim, com isto acreditamos que para 75 clientes, a escolha do algoritmo a ser utilizado possa também ser arbitrária, pois para os testes efetuados, todos demonstraram resultados relativamente semelhantes. No que toca ao tempo de procura de soluções, este deve ser de pelo menos 50 segundos para qualquer um dos algoritmos, de forma a obter uma solução otimizada em cerca 65% em relação à solução obtida inicialmente.

A Figura 29 apresenta o valor da função obtido para cada um dos algoritmos em função do tempo de procura para os testes efetuado para 100 clientes. Note-se que para estes testes, os tempos de procura foram estendidos até 90 segundos, visto que o momento de estabilização considerado para os testes efetuados para 75 se demonstrar relativamente próximo do último tempo de procura limite considerado anteriormente (de 60 segundos).

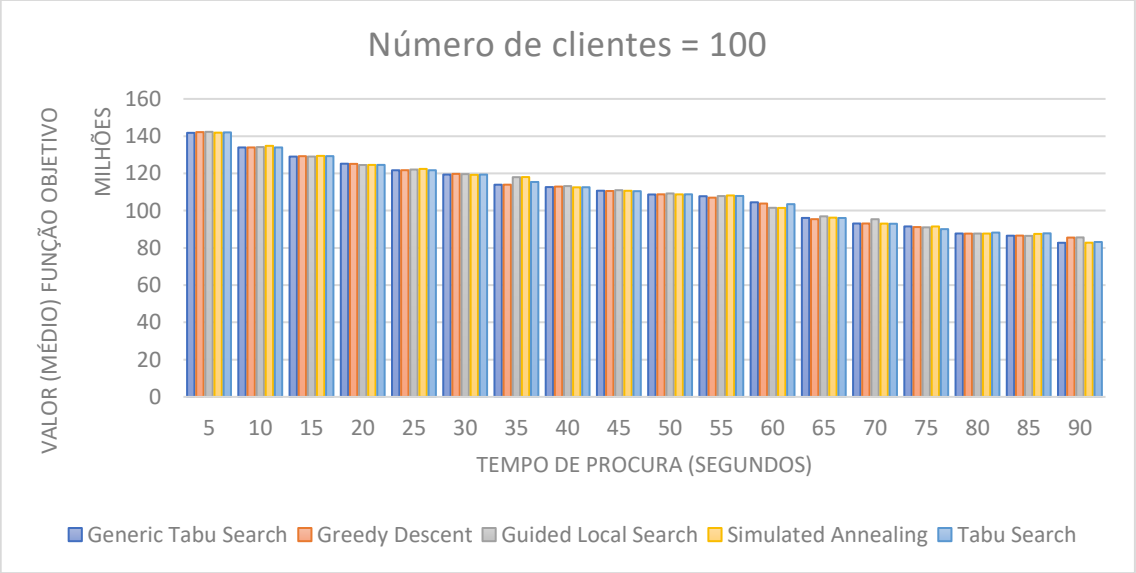

Figura 29 - Valor (médio) da função objetivo de cada algoritmo em função do tempo de procura para 100 clientes

A partir da Figura 29, é também observável que para as instâncias de teste de 100 clientes, o valor médio da função objetivo diminui com o aumento do tempo de procura, sendo isto semelhante ao que foi observado para os testes efetuados para 50 e 75 clientes. Também é notório que a diferença entre o valor médio da função objetivo para um dado tempo de procura em relação ao tempo de procura imediatamente anterior é relativamente mais baixo, indicando assim que estes problemas são relativamente mais complexos que os considerados anteriormente, resultando numa maior necessidade a nível de temporal e computacional para efetuar a otimização.

A Tabela 10 permite comparar a qualidade das soluções geradas por cada um dos algoritmos com a solução obtida inicialmente (quando o tempo de procura é 5 segundos) para esse mesmo algoritmo.

Percentagem em relação ao valor médio da função objetivo inicial (para t = 5 s) quando são considerados 100 clientes					
	Algoritmos				
Tempo de procura	**Generic Tabu Search**	**Greedy Descent**	**Guided Local Search**	**Simulated Annealing**	**Tabu Search**
5	100%	100%	100%	100%	100%
10	94%	94%	94%	95%	94%
15	91%	91%	91%	91%	91%
20	88%	88%	87%	88%	88%
25	86%	86%	86%	86%	86%
30	84%	84%	84%	84%	84%
35	80%	80%	83%	83%	81%
40	79%	79%	80%	79%	79%
45	78%	78%	78%	78%	78%
50	77%	77%	77%	77%	77%
55	76%	75%	76%	76%	76%
60	74%	73%	71%	72%	73%
65	68%	67%	68%	68%	68%
70	66%	65%	67%	66%	65%
75	65%	64%	64%	64%	63%
80	62%	62%	62%	62%	62%
85	61%	61%	61%	62%	62%
90	58%	60%	60%	58%	59%

Tabela 10 - Percentagem em relação ao valor médio da função objetivo obtido inicialmente (em t = 5 segundos) para os diferentes algoritmos e tempos de procura e para 100 clientes

Se observamos a diferença entre as soluções geradas para cada um dos algoritmos tendo em conta os diferentes tempos de procura, é possível ver que estas diferenças são relativamente muito inferiores em relação às observadas para os testes efetuados para 50 ou 75 clientes. O critério utilizado anteriormente para estabelecer em que momento é que um dado algoritmo estabiliza, no qual se o valor médio da função objetivo de um dado tempo não for superior a 1% em relação ao tempo imediatamente superior, não pode ser utilizado para esta situação, pois as mudanças em relação à função objetivo obtida inicialmente entre cada um dos tempos de procura são relativamente inferiores, em relação às encontradas nos testes anteriores (para 50 e 75 clientes).

Para além disto, embora tenha sido estendido o intervalo de tempo dos testes efetuados até aos 90 segundos, acreditamos que qualquer um dos algoritmos não chegou ainda ao seu momento de estabilização. No entanto, estes algoritmos parecem estar relativamente próximos desse momento, pois, as diferenças, entre os diferentes tempos de procura parecem ser cada vez menores quanto mais próximos estes se encontram dos 90 segundos.

Se considerarmos o tempo de procura de 90 segundos e qualquer um dos algoritmos, é possível perceber que as soluções são em média otimizadas (melhoradas) em cerca de 40% em relação à solução obtida inicialmente. Com isto, comparativamente à otimização observada em 50 ou 75 clientes, é notório que a otimização efetuada é cada vez menor com o aumento do número de clientes e considerando os mesmos tempos de procura, indicando assim que os problemas de *routing* ficam cada vez mais complexos e difíceis de otimizar.

Com o observado acreditamos que para 100 clientes, a escolha do algoritmo possa também ser arbitrária, uma vez que todos apresentam soluções relativamente semelhantes. No que toca ao tempo de procura, estes resultados indicam que o intervalo de tempo considerado (de 5 até 90 segundos) não é suficiente para atingir o momento da estabilização. Com isto, só podemos inferir que este tempo deva ser no mínimo superior a 90 segundos.

5.1.2 Número de clientes

A Tabela 11 apresenta um resumo dos resultados obtidos que permite comparar os algoritmos que apresentam em média os melhores e piores resultados para cada um dos números de clientes considerados. Note-se que o valor da função objetivo apresentado para cada um dos algoritmos é a média efetuada para todos os tempos de procura do número de clientes considerado.

Número de clientes	Melhor valor da função objetivo	Pior valor da função objetivo	Algoritmo com melhor resultado médio	Algoritmo com pior resultado médio	Diferença (%) do melhor resultado em relação ao pior
25	14363463	14364350	Guided Local Search	Greedy Descent	-0.01%
50	18709989	19955261	Tabu Search	Simulated Annealing	-6.24%
75	56374158	56651982	Tabu Search	Generic Tabu Search	-0.49%
100	109278439	109756456	Generic Tabu Search	Guided Local Sarch	-0.44%
Média Geral	56317416	56736835	Tabu Search	Guided Local Search	-0.74%

Tabela 11 - Tabela resumo com os algoritmos que demonstram em média os melhores e piores resultados

Ao olharmos para a tabela acima apresentada (Tabela 11), é possível perceber que em geral a diferença (em média) do melhor valor da função objetivo em relação ao pior é relativamente baixo, sendo este geralmente inferior a 1%, se considerarmos os números de clientes 25, 75 e 100. Isto indica que o algoritmo que obtém em média os melhores resultados para um dado tempo de procura irá ter um desempenho relativamente semelhante ao que obtém os piores, indicando então que em média as soluções encontradas por cada um destes algoritmos serão relativamente semelhantes. Dito isto, acreditamos que para 25, 75 e 100

clientes qualquer um dos algoritmos de procura de soluções possa ser utilizado, sustentando então o observado anteriormente quando foram analisados os resultados para cada um dos números de clientes distintos, algoritmos e tempos de procura considerados.

No entanto se considerarmos os testes efetuados para 50 clientes, é possível perceber que existe uma diferença mais acentuada em relação aos melhores e piores resultados médios. Para esta situação, é possível compreender que o algoritmo *Tabu Search* é o que apresenta em média os melhores resultados. Também, para esta mesma situação, é possível constatar que o algoritmo *Simulated Annealing* é o que apresenta em média os piores resultados, e a diferença entre os valores obtidos da função objetivo, para este algoritmo e o algoritmo que melhores resultados apresenta é de cerca de 6%. Com base nestes resultados, acreditamos então que o algoritmo *Tabu Search*, deva ser utilizado para problemas de *routing* nos quais são considerados 50 clientes.

Ao olharmos para as médias gerais é observado que a o valor médio da função objetivo do algoritmo, que obtém em média os piores resultados, diferem em cerca de 0.74% em relação ao que obtém os melhores resultados, sendo esta diferença relativamente insignificante. Com base nisto e no que já foi discutido anteriormente, acreditamos que qualquer um dos algoritmos considerados, irão obter em média, resultados relativamente semelhantes.

A Tabela 12 permite comparar o valor médio da função objetivo em t = 60 segundos, em relação ao valor médio da função objetivo obtido inicialmente em t = 5 segundos para os diferentes números de clientes. Note-se que os valores aqui apresentados são as médias efetuadas aos valores da função objetivo de todos os algoritmos, para o tempo de procura e número de clientes em questão. A otimização média é calculada usando a seguinte formula:

$$\frac{f_{t=5s} - f_{t=60s}}{f_{t=5s}}$$

Onde $f_{t=5s}$, é o valor médio da função objetivo para t = 5 segundos, e $f_{t=60s}$ é o valor médio da função objetivo para t = 60 segundos.

Número de clientes	Valor médio função objetivo para t = 5 segundos	Valor médio função objetivo para t = 60 segundos	Otimização média (%) em relação ao valor para t = 5 segundos
25	14364297	14364021	0.00%
50	52969724	12065812	77.22%
75	98477290	32077138	67.43%
100	142034780	102956260	27.51%

Tabela 12 - Comparação entre os valores médios da função objetivo obtidos para o tempo de procura de 60 segundos em relação aos valores obtidos inicialmente (para t = 5 segundos), para cada um dos números de clientes considerados

Ao observarmos a Tabela 12, é possível compreender que em média não existe algum tipo de otimização efetuada para 25 clientes, pois a diferença entre a média dos valores para um tempo de procura de 60 segundos e um tempo de procura de 5 segundos é mínima, indicando assim que para os testes efetuados, 5 segundos é mais que suficiente para a otimização de problema de *routing* para 25 clientes. Ao olharmos para 50 clientes, é notório que

existe uma otimização média de cerca de 77% em relação ao valor obtido inicialmente (em t = 5 segundos). Relativamente a 75 clientes, é observável que existe uma otimização média de cerca de 67%, sendo esta otimização inferior à observada anteriormente para 50 clientes. Por fim, ao olharmos para 100 clientes, é possível perceber que existe uma otimização média de cerca de 27%, sendo esta muito inferior em relação à otimização observada anteriormente para 50 e 75 clientes. Com isto é possível perceber que para problemas de *routing* nos quais possa existir otimização na função objetivo (50, 75 e 100 clientes) e para um mesmo intervalo de tempo, o aumento do número de clientes, faz com que o valor no qual a função objetivo é otimizada diminua significativamente. Isto leva-nos a concluir que o aumento do número de clientes, torna os problemas cada vez mais complexos, levando então à necessidade da existência de cada vez mais tempo para efetuar a procura de soluções de forma a os conseguir melhor otimizar.

Para além disto, foram ainda analisados os resultados gerais obtidos para os diferentes números de clientes, de forma a compreender que impacto é que os mesmos têm no valor da função objetivo. A Figura 30 permite compreender a relação que o número de clientes tem no valor médio da função objetivo. Note-se que os valores da função objetivo nesta figura apresentados, são as médias gerais de todos os testes efetuados para todos algoritmos e para os tempos de procura desde 5 até 60 segundos.

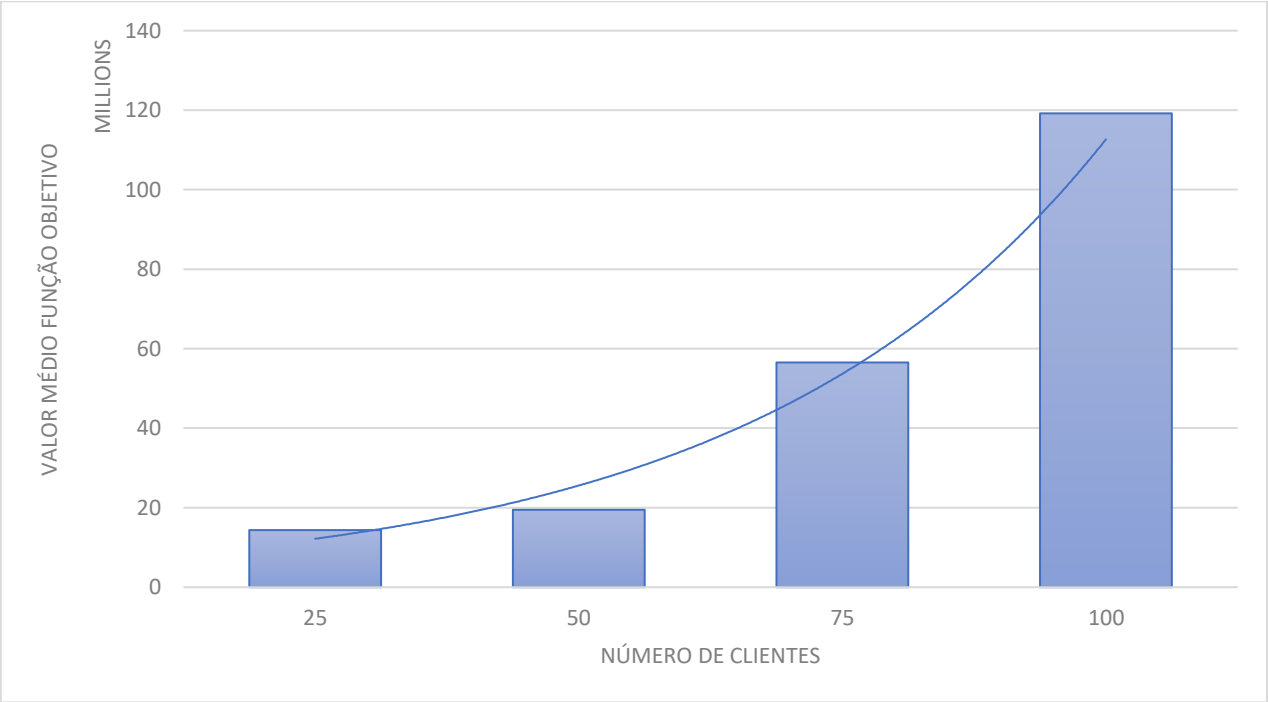

Figura 30 - Valor médio geral da função objetivo em função do número de clientes para todos os algoritmos considerados

A partir da Figura 30, é possível compreender que para os testes efetuados, o valor médio da função objetivo, aumenta em função do número de clientes.

A seguinte tabela permite compreender o crescimento do valor médio da função objetivo de 50, 75 e 100 clientes em relação ao valor médio da função objetivo de 25 clientes:

Número de clientes	Crescimento (%) em relação ao valor médio geral da função objetivo para 25 clientes
50	+36%
75	+294%
100	+730%

Tabela 13 - Crescimento do valor médio geral da função objetivo tendo em conta os diferentes números de clientes

Na Tabela 13, é possível perceber que se considerarmos um número de clientes de 50, o valor da função objetivo aumenta em média, por cerca de 36% em relação ao valor da mesma para 25 clientes. Se considerarmos o crescimento entre 25 e 75 clientes, este aumenta significativamente (294%) em relação ao crescimento observado anteriormente (para 50 clientes). Por último, considerando o crescimento observado entre os valores médios da função objetivo de 25 e 100 clientes, este crescimento aumenta cerca de 730%, sendo este ainda maior que o observado entre 25 e 75 clientes. Assim, estes resultados permitem de certa forma compreender que o valor médio da função objetivo cresce cada vez mais com o aumento do número de clientes, assemelhando-se assim a uma distribuição exponencial e tornando então o problema de *routing* cada vez mais complexo. Este aumento na complexidade do problema, faz com que de certa forma, exista também cada vez maior necessidade de otimização do problema, uma vez que o espaço de procura de soluções será muito maior.

5.2 Testes em simulação

Foram ainda efetuados testes usando simulação, de forma averiguar como é que o simulador se comporta juntamente com o *Solver* para uma perspetiva dinâmica do *DARP* (*Dial-a-Ride-Problem*) formulado, utilizando novamente a rede de transporte público do Funchal. Enquanto que nos testes efetuados anteriormente (situação estática) era necessário apenas obter uma única solução inicial, que continha o conjunto de percursos a ser efetuados de forma a servir todos os clientes conhecidos inicialmente. A situação dinâmica, distingue-se pelo facto de ser necessário construir os percursos dos diferentes veículos continuamente com o surgimento de novos pedidos de serviço, ou seja, existindo assim uma geração continua de soluções com a receção de novos pedidos. Os testes foram também efetuados utilizando um computador equipado com um processador Intel core i5 4210U e 4GB de memória RAM.

As configurações do *Solver* utilizadas para estes testes foram as seguintes:

- **Algoritmo de encontro primeira solução** – similarmente aos testes efetuados para os algoritmos de procura, foi também utilizado o algoritmo *Parallel Cheapest Insertion* como algoritmo de primeira solução;
- **Algoritmo de procura de soluções** – foi sempre utilizado o algoritmo *Tabu Search*, pelo facto de este mostrar-se em geral, na Tabela 11 - Tabela resumo com os algoritmos que demonstram em média os melhores e piores resultados, como o que obtém em média os valores médios mínimos da função objetivo. No entanto, note-se que poderia ter sido usado qualquer um dos outros algoritmos considerados, uma vez

que foi demonstrado que o desempenho dos mesmos é relativamente semelhante, existindo diferenças relativamente mínimas, para a maioria dos testes efetuados;

- **Tempo de procura de soluções –** o tempo de procura de soluções foi adaptado conforme o número de clientes existentes para o problema de *routing* a ser tratado no momento. O tempo de procura definido foi baseado no momento de estabilização considerado para os diferentes números de clientes com base nos resultados obtidos anteriormente para a situação estática (ver Figura 26, Figura 27, Figura 28). Quando este número fosse inferior a 25 clientes, foi utilizado um tempo de procura de 5 segundos, quando fosse superior a 25 e inferior a 50 foi utilizado um tempo de procura de 30 segundos, e por fim, quando fosse superior a 50 clientes foi utilizado um tempo de procura de 50 segundos;
- **Tempo máximo de viagem relativo dos clientes** – o tempo máximo de viagem relativo de cada um dos clientes considerados foi fixado em 30 minutos, sendo este o mesmo utilizado nos testes dos algoritmos efetuados anteriormente;
- **"Relaxamento" dos intervalos de tempo** - foi também definido o mesmo tempo máximo de 30 minutos de atraso aceitável em relação ao tempo de "entrega" (na paragem destino), originalmente desejado pelo cliente, de forma a "relaxar" certos problemas tornando-os possíveis de resolver;

Relativamente às configurações de simulação utilizadas, estas incluíam sucintamente:

- **Tempo de simulação** – tempo fixo de 4 horas de simulação. No entanto, notemos que a simulação decorria até que todos os clientes aceites para os diferentes serviços fossem servidos;
- **Número de veículos** – foi considerado um número fixo de 5 veículos;
- **Número de pedidos dinâmicos que surgem por hora de simulação (dinâmicos)** – foram efetuadas várias simulações, sendo variados os números de pedidos que surgem por hora. Foram efetuados testes para 20, 40 e 60 clientes dinâmicos por hora;
- **Número de pedidos conhecidos inicialmente (estáticos)** – não eram conhecidos nenhuns pedidos de serviço inicialmente, de forma a simular uma perspetiva puramente dinâmica. Esta abordagem foi tomada de forma compreender como é que o *Solver* irá se comportar juntamente com o Simulador para uma situação puramente dinâmica, na qual os pedidos só são conhecidos no momento do seu surgimento e não existindo assim nenhum conjunto de pedidos conhecidos inicialmente;
- **Capacidade dos veículos** – foi considerada uma frota homogénea, na qual cada veículo tinha uma capacidade máxima de 20 clientes;
- **Velocidade de deslocamento dos veículos** – foi considerado também que todos os veículos de deslocam à mesma velocidade média de deslocamento de 40 km/h;
- **Paragem de começo e termino** – todos os veículos começavam e terminavam o seu percurso na mesma paragem (*Depot*).

Cada pedido de serviço dinâmico foi gerado aleatoriamente, seguindo a mesma abordagem referida no Inicio da simulação (em 4.1.3), na qual as paragens de "entrega" e "recolha" eram escolhidas aleatoriamente, dentro da rede de transporte considerada, e sendo também os tempos de "entrega" e "recolha" desejados pelo mesmo escolhidos aleatoriamente.

Relativamente ao surgimento dos pedidos dinâmicos, foi seguida a abordagem referida no tratamento dos eventos de pedidos dinâmicos (subcapítulo 4.1.2.5), na qual sempre que surgia um novo cliente, era passado ao *Solver* um modelo de dados que refletia o contexto atual da simulação, isto é, um modelo dados que englobava todos os clientes que estão ou que serão servidos atualmente, o novo cliente que acaba de surgir, como também todos os veículos a serem utilizados na simulação. Sendo o *Solver* depois, responsável por construir uma nova solução (conjunto de percursos) de forma a servir todos esses clientes contidos no modelo de dados que reflete esse mesmo contexto (atual) da simulação.

Para cada um dos números de clientes dinâmicos por hora distintos considerados (20, 40 e 60 clientes), foram efetuadas 10 simulações distintas e sendo armazenadas informações importantes relativas às mesmas.

A Tabela 14 apresenta os resultados médios obtidos para as 10 simulações distintas efetuados para cada um dos diferentes números de pedidos dinâmicos por hora considerados.

	Número de pedidos dinâmicos (procura) por hora		
Valor	**20**	**40**	**60**
Número (média) total de clientes	80	160	240
Número (média) total de clientes servidos	68	91.9	100.6
Número total de testes (simulações)	10	10	10
Número de clientes do problema com maior número de clientes encontrado	19	29	34
Métricas relacionadas com a Qualidade do serviço			
% de clientes servidos	85.0%	57.4%	41.9%
% de clientes "entregues" sem atraso	70.7%	53.6%	51.2%
Tempo médio de espera dos clientes	8.5	10.3	11.3
Tempo médio de viagem dos clientes	14.5	15.9	15.2
Métrica relacionadas com custos para o operador de transporte			
Número (média) de veículos utilizados	5	5	5
Número médio de clientes servidos por veículo	13.6	18.4	20.1
Distância média percorrida por cada veículo	149.3	165.6	172.4
Distância média percorrida por cliente servido	11.0	9.0	8.6

Tabela 14 - Resultados dos testes efetuados em simulação para um problema de routing dinâmico para 20, 40 e 60 pedidos dinâmicos por hora

Como é possível ver na Tabela 14, através do valor do número de clientes do problema de *routing* maior resolvido, para os testes efetuados, o número de clientes para um dado problema que reflete o contexto atual da simulação, nunca ultrapassou os 50 clientes, mesmo quando eram considerados 60 pedidos dinâmicos por hora, assim sendo o tempo de procura imposto ao *Solver* durante a simulação nunca excedeu os 30 segundos.

5.2.1 Qualidade do serviço

Ao olharmos para a Tabela 14 - Resultados dos testes efetuados em simulação para um problema de routing dinâmico para 20, 40 e 60 pedidos dinâmicos por hora

Como é possível ver na Tabela 14, através do valor do número de clientes do problema de *routing* maior resolvido, para os testes efetuados, o número de clientes para um dado problema que reflete o contexto atual da simulação, nunca ultrapassou os 50 clientes, mesmo quando eram considerados 60 pedidos dinâmicos por hora, assim sendo o tempo de procura imposto ao *Solver* durante a simulação nunca excedeu os 30 segundos.

, é possível compreender que ao mantermos sempre o mesmo número fixo de veículos, de forma geral, a qualidade do serviço em média deteriora-se com o aumento do número de pedidos dinâmicos por hora.

Nomeadamente, a percentagem de clientes servidos diminui com este aumento, sendo esta cerca de 85% para uma procura de 20 clientes por hora, diminuindo depois para 57% para 40 clientes por hora, e, por fim, diminuindo depois para 42% para uma procura de 60 clientes por hora. No entanto, é importante referir que embora a percentagem de clientes servidos diminua significativamente, o número total de clientes servidos aumenta, indicando então que provavelmente o operador de transporte acabará por ter maiores lucros, ou pelo menos, custos inferiores por cliente transportado.

Relativamente à percentagem de clientes "entregues" sem atraso, ou seja, clientes cujo tempo desejado para estar na paragem de destino foi respeitado, é também observável que a mesma sofre uma diminuição com o aumento da procura. Indicando assim que se torna cada vez mais difícil de respeitar os tempos de "entrega" dos clientes com o aumento da procura.

Relativamente, ao tempo médio de viagem dos clientes, foi observável um aumento de quase 1 minuto e 30 segundos (1.4 minutos) neste tempo com o aumento da procura de 20 para 40 clientes, e depois uma diminuição do mesmo de 40 para 60 clientes em cerca de quase 45 segundos (0.7 minutos). No entanto, a diferença entre estes valores, para as diferentes procuras consideradas, não é muito significativa, não sendo possível então identificar grande relação entre a procura e os tempos médios de viagem dos clientes. É ainda importante referir que estes tempos serão sempre influenciados pela localização de cada uma das paragens de "entrega" e "recolha", como também pelos tempos para chegar e partir das mesmas, impostos pelo pedido de cliente. Uma vez que as paragens e tempos de "entrega" e "recolha" de cada cliente são geradas aleatoriamente, o tempo de viagem será então sempre influenciada por ambas estas informações geográficas e temporais.

Relativamente ao tempo médio de espera dos clientes, é também novamente observado que este deteriora-se (aumenta) com o aumento da procura. Mais concretamente, o tempo médio de espera dos clientes é cerca de 8 minutos para uma procura de 20 clientes por hora, aumentando depois para cerca de 10 minutos para uma procura de 40 clientes, e por fim para 12 minutos para uma procura de 60 clientes. Com base nisto, para uma frota de veículos fixa os clientes irão ter que esperar mais, em média, diminuindo então a qualidade do serviço com o aumento da procura, pelo facto de ser necessário servir mais clientes.

Com isto, de modo geral foi observado que a qualidade do serviço se deteriora com o aumento da procura. No entanto, é importante notar que existem algumas estratégias que

podem ser tomadas de forma a melhorar a qualidade do serviço refletida por uma certa métrica. Nomeadamente, para o caso de querermos aumentar o número de clientes servidos, uma estratégia poderia ser simplesmente aumentar o número de veículos disponíveis, permitindo então servir mais clientes, pelo simples facto de existir mais recursos para o fazer. No entanto é notório que embora esta estratégia tenha vantagens a nível da qualidade de serviço, esta também iria impor maiores custos operacionais.

Uma outra abordagem poderia ser o "relaxamento" de alguma das restrições relacionadas com os tempos que devem de ser cumpridos, como por exemplo aumentar o tempo máximo de viagem do cliente, ou aumentar os tempos de "entrega" desejados pelo cliente. Esta abordagem por um lado iria melhorar a qualidade do serviço ao permitir servir mais clientes, mas por outro lado iria deteriorar os tempos de espera como também os tempos de viagem dos mesmos, deteriorando também a qualidade do serviço, existindo então a necessidade de encontrar algum tipo de balanço entre ambas estas métricas de qualidade de serviço.

Para diminuir os tempos médios de viagem de cada cliente, bastava diminuir a restrição imposta ao *Solver* do tempo máximo relativo de viagem considerado para cada cliente. Contudo, a diminuição desta métrica iria, de certa forma, levar a que menos clientes fossem servidos, atendendo a existirem intervalos de tempo mais restritos a serem cumpridos. Assim sendo, ao ser utilizada esta abordagem iria existir um certo ganho a nível de qualidade de serviço, no que toca ao tempo de viagem dos clientes, mas também existiria uma perda na qualidade do serviço, pelo facto de serem servidos menos clientes. Isto realça novamente, a necessidade de encontrar um certo balanço entre as diferentes métricas consideradas.

5.2.2 Custos para o operador de transporte

No que toca ao custo do operador é notório que para qualquer uma das procuras consideradas, o número de veículos utilizado foi o mesmo. Indicando assim que o *Solver*, geralmente utiliza os veículos todos disponíveis de forma a tentar cumprir todas as restrições impostas ao problema.

Embora a percentagem de clientes servidos sofra com o aumento da procura, é possível observar que os veículos são mais bem aproveitados, aumentando assim a ocupação (média de clientes servidos por veículo) com o aumento dos pedidos de serviço por hora. Assim, para uma frota fixa de 5 veículos, será mais benéfico em termos de custo operacional tentar servir 60 clientes por hora do que 20, pois será menos custoso servir cada um dos clientes tendo em conta o número de veículos utilizados.

Relativamente à distância total percorrida por cada um dos veículos é possível compreender que esta aumenta com o aumento da procura. Isto já era expectável, uma vez, que se o número total de clientes servidos aumenta, em média a distância total percorrida por cada um dos veículos acabará por aumentar por simplesmente, existirem mais paragens a ser visitadas.

Por fim ao observarmos a distância total percorrida por cada cliente servido, é notório que existe uma diminuição nos custos operacionais em relação ao número de clientes servidos, indicando novamente que provavelmente a nível de custos operacionais seja mais benéfico considerar uma procura mais elevada de 40 ou 60 clientes.

Ao contrário do que acontecia para a qualidade do serviço, de modo geral, o aumento da procura, traz benefícios a nível de custos operacionais, permitindo assim de certa forma melhor aproveitar os recursos existentes. Todavia, é importante referir que podem ser exploradas algumas possíveis soluções de forma a tentar melhor reduzir os custos operacionais no *Solver*, de certa forma tentando assim melhor otimizar a utilização dos recursos disponíveis.

Uma delas podia ser o facto de testar a resolução dos problemas de *routing* com um número mínimo de veículos (em relação ao número de veículos disponíveis). Imaginemos a seguinte situação na qual um novo pedido dinâmico acaba de surgir, e sabemos que existe um dado número de veículos disponíveis, em vez de passar o contexto atual da simulação tendo em conta todos esses veículos, era efetuado por tentativas com um número veículos mínimo, caso fosse possível resolver, com esse número reduzido, eram utilizados apenas esses veículos, caso contrário era incrementado o número de veículos considerados e novamente efetuada a resolução até que chegasse ao número máximo de veículos disponíveis para utilização ou até que fosse encontrado o número mínimo de veículos necessários para resolução desse problema de *routing*.

Uma outra abordagem que poderia ser também tomada para melhor aproveitamento dos veículos utilizados, seria a imposição de um custo relativamente elevado no *Solver*, à utilização de cada um dos diferentes veículos. Este custo começaria relativamente baixo para um único veículo, mas com a adição de mais veículos iria aumentado rapidamente. Por exemplo, era adicionado um custo de 1000 à função objetivo se fosse utilizado o primeiro veículo, para o segundo era adicionado um custo de 10000 (10 vezes maior em relação ao primeiro), para o terceiro era adicionado um custo de 100000 (10 vezes maior em relação ao segundo), aumentado cada vez mais com o aumento da quantidade veículos considerados. Isto faria com que o *Solver* tentasse de certa forma reduzir os custos operacionais associados à solução, uma vez que o objetivo do mesmo é minimizar os custos totais da função objetivo, permitindo assim então, reduzir ou pelo menos restringir a utilização de veículos adicionais.

Uma outra abordagem que poderia ser tomada para reduzir os custos operacionais podia ser a adição de restrições ao *Solver* de forma a limitar o número de paragens visitadas por cada veículo, ou a limitação da distância total percorrida por cada veículo considerado.

É notório que a utilização de qualquer uma das abordagens acima referidas, iriam permitir de certa forma reduzir os custos operacionais, ao restringir os recursos utilizados. No entanto, estas teriam também um impacto negativo na qualidade do serviço, pelo facto de estas restringirem a forma como o mesmo é prestado, o que acabará por resultar numa redução no número total de clientes servidos. Assim sendo, será necessário encontrar um balanço entre as métricas que refletem qualidade de serviço, como também as que refletem custos operacionais, de forma a ser possível obter ganhos para ambas as partes interessadas (cliente e fornecedor de transporte).

6 Conclusão e Trabalho futuro

Com este trabalho foi possível conceber um modelo para um problema de *routing* de transporte flexível geralmente chamado de *Dial-a-Ride-Problem* (*DARP*) na literatura corrente. Mais concretamente, foi especificado o raciocínio utilizado para modelação e implementação de tal modelo, tendo sido este implementado com o auxílio da biblioteca *Google Operations Research Tools* (*Google OR-Tools*). Para mais, foi ainda efetuada a modelação e construção de um modelo de simulação de eventos discretos de forma a permitir testar e avaliar o referido modelo de *routing* para as duas abordagens (estática e dinâmica) existentes do *DARP*.

Foram ainda efetuados testes para um conjunto de situações estáticas distintas do problema de transporte modelado e utilizando a rede de transporte público do Funchal, de forma avaliar um conjunto de algoritmos de procura de soluções presentes na já referida biblioteca. Estes testes foram efetuados para instâncias distintas, mas semelhantes às encontradas na simulação, onde são considerados diferentes números de clientes (desde 25 até 100 clientes). Foi ainda variado o tempo de procura de soluções utilizado em cada um dos algoritmos. Através destes testes foi possível compreender que, de modo geral, os algoritmos testados acabam por ter um desempenho relativamente semelhante para os diferentes números de clientes considerados. Todos estes chegam a um dado momento no tempo, no qual estabilizam, não havendo melhorias nas soluções encontradas (para o intervalo de tempo de procura de soluções considerado). Para além disto, foi ainda possível perceber que para os testes efetuados o aumento do número de clientes torna os problemas cada vez mais complexos, o que leva a que os algoritmos de procura de soluções necessitem de cada vez mais tempo para efetuar a otimização.

Além do mais, foram ainda efetuados testes em simulação para a situação dinâmica do problema de *routing* modelado para um número de veículos fixos. O objetivo foi compreender o efeito que o aumento da procura (número de pedidos dinâmicos por hora) tinha em diversas métricas, que refletem os custos para o operador de transporte, como também a qualidade do serviço prestado aos clientes. Foi observado que a qualidade do serviço deteriora-se com o aumento da procura, sendo cada vez mais difícil garantir métricas de qualidade de serviço. Embora exista uma perda na qualidade do serviço com o aumento da procura, foi ainda observado que este aumento permite de certa forma diminuir os custos operacionais, se tivermos em consideração o custo benefício, concluindo-se que a disponibilidade dos veículos utilizados será melhor aproveitada se existir maior procura (naturalmente). Para além disto, foram ainda referidas algumas abordagens que poderiam ser tomadas de forma a tentar melhorar algumas das métricas relacionadas com a qualidade do serviço ou custos operacionais. Ademais, foi ainda demonstrado que o melhoramento em alguma métrica que reflete algum aspeto de qualidade relacionado com o cliente ou operador, por vezes deteriora o aspeto oposto, existindo então a necessidade de encontrar um balanço entre ambos estes aspetos de qualidade, isto é, os custos operacionais e a qualidade de serviço.

Relativamente a trabalho futuro, um aspeto que achamos interessante explorar, seria efetuar um conjunto de simulações baseadas em dados reais, nas quais poderiam ser experimentadas diversas configurações do problema. Tanto a procura existente em cada momento, como também as paragens referentes ao pedido de cada cliente e seus respetivos intervalos de tempo, seriam então de alguma forma baseadas nessas informações reais. Este

aspeto seria uma mais valia do ponto de vista de avaliação do problema de transporte flexível modelado, uma vez que seria possível compreender como é que o mesmo se comporta numa situação muito mais aproximada da real. O conhecimento adquirido através destas simulações, irá permitir ajustar o modelo do problema de forma a ajustá-lo à realidade. Em particular, poderia existir a necessidade de "relaxar", remover ou adicionar algum tipo de restrição ou característica imposta ao mesmo, de forma a permitir melhorar algum aspeto relacionado com os custos operacionais, ou qualidade do serviço.

Será, também, interessante efetuar testes para os algoritmos de primeira solução contidos na biblioteca utilizada para formular e resolver o problema de *routing*. Como já foi referido anteriormente, os algoritmos de primeira solução são responsáveis por gerar uma solução inicial para um dado problema. Assim, a analise destes testes seria interessante de forma a tentar compreender se existe algum algoritmo de primeira solução que, em geral, gere uma solução inicial com custo relativamente inferior e mais aproximada da "ótima". Caso isto seja verificável, possibilitará reduzir os tempos de procura utilizados nos algoritmos de procura de soluções, aumentando assim o desempenho, uma vez que seria necessário menos tempo para efetuar a otimização da solução, uma vez que a solução inicial se encontraria mais próxima da "ótima".

7 Referências

[1] J. Cordeau, “A Branch-and-Cut Algorithm for the Dial-a-Ride Problem,” *Oper. Res.*, vol. 54, no. 3, 2006, doi: 10.1287/opre.1060.0283.

[2] R. M. Jorgensen, J. Larsen, and K. B. Bergvinsdottir, “Solving the Dial-a-Ride problem using genetic algorithms,” *J. Oper. Res. Soc.*, vol. 58, no. 10, pp. 1321–1331, 2007, doi: 10.1057/palgrave.jors.2602287.

[3] R. W. Calvo and A. Colorni, “An effective and fast heuristic for the Dial-a-Ride problem,” *4or*, vol. 1, pp. 61–73, 2007, doi: 10.1007/s10288-006-0018-0.

[4] L. Bodin and T. R. Sexton, “The multi-vehicle subscriber dial-a-ride problem,” 1986.

[5] Z. Xiang, C. Chu, and H. Chen, “A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints,” *Eur. J. Oper. Res.*, vol. 174, no. 2, pp. 1117–1139, 2006, doi: 10.1016/j.ejor.2004.09.060.

[6] K. I. Wong and M. G. H. Bell, “Solution of the Dial-a-Ride Problem with multi-dimensional capacity constraints,” *Int. Trans. Oper. Res.*, vol. 13, no. 3, pp. 195–208, 2006, doi: 10.1111/j.1475-3995.2006.00544.x.

[7] M. E. Bruni, F. Guerriero, and P. Beraldi, “Designing robust routes for demand-responsive transport systems,” *Transp. Res. PART E*, vol. 70, pp. 1–16, 2014, doi: 10.1016/j.tre.2014.06.002.

[8] L. Coslovich, “A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem,” *Eur. J. Oper. Res.*, vol. 175, no. 3, pp. 1605–1615, 2006, doi: 10.1016/j.ejor.2005.02.038.

[9] I. Zidi, K. Mesghouni, K. Zidi, and K. Ghedira, “A new Approach based on the Multi-Objective Simulated Annealing to solving the Dynamic Dial a Ride Problem,” *2011 4th Int. Conf. Logist.*, pp. 157–163, 2011, doi: 10.1109/LOGISTIQUA.2011.5939419.

[10] L. Khelifi, I. Zidi, K. Zidi, and K. Ghedira, “A Hybrid Approach Based on Multi-Objective Simulated Annealing and Tabu Search to solve the Dynamic Dial a Ride Problem,” *2013 Int. Conf. Adv. Logist. Transp.*, vol. 2, no. 1, pp. 227–232, 2013, doi: 10.1109/ICAdLT.2013.6568464.

[11] D. Teodorovic and G. Radivojevic, “A fuzzy logic approach to dynamic Dial-A-Ride problem,” *Fuzzy Sets Syst.*, vol. 116, no. 1, pp. 23–33, 2000, doi: 10.1016/S0165-0114(99)00035-4.

[12] M. Schilde, K. F. Doerner, and R. F. Hartl, “Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports,” *Comput. Oper. Res.*, vol. 38, no. 12, pp. 1719–1730, 2011, doi: 10.1016/j.cor.2011.02.006.

[13] Z. Xiang, C. Chu, and H. Chen, “The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments,” *Eur. J. Oper. Res.*, vol. 185, no. 2, pp. 534–551, 2008, doi: 10.1016/j.ejor.2007.01.007.

[14] G. Laporte, “A tabu search heuristic for the static multi-vehicle dial-a-ride problem,” *Transp. Res. Part B Methodol.*, vol. 37, no. 6, pp. 579–594, 2003, doi: 10.1016/S0191-2615(02)00045-0.

[15] S. Ropke, C. Ht, and G. Laporte, “Models and branch-and-cut algorithms for pickup and delivery problems with time windows,” *Networks An Int. J.*, vol. 49, no. 4, pp. 258–272, 2007, doi: 10.1002/net.

[16] M. Diana and M. M. Dessouky, “A new regret insertion heuristic for solving large-scale

dial-a-ride problems with time windows," *Transp. Res. Part B Methodol.*, vol. 38, no. 6, pp. 539–557, 2004, doi: 10.1016/j.trb.2003.07.001.

[17] J.-J. Jaw, A. R. Odoni, H. N.Psaraftis, and N. H.M. Wilson, "A heuristic algorithm for the multivehicle advance request dial-a-ride problem with time windows," *Transp. Res. Part B Methodol.*, vol. 20, no. 3, pp. 243–257, 1986, doi: 10.1016/0191-2615(86)90020-2.

[18] O. B.G. Madsen, H. F. Ravn, and J. Moberg Rygaard, "A heuristic algorithm for a dial-a-ride problem with time windows , multiple capacities , and multiple objectives," *Ann. Oper. Res.*, vol. 60, pp. 193–208, 1995, doi: 10.1007/BF02031946.

[19] A. Attanasio, J.-F. Cordeau, G. Ghiani, and G. Laporte, "Parallel Tabu Search Heuristics for the Dynamic Multi-Vehicle Dial-a-Ride Problem," *Parallel Comput.*, vol. 30, no. 3, pp. 377–387, 2004, doi: 10.1016/j.parco.2003.12.001.

[20] A. Lois and A. Ziliaskopoulos, "Algorithm for dynamic dial a ride problem and its metrics," *Transp. Res. Procedia*, vol. 24, no. 2016, pp. 377–384, 2017, doi: 10.1016/j.trpro.2017.05.097.

[21] J. C. G. Laporte, "The dial-a-ride problem : models and algorithms," *Ann. Oper. Res.*, vol. 153, pp. 29–46, 2007, doi: 10.1007/s10479-007-0170-8.

[22] F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*. 2015.

[23] "OR-Tools | Google Developers." [Online]. Available: https://developers.google.com/optimization. [Accessed: 21-Nov-2019].

[24] "Requisitos funcionais e requisitos não funcionais, o que são?" [Online]. Available: https://codificar.com.br/blog/requisitos-funcionais-nao-funcionais/. [Accessed: 07-Oct-2019].

[25] "RoutingDimension | OR-Tools | Google Developers." [Online]. Available: https://developers.google.com/optimization/reference/constraint_solver/routing/RoutingDimension. [Accessed: 15-Nov-2019].

[26] "or-tools/routing.cc at 554cbccaa95b4c11eced13f145de1bf468e1f919 · google/or-tools." [Online]. Available: https://github.com/google/or-tools/blob/554cbccaa95b4c11eced13f145de1bf468e1f919/ortools/constraint_solver/routing.cc#L4251-L4261. [Accessed: 19-Nov-2019].

8 Anexos

8.1 Anexo 1 - Tabela com os dados do valor médio da função objetivo para os testes efetuados para 25, 50, 75 e 100 clientes

Valor médio da função objetivo (cada algoritmo e tempo de procura foi testado 10 vezes para cada número de clientes considerado) *					
	Algoritmo				
Tempo de procura (segundos)	**Generic Tabu Search**	**Greedy Descent**	**Guided Local Search**	**Simulated Annealing**	**Tabu Search**
Número de clientes = 25					
5	14364 322	14364 350	14364 310	14364 350	14364 156
10	14364 322	14364 350	14364 306	14364 350	14364 156
15	14364 322	14364 350	14364 201	14364 350	14364 106
20	14364 322	14364 350	14364 085	14364 350	14364 075
25	14364 322	14364 350	14363 107	14364 350	14364 074
30	14364 322	14364 350	14363 118	14364 350	14364 074
35	14364 322	14364 350	14363 102	14364 350	14364 074
40	14364 322	14364 350	14363 072	14364 350	14364 036
45	14364 322	14364 350	14363 069	14364 350	14364 036
50	14364 322	14364 350	14363 069	14364 350	14364 036
55	14364 322	14364 350	14363 058	14364 350	14364 024
60	14364 322	14364 350	14363 058	14364 350	14364 024
Média	**14364 322**	**14364 350**	**14363 463**	**14364 350**	**14364 073**
Número de clientes = 50					
5	52029 905	51507 453	53594 416	55968 667	51748 179
10	39084 308	33633 902	36983 089	34789 189	35066 056

15	24340351	28521568	28539178	26970858	23196426
20	18022502	21312684	17731526	21983667	17335410
25	15770890	13858080	15089204	14633026	12718053
30	12719150	12194420	12066075	12719154	12065617
35	12066599	12066428	12065918	12066428	12065088
40	12066400	12066428	12065890	12066428	12065010
45	12066400	12066428	12065887	12066428	12065008
50	12066400	12066428	12065887	12066428	12065008
55	12066400	12066428	12065877	12066428	12065008
60	12066400	12066428	12064796	12066428	12065008
Média	**19530475**	**19452223**	**19699812**	**19955261**	**18709989**
Número de clientes = 75					
5	98343256	99106332	98448255	98247284	98241322
10	90552113	91007161	90365176	90806136	90457184
15	84595592	84587559	85342154	84311613	84857570
20	73514867	73182080	75096532	72450122	71946984
25	65286360	62827695	63201644	63996168	61023359
30	49713141	49933342	49414242	49700417	49365153
35	44988154	45268210	44748452	44748616	44640985
40	39146848	40009681	39333933	38879094	42086541
45	35543989	35266750	35446527	35857787	35059186

50	33263 080	33696 302	33364 891	33284 790	33833 102
55	32869 603	32424 638	32504 497	32358 429	32971 674
60	32006 776	32007 115	32147 310	32217 652	32006 838
Média	**56651 982**	**56609 739**	**56617 801**	**56404 842**	**56374 158**
Número de clientes = 100					
5	14178 5127	14217 2201	14231 1216	14188 2181	14202 3173
10	13393 8277	13393 8282	13412 8270	13479 8215	13393 7292
15	12903 0367	12924 4262	12904 4364	12940 8260	12927 0273
20	12520 9453	12511 9472	12450 7530	12455 4575	12456 9544
25	12168 5629	12171 6596	12207 9564	12241 5563	12169 9631
30	11938 2574	11974 0574	11960 8527	11923 5569	11937 0603
35	11393 2786	11394 9669	11798 1504	11803 7497	11538 8708
40	11265 7557	11292 8575	11318 9748	11252 6839	11257 2812
45	11070 3582	11054 9615	11098 9514	11067 0581	11045 7550
50	10868 2386	10876 5383	10923 9489	10873 9394	10877 3464
55	10774 8374	10698 4636	10783 1349	10816 1350	10788 5330
60	10448 5506	10382 0816	10152 6009	10146 4015	10348 4956
65	96092 416	95425 361	96946 442	96248 295	96058 627
70	93102 302	93106 874	95399 598	93051 444	92971 566
75	91530 625	91186 226	91039 164	91510 061	90078 121
80	87735 292	87677 686	87703 350	87696 457	88225 778

85	86556342	86611313	86461528	87516151	87819212
90	82753304	85546117	85629036	82803191	83175193
Média	**10927 8439**	**10936 0203**	**10975 6456**	**10948 4424**	**10932 0102**

*Note-se que o valor médio da função objetivo apresentado para cada tempo de procura, é a média de 10 testes distintos efetuados para o algoritmo e número de clientes em questão.