Nuno Manuel Couto Barreira

Licenciado em Ciências de Engenharia e de Computadores

Sistema Inteligente para Otimização de Rotas

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientador: Ricardo Luís Rosa Jardim Gonçalves, Professor Associado com Agregação, Departamento de Engenharia Eletrotécnica Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Coorientador: Carlos Manuel de Melo Agostinho, Investigador, Centre of Technology and Systems - UNINOVA

Júri:

Presidente: Doutor João Francisco Alves Martins

Arguente: Doutora Anabela Monteiro Gonçalves Pronto

Vogal: Doutor Carlos Manuel de Melo Agostinho

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro de 2016

Sistema Inteligente para Otimização de Rotas

Aos meus pais e namorada

Agradecimentos

A finalização deste trabalho é o culminar de anos de estudo e trabalho, na qual o nosso esforço é recompensado com o término da vida académica e uma nova fase se inicia. Claro que não era possível atingir este dia importante sem o apoio e ajuda das pessoas que nos rodeiam.

Em primeiro lugar agradecer ao orientador Doutor Ricardo Gonçalves pela oportunidade de realizar esta dissertação e ao coorientador Doutor Carlos Agostinho pelos conselhos e ajuda que ofereceu para atingir os objetivos propostos. Ainda um especial agradecimento à Raquel Melo e ao Paulo Figueiras que se mostraram sempre disponíveis a ajudar e por terem sido persistentes.

Agradecer ao projeto C2NET por apostar na melhoria e desenvolvimento de novas tecnologias, tanto na produção como no aumento de produtividade das empresas.

Um agradecimento à instituição onde passei estes últimos anos, a Faculdade de Ciências e Tecnologia, que proporcionou-me grandes momentos e onde fiz grandes amizades, nomeadamente o Diogo Pinto, Pedro Bueno, Hugo Pereira, Bruno Caixinha, Pedro Oliveira, David Aleixo, Nuno Pinheiro e novamente à Raquel Melo. Agradecer às minhas queridas amigas Ana Faria, Ana Cruz, Inês Pinheiro, Joana Urbano, Telma Barroso, Mariana Borges e Ana Sofia.

Ainda referir a Equipa de Futebol da Faculdade que me abriu a um vasto mundo de oportunidades e na qual partilhei grandes alegrias com o Bruno Cunha, Rafael Coelho, Guilherme Rosa e Zé Soares.

Um especial agradecimento à minha namorada Liliana Marinho, pois tem sido o meu pilar, o meu grande apoio nestes últimos anos e pelo seu amor e carinho.

Finalmente e mais importante, as pessoas que foram responsáveis por eu conseguir viver todas estas alegrias, partilhar momentos únicos com os meus amigos e de ter uma vida académica inesquecível, os meus pais Manuela Barreira e Manuel Barreira, devo tudo a eles e a eles estarei sempre grato por tudo o que fizeram por mim.

Um agradecimento geral a todos os colegas e amigos que de um modo geral contribuíram para a realização deste trabalho.

Resumo

Hoje em dia, tem-se assistido a um grande aumento do transporte rodoviário, quer seja para realizarem viagens de longa distância ou simplesmente para circular dentro da cidade, o que cria um aumento de veículos na estrada. Tal facto deriva em situações problemáticas, principalmente relacionadas com o consumo de tempo, atrasos, a possibilidade de passar inúmeras horas presos no trânsito, e o aumento dos custos expectáveis das viagens.

Devido a essas desvantagens, este problema tem sido estudado por diversos investigadores e usado em diferentes aplicações no mercado que fazem uso de algoritmos de otimização de rotas, e também da recolha de dados em tempo real do tráfego rodoviário. A questão principal é que as soluções existentes são geralmente proprietárias e as informações não são acessíveis a qualquer um.

Assim, esta dissertação aborda o problema de otimização de rota com a possibilidade de recolha de dados em tempo real, com a particularidade de poder escolher pontos intermédios e ainda de poder escolher uma rota mais económica, em termos de combustível. Na solução apresentada é possível determinar qual a melhor rota, consoante as preferências do utilizador, recolhendo informação de eventos que possam originar algum tipo de trânsito. Será possível ainda adicionar passagens intermédias obrigatórias na rota inicial, redefinindo a rota inicialmente proposta.

Palavras-chave: otimização de rotas, recolha de dados, algoritmos de otimização de rotas.

Abstract

Nowadays, we experience an augmentation usage of road transports for travel long distances or simply move around in the city, which creates an increasing of vehicles in the road. There are problematic situations mainly related to the time consumption, delays, the possibility of spending countless hours stuck in the traffic and the increase of the expected travel costs.

Due to these drawbacks, this problem has been studied by several researches and been used in different applications on the market that make use of route optimization algorithms and collect traffic data in real time. The main issue is that existing solutions, in general, are propriety and the information within is not accessible for everyone.

Thus, this paper addresses the problem of route optimization with the possibility of real time data collection, with the particularity to choose intermediate locations and still be able to choose a route more economical in terms of fuel. In the presented solution there is the possibility to calculate the best travelling route, according to the user preferences, and by collecting events information that can eventually lead to a certain traffic problem. It is also still possible to add intermediate locations for a route.

Keywords: route optimization, data collection, route optimization algorithms.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

GPS – Sistema de Posicionamento Global

GRASP – Procedimento de Pesquisa Adaptativo Aleatório Ganancioso

API – Interface de Programação de Aplicações

RFID – Identificação por Radiofrequência

MVTDC – Recolha de Dados de Tráfego dos Veículos Multi-tipo

STA – Agentes de Tráfego Inteligente

RDS – Sistema de Dados de Rádio

ICEDB – Base de Dados Conectado Intermitente

VNSRTTI – Sistema de Navegação do Veículo com Informações de Tráfego em Tempo Real

MVC – Modelo – Visão – Controlador

TTCN – Notação Combinada de Árvore e Tabular

HTML5 – Linguagem de Marcação Hipertexto, versão 5

PME – Pequenas e Médias Empresas

Capítulo 1 - Introdução

Com a evolução das empresas modernas relacionadas com os transportes, tem-se assistido a um melhoramento dos serviços rodoviários prestados, tanto pela melhoria tecnológica como pelos esforços das empresas e pesquisadores no intuito de melhorar e desenvolver métodos que minimizem alguns fatores adversos provenientes do tráfego rodoviário (Kawamura, 2006). No entanto, nem sempre é possível acompanhar a evolução tecnológica. Assistisse a uma vasta gama de produtos ou melhorias que as empresas podem ter acesso mas pelo simples facto de acarretar em mais custos em renovação ou atualização de produtos e de pessoal, as empresas não conseguem suportar.

A evolução também pode ser benéfica para ajudar a tomar decisões no dia-a-dia. E numa empresa é bastante fulcral tomar as melhores decisões, quer seja a nível de investimentos, contratação de pessoal, aposta em melhorias em determinadas áreas, acordos com outras empresas, ou como ilustrado na figura 1.1, definir a melhor rota no caso de empresas que dependam de transportes para exercer as suas funções.

A otimização de rotas de transportes acarreta algumas dificuldades na sua resolução e têm muitas aplicações práticas. E ainda beneficiam as empresas economicamente e no seu impacto ambiental. De acordo com (Benslimane, 2014), é possível minimizar os custos de transportes das empresas na ordem dos 5% a 20%, com procedimentos computadorizados baseados em técnicas de otimização. Segundo (Polit, 2006), as transportadoras têm que manter a boa qualidade dos serviços. Atrasos e adiantamentos devem ser evitados, ou em último recurso minimizados pois são prejudiciais e provocam perda de reputação e fiabilidade no serviço prestado. Este facto associado ao elevado congestionamento nas cidades e em certos percursos, para além do consumo de combustíveis, faz com que haja cada vez mais estudos a ser desenvolvidos tanto a nível profissional como académico, para otimizar os processos de rotas de transportes com a finalidade de melhorar a sua gestão (SILVA, 2013). Em suma, não basta oferecer um serviço de qualidade se a entrega não for feita a tempo e sem problemas (Souza, 2013).

Se houver por parte das transportadoras e das empresas clientes um trabalho conjunto no âmbito de melhorar os serviços relacionados com os transportes, o esforço de cada entidade será reduzido. Por vezes as melhores rotas de transportes para as transportadoras, podem não ser as melhores para as empresas clientes. Estes fatores e a exaustiva procura de minimizar estes problemas, leva à necessidade das transportadoras evoluírem no sentido de se aproximarem mais das exigências dos clientes. O simples facto de haver comunicação entre dois veículos da mesma transportadora ou saber de antemão que determinado percurso está

congestionado pode reduzir o tempo de transporte e consequentemente reduzir o custo e até melhorar a qualidade do serviço (Ling-hui, 2010).

Figura 1.1 - Caminho mais curto/Caminho de melhor qualidade
("http://passosapsicologia.blogspot.pt/," 2015)

No entanto, hoje em dia, dada a diversidade de métodos ou modelos que permitem ajudar a definir uma rota para veículos, nem sempre sabe-se o que usar para melhor satisfazer as necessidades de toda a gente. As pessoas são bombardeadas com programas e aplicações que de uma maneira ou de outra tentam usar diariamente, mas que nem sempre alcançam as expectativas. Ou porque são demasiado complexos ou por estarem incompletos em alguns aspetos ou pelo simples facto de não ser o que se procura. Normalmente, procura-se algo que seja intuitivo e de fácil acesso. Que possa andar sempre connosco e disponível para uso frequente. Mas dada a complexidade do problema, (Malaquias, 2006) refere que esta exigência na procura por um método, modelo ou algoritmo deve apresentar algumas características importantes:

- Precisão, para ter uma noção de que uma solução é ótima ou que está perto do ótima.
- Velocidade, pois precisa-se de algo que seja rápido a ser executada.
- Simplicidade, onde a facilidade de compreensão é dos aspetos mais importantes.
- Flexibilidade, para que se possa adaptar à maioria das aplicações na vida real.

Todas estas características fazem sentido mas geralmente nem todas são possíveis de obter. O ideal seria ter o máximo dessas características juntas, e quando não for possível ter todas, pelo menos o máximo possível.

Neste documento o foco do trabalho está relacionado com o transporte de matérias-primas ou de mercadorias, como produto acabado. Neste caso, irá ser dada maior importância à

determinação em tempo real da melhor rota de transporte consoante as necessidades específicas do utilizador, visto ter bastante peso numa empresa que dependa de transportadoras para poder operar eficientemente.

Este trabalho propõe-se a achar uma resposta para a seguinte questão:

Como criar uma ferramenta ou método flexível que permita personalizar a otimização em tempo real do tipo de rota que um veículo pode tomar?

E formulando a seguinte hipótese:

Se for possível personalizar qual a rota que um veículo deve seguir a cada momento, com recolha de dados em tempo real e recurso a algoritmos de otimização, então será possível obter um resultado de qualidade do percurso a seguir.

1.1. Método de Pesquisa

O método cientifico adotado neste documento refere-se ao método clássico referido em (Camarinha-matos & Terminology, 2016). É um método iterativo onde os resultados podem não ser os esperados mas é prático voltar às etapas iniciais e fazer uma nova abordagem. Está dividido em sete etapas, na qual se inicia numa abordagem teórica e vai-se evoluindo para uma mais prática.

Figura 1.2 – Método Científico Clássico (Camarinha-matos & Terminology, 2016)

Cada etapa é definida e explicada de seguida:

- **Questão / Problema de Pesquisa**

Nesta etapa é definida a área de interesse da investigação e termina com uma questão de pesquisa que será a base de todo o estudo da dissertação. A questão de pesquisa tem que ser clara e direta, havendo a possibilidade de ser confirmada ou refutada, tal como é demonstrado neste capítulo.

- **Observação / Investigação**

Esta é a etapa onde se recolhe informações científicas existentes sobre o assunto a retratar. Pode ser realizado através da revisão da literatura ou projetos científicos e realçando as ideias de outros autores. O estado de arte é um estudo importante, pois a informação recolhida embora fiável, pode estar ultrapassada ou por outro lado, alguns documentos podem ser recentes e ter ideias inovadoras, mas com baixa fiabilidade. Neste caso, é feita a recolha de métodos e algoritmos e de plataformas de recolha de dados que podem beneficiar no problema em questão, como é exemplificado no capítulo 2.

- **Formulação da Hipótese**

A hipótese deve ser formulada de maneira a tornar o problema simples de entender, trazendo clareza, especificidade e explicar uma solução que responda às características do problema. Nesta dissertação é necessário que a hipótese se centre na customização de rotas de veículos em tempo real com recurso a algoritmos de otimização.

- **Experimentação**

Esta etapa funciona como o planeamento detalhado da fase experimental. Nesta dissertação esta etapa refere-se à criação de uma implementação tecnológica para personalizar rotas de veículos, como é explicado no capítulo 3.

- **Testar a Hipótese**

Inicialmente deve ser definido um conjunto de testes de acordo com as características do problema e da hipótese formulada. Assim, para cada teste deve ser recolhido os dados para posterior análise e validação das hipóteses. Esses testes são realizados no capítulo 4. Se possível os dados recolhidos devem ser analisados qualitativamente e quantitativamente.

- **Interpretação / Análise de Resultados**

Nesta etapa é onde se avalia e analisa os resultados alcançados, pondo à prova a veracidade e confiança da hipótese para o problema. Pode acontecer duas situações, os resultados foram positivos e pode-se considerar um trabalho válido e fornecer recomendações para futuros trabalhos, tal como está exposto no capítulo 5. Mas se os resultados não forem os esperados, não se deve considerar um fracasso, mas sim um modo de melhorar a abordagem inicial. É aqui que deve-se voltar às etapas iniciais e tentar uma abordagem diferente.

- **Publicação de Resultados**

Os resultados devem contribuir para a comunidade científica, mais concretamente como artigos científicos. Estes documentos podem ser apresentados em conferências, onde o autor tem a oportunidade de mostrar pessoalmente as suas ideias, apresentando os resultados obtidos. Esta etapa deve ser finalizada com uma dissertação sobre a hipótese.

1.2. Organização da Dissertação

Esta dissertação está dividida em seis capítulos, exemplificando as seguintes características.

O primeiro capítulo começa com o contexto e motivação pessoal para desenvolver este trabalho e originando a questão de pesquisa. Ainda está presente o método científico, que é a abordagem usada para realizar este trabalho. O capítulo termina com esta estrutura do resumo da dissertação.

O capítulo seguinte, Estado de Arte, vem contemplado com os Algoritmos de Otimização de Rotas e Plataformas de Recolha de Dados, e são onde estão alguns estudos realizados antes deste trabalho. É apresentado algoritmos de otimização de rotas relevantes para o apoio deste trabalho e também é feito um estudo de algumas plataformas e soluções desenvolvidas anteriormente.

Sistema Inteligente para Otimização de Rotas é o terceiro capítulo. Aqui apresenta-se a solução criada para definir a rota ideal, com a explicação de cada módulo presente na solução e o seu funcionamento.

No capítulo quatro, Testes e Validação, são apresentados os testes usados para validar a hipótese formulada. Inicialmente está a descrição da metodologia adotada para testar a hipótese e posteriormente são apresentados os resultados, que através da sua análise é verificado se os objetivos iniciais foram alcançados e o cenário industrial validado.

Por fim, o capítulo cinco resume esta dissertação, destacando os aspetos mais importantes do trabalho realizado. Ainda foca os resultados obtidos e uma potencial direção para futuros desenvolvimentos.

Capítulo 2 – Estado de Arte

2.1 Algoritmos de Otimização de Rotas

A área da otimização de rotas apresenta um elevado grau de complexidade, fazendo recorrer a modelação matemática e uso tecnológico de modo a simplificar o problema. Existem diversas hipóteses de resolver parte dessas dificuldades no âmbito do problema apresentado, tal como as expostas no seguinte exemplo.

Um camião ao atravessar uma floresta fica perdido. E com a bateria do GPS e telemóvel quase a acabar, apenas consegue avisar a sede da sua situação atual e que estava perdido no vale mais fundo da região. Com base na informação recebida, foi disponibilizado outro camião para ir ao seu auxílio. Visto ser uma região extensa e com pouca informação, a sede vê-se dividido em três maneiras de tentar encontrar o primeiro camião:

- A primeira maneira é de enviar um helicóptero para percorrer a região e identificar todos os vales existentes e o mais fundo seria o local exato do primeiro camião.
- A segunda maneira é que ao longo do tempo fosse escolhida aleatoriamente uma direção identificando os vales existentes e então seria escolhido o vale mais fundo até à data.
- A terceira e última maneira seria o meio-termo dos dois primeiros. Basear em informações deixadas pelo primeiro camião e utilizar para determinar conjuntos de regiões menores e aí intensificar a procura pelo vale mais fundo.

Assim teríamos diferentes focos:

- A primeira ideia, que normalmente encontraria o local exato do vale mais fundo, conhecido em otimização como métodos exatos, não seria aceite, porque o tempo gasto na procura comprometeria a qualidade dos produtos a ser transportados tal como a integridade física do condutor.
- A segunda ideia, provavelmente também não seria aceite porque a procura seria feita sem qualquer informação prévia da região. A isto chamamos de heurísticas, cuja excessiva flexibilidade pode levar a locais perto do local exato.
- A terceira ideia, conjuga aspetos que permitem evitar os erros dos dois anteriores, pois leva em consideração informações previamente conhecidas da região. O que em otimização pode ser descrito como metaheurísticas.

No entanto, com a melhoria significativa da tecnologia e a facilidade de recursos, podemos recorrer a qualquer uma das três ideias para formular o nosso problema, ou seja, os métodos

exatos e as heurísticas não se tornaram obsoletos. Apenas necessitam de melhorias ou acesso a melhores recursos tecnológicos.

De seguida, são apresentados mais detalhadamente alguns dos métodos mais importantes para a resolução do nosso problema.

2.1.1. Algoritmos de Métodos Exatos

Os métodos exatos são algoritmos de pesquisa exaustiva que verificam todo o conjunto de soluções de determinado problema até encontrar a solução ótima. Sendo esta a sua principal vantagem, ou seja, garantem a solução ótima. No entanto, a sua modelação torna-se mais complexa e a sua aplicação a problemas mais complexos ou de grandeza maior faz com que tenha grandes dificuldades a encontrar a solução ótima num intervalo de tempo adequado. Ou seja, o esforço computacional para a sua resolução cresce exponencialmente (Roque & Junior, 2006).

- Branch and Bound

Branch and Bound é o típico algoritmo pertencente aos métodos exatos. Baseia-se em três partes distintas, como refere (Filipe et al., 2003), ou seja, inicia-se com a construção de árvores de nós onde divide o problema em conjuntos menores ou subproblemas menores. De seguida cria estratégias para escolher o próximo subproblema e consequentemente realiza comparações com o seu limite superior e inferior (parâmetros a estabelecer), onde uma solução tem que obedecer para ser considerada viável.

(Polit, 2006) ainda refere que a seleção do nó para ramificação, pode ter um papel importante na eficácia do algoritmo. E que a inserção dos parâmetros pode ser feita de dois modos: regras definidas que são determinadas antes do início do processamento ou regras adaptativas, que variam consoante as informações dos nós ativos e do processamento corrente.

Na figura 2.1, encontra-se um exemplo ilustrativo deste algoritmo. O seu objetivo é encontrar o caminho mais curto entre duas cidades. O algoritmo analisa as distâncias entre cada cidade e organiza através de uma árvore ramificada. A ordem de procura do caminho está disposta com as bolas azuis numeradas, até ser encontrada a solução. Na parte de cima da figura encontra-se as cidades e as suas distâncias e na parte inferior a árvore ramifica organizada.

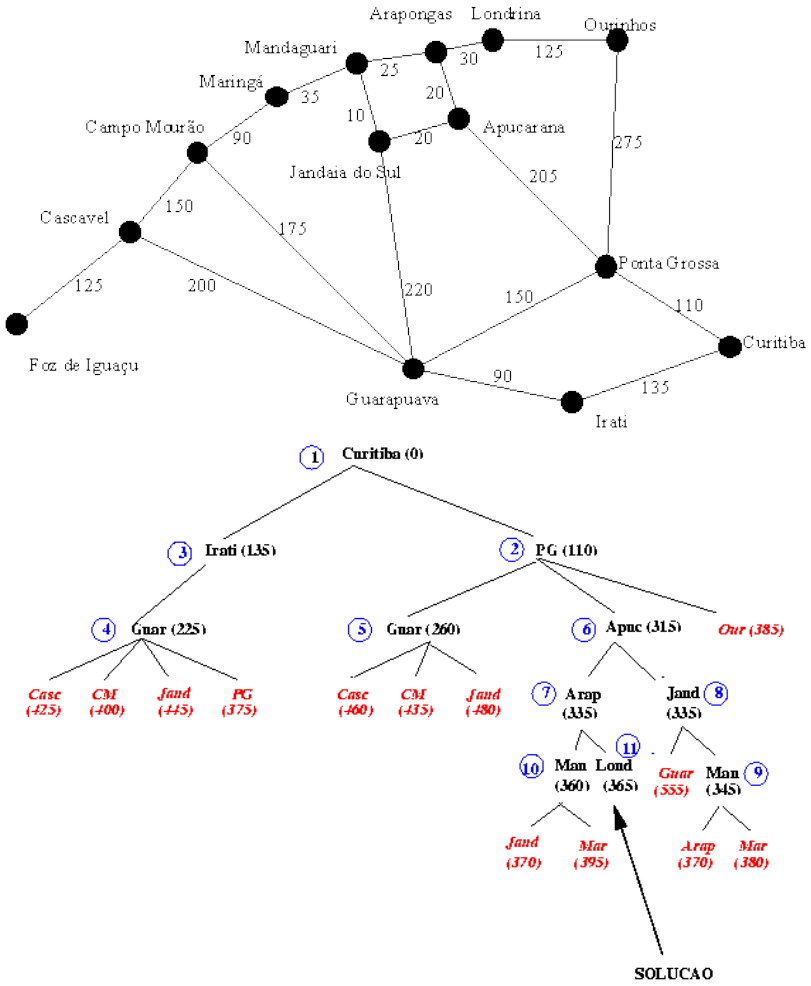

Figura 2.1 – Exemplo do Algoritmo Branch and Bound. (“www.professeurs.polymtl.ca,” 2000)

Nesta procura minuciosa, o algoritmo é eficiente a encontrar a solução ótima e adaptável a outras situações devido aos seus limites. Em contrapartida, peca um pouco pela sua complexidade e pela sua lentidão em problemas bastante mais complexos.

- Branch and Cut

Este algoritmo pode ser determinado como uma derivação ou aperfeiçoamento do algoritmo Branch and Bound, com a particularidade de ter um procedimento de corte, ou seja, uma restrição adicionada ao método, como demonstrado na figura 2.2. O propósito de adicionar um corte é para limitar o tamanho do domínio das soluções. Assim sendo, o algoritmo realiza decomposições do domínio de soluções com a ramificação e em seguida corta esses subdomínios com o intuito de reduzir regiões que não possuam soluções viáveis (Botinˇ, 2006).

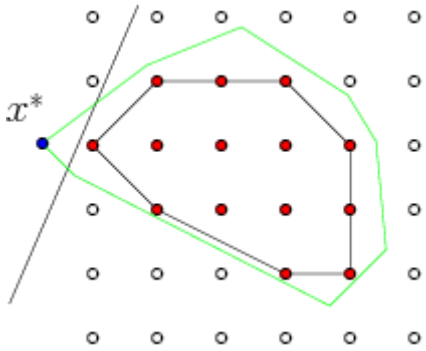

Figura 2.2 – Exemplo do algoritmo Branch and Cut. (“kuniga.wordpress.com,” 2010)

Em termos de performance este algoritmo é muito semelhante ao Branch and Bound, com o acréscimo do procedimento de “Cut”, em que pode aumentar a rapidez a encontrar as soluções ótimas. Por vezes, um corte pode não ser muito eficaz porque pode eliminar a solução ótima, conseguindo apenas encontrar soluções perto do ótimo.

- Algoritmo de Dijkstra

É um algoritmo um pouco diferente da categoria a que pertence (métodos exatos). A sua função é encontrar o caminho mais curto entre duas arestas dentro do mesmo grafo, ou seja, calcula o custo mínimo de um vértice para todos os outros vértices do grafo.

Em (Chao, 2010) refere que o algoritmo parte de uma estimativa inicial para o custo mínimo e vai sucessivamente ajustando essa estimativa. À medida que vai percorrendo o grafo, ele vai considerar que um vértice está fechado quando for obtido o custo mínimo do vértice do caminho principal. Caso não seja, ele considera aberto. Esta análise é realizada a partir dos valores entre cada vértice, neste caso é considerado as distâncias. Quando todos os vértices tiverem sido executados, a solução obtida será a de menor custo tal como ilustrado na figura 2.3.

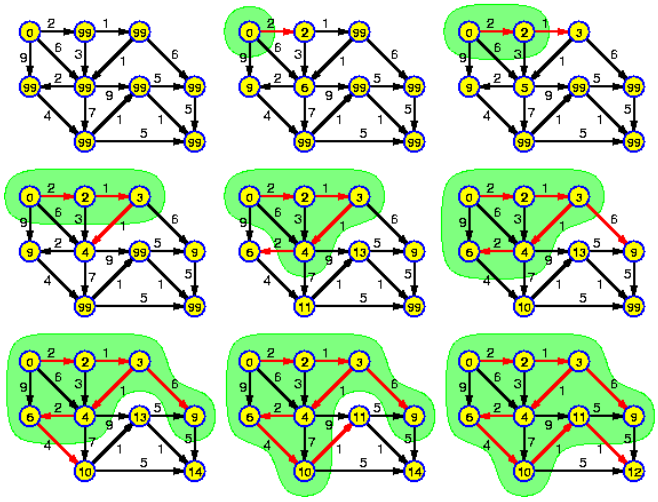

Figura 2.3 – Exemplo do algoritmo Dijkstra.(Jasika et al., 2012)

Este algoritmo é bastante simples e com um bom nível de performance. Não garante, contudo, a exatidão da solução caso haja a presença de arcos com valores negativos. Por outro lado, em situações mais complexas ou mais abrangentes, a velocidade pode diminuir bastante.

2.1.2. Heurísticas

Os algoritmos heurísticos foram desenvolvidos para evitar uma elevada demora na procura de uma solução ótima, ou seja, que fossem de fácil implementação e que produzissem boas soluções rapidamente.

Num determinado problema o ideal seria analisar todas as soluções possíveis para chegar à melhor. No entanto, as heurísticas têm a desvantagem de explorar parcialmente a totalidade das soluções possíveis, ou seja, adotam uma estratégia que se apoiam numa abordagem intuitiva (Malaquias, 2006).

As heurísticas ainda carecem de não garantir se uma solução é ótima ou o quão próximo está da solução ótima, mas encontram soluções boas num tempo razoável (SILVA, 2013).

- Algoritmo Savings

Algoritmo Savings é um algoritmo baseado nas poupanças. Ele tenta encontrar o custo de rotas através da combinação de duas rotas em apenas uma, como mostra a ilustração seguinte.

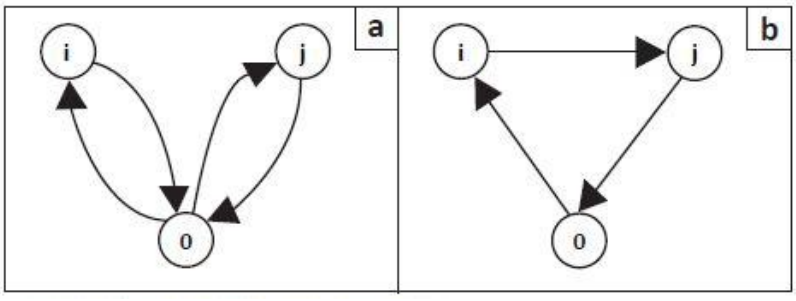

Figura 2.4 - Exemplo de Algoritmo Savings (“www.jtscm.co.za,” 2013)

Este algoritmo tem duas visões de abordar o problema, uma delas é visitar os clientes *i* e *j* em rotas separadas (2.4a). Em alternativa é fazê-lo na mesma rota, ou seja, visitar os clientes sequencialmente (2.4b). Visto que os custos das rotas são dadas, o resultado pode ser calculado para saber qual a melhor rota. Ainda é possível dividir o algoritmo em duas versões, sequencial e paralela. Na versão sequencial apenas é formada uma rota de cada vez, enquanto na versão paralela é criada mais que uma rota de cada vez (Lysgaard, 1997).

É um algoritmo simples e fácil de usar, no entanto apenas apresenta soluções perto do ótimo.

- Nearest Neighbor

Esta heurística é um algoritmo simples e intuitivo. O seu principal objetivo é procurar qual o próximo cliente a visitar, com a finalidade de gerar poupanças de rotas. Como exemplificado na figura 6, começa-se de um ponto inicial, que no caso das rotas de veículos é a sede. A cada ponto que o veículo visita, o algoritmo observa os vizinhos do ponto atual e procura o vizinho mais próximo e viável. O cálculo de proximidade não leva em conta somente a distância entre os clientes.

Este algoritmo como refere (SILVA, 2013), tem uma metodologia que constrói a solução passo a passo, segundo um conjunto de critérios pré-estabelecidos. Este processo iterativo repete-se até que não exista mais nenhum cliente a ser visitado, ou seja, todos os elementos sejam percorridos e a rota seja concluída. Neste exemplo da figura 2.5, o número total de clientes é 36 e vai corresponder a uma rota total de 55,69km.

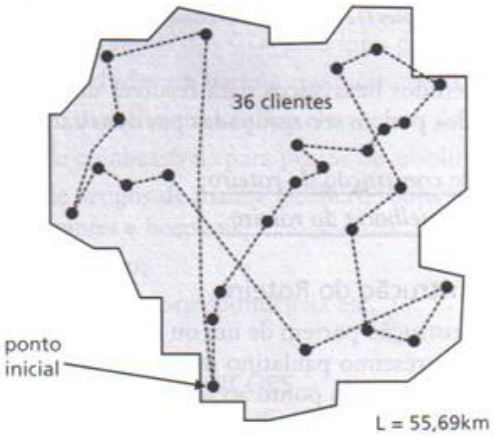

Figura 2.5 - Exemplo do Algoritmo Nearest Neighbor. (“slideplayer.com.br,” 2007)

Visto ser semelhante ao algoritmo Savings, este apenas encontra soluções perto do ótimo e é mais lento. No entanto, em comparação com os algoritmos dos métodos exatos, este é bastante mais simples.

- A-Star

Este algoritmo é uma heurística que encontra um caminho desde um ponto inicial até a um ponto final. Apresenta como principais características a intuição que é muito vista nos algoritmos heurísticos e a rigorosidade a encontrar o caminho ideal eficientemente dos algoritmos dos métodos exatos. Este facto deve-se porque o algoritmo A-Star foi uma tentativa de aperfeiçoar o algoritmo Dijkstra em 1968 pelos autores (Hart, Nilsson, & Raphael, 1968).

O algoritmo A-Star usa outras heurísticas para decidir qual o caminho que deve seguir e apresenta performances melhores do que o algoritmo Dijkstra, em termos de tempo de execução (Ji, Liu, Zhao, & Wang, 2015).

Para melhor exemplificar este algoritmo temos a figura 2.6, que demonstra as decisões feitas para alcançar o caminho ideal. Os números 1's são as paredes que terão que ser contornadas. Assim o A-Star começa por percorrer junto às paredes efetuando o percurso e quando depara-se com um espaço mais aberto em que tem várias hipóteses para continuar o caminho (parede ao meio da figura), ele vai adotar uma abordagem intuitiva, proveniente das heurísticas, tentando percorrer o menor caminho até alcançar a parede mais à direita

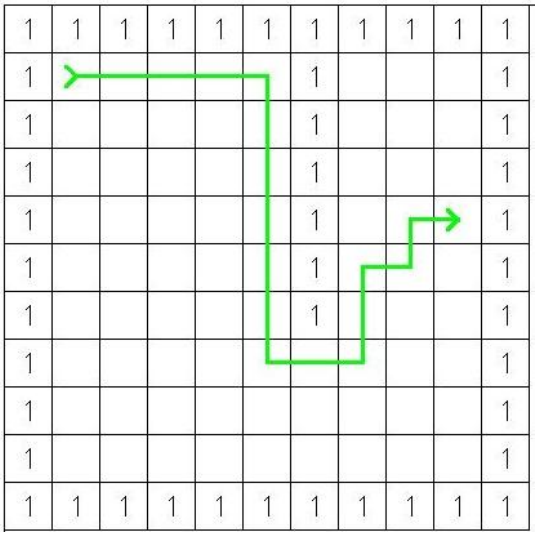

Figura 2.6 - Exemplo do algoritmo A-Star ("michalhr.ehost.pl," n.d.)

Este algoritmo apresenta uma abordagem bastante interessante, ao combinar dois tipos de algoritmos, os métodos exatos e as heurísticas. No entanto, acarreta sempre os problemas mais típicos destes tipos de algoritmos, o tempo de execução e a procura do caminho pela intuição, mas em menor escala, o que poderá ser bastante vantajoso.

2.1.3. Algoritmos de Metaheurística

Metaheurísticas são algoritmos que encontram boas soluções e por vezes a ótima. Baseiam-se na aplicação de uma heurística subordinada onde tem de ser modificada para cada problema específico. As suas principais características são evitar as desvantagens dos métodos anteriores, ou seja, ter a capacidade de explorar um leque de soluções onde evita os ótimos locais e encontra soluções num tempo razoável (SILVA, 2013). Os procedimentos criados a partir dos métodos metaheurísticos têm tido um grande foco neste tipo de estudos mais complexos, pois têm-se tornado eficientes.

- Simulated Annealing

O método de Simulated Annealing é uma metaheurística amplamente divulgada na comunidade científica utilizada para lidar com problemas de otimização discreta e, com menor abrangência, alguns problemas de natureza contínua. Exemplifica uma analogia com o comportamento termodinâmico, mais concretamente no processo de arrefecimento de sólidos (César, Oliveira, Vasconcelos, & Bastos, 2006).

Como refere (Gheisari & Haghighat, 2008), o algoritmo em cada iteração compara a solução atual com o seu vizinho e avalia a sua qualidade. Soluções com melhor qualidade são sempre aceites, ao passo que soluções menos boas são aceites com uma probabilidade que diminui com a temperatura. Para altas temperaturas, a probabilidade é alta, mas diminuindo a temperatura também faz com que a probabilidade diminua. Assim sendo, à medida que o parâmetro temperatura tende para zero as soluções menos boas serão aceites com menor frequência, logo nas últimas iterações o algoritmo irá convergir para um ótimo local, que poderá ser ou não um ótimo global.

Na figura 2.7 temos a disposição de uma cidade onde necessitamos de encontrar um caminho para visitar todos os pontos. Inicialmente é definido uma temperatura inicial e criada uma solução aleatória. De seguida entra em “loop” até atingir uma condição de paragem, que normalmente é encontrar uma solução boa. A partir daqui é selecionar um vizinho, diminuir a temperatura e continuar com os “loops”. Esta sequência repete-se até encontrar a solução final.

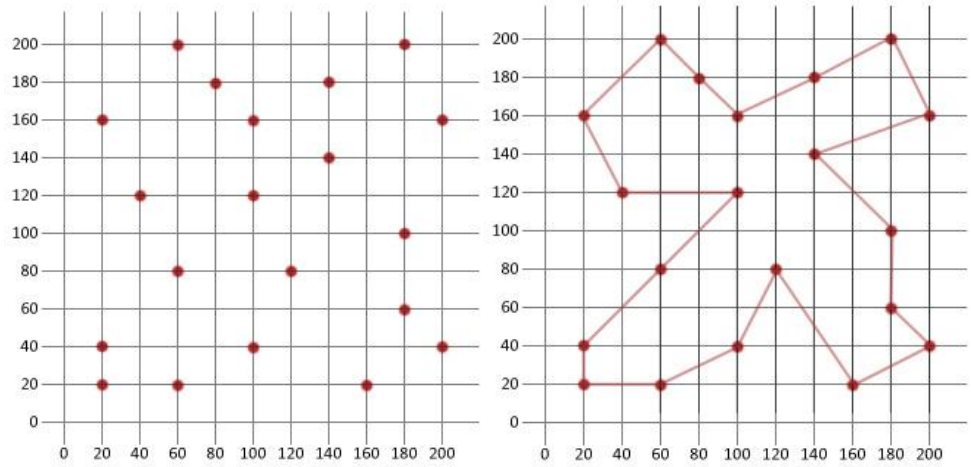

Figura 2.7 - Exemplo do Algoritmo Simulated Annealing. (“www.theprojectspot.com,” 2013)

Não é um algoritmo rápido, no entanto carateriza-se por ser eficiente e mesmo tendo muitos parâmetros para “calibrar” é bastante flexível. Depende sempre de uma solução inicial aleatória e normalmente encontra a solução ótima. É bastante eficaz a trabalhar em conjunto com outros algoritmos, tanto em paralelo como sequencialmente.

- Tabu Search

O Tabu Search explora o espaço de soluções movendo-se em cada iteração, da solução atual para a melhor solução encontrada num subconjunto da vizinhança. Pode ser simplesmente vista como uma combinação de pesquisa local com memórias de curta duração. É um algoritmo que tenta evitar ficar preso em ótimos locais ou retornar a um previamente visitado.

Como refere (Charbonneau & Vokkarane, 2010), inicialmente deve ser gerada uma solução aleatória ou proveniente de outro algoritmo. A partir desta solução inicial serão geradas novas soluções e a melhor será escolhida como solução atual como ilustrado na figura 2.8. Por vezes, os algoritmos tendem a bloquear em ótimos locais e para evitar isso, este algoritmo cria uma lista tabu. (Guan, 1986) refere que essa lista tabu regista movimentos que foram usados para gerar as soluções selecionadas, para que não possam ser realizados outra vez enquanto estiverem na lista. Concluindo assim que a lista tabu é um dos principais fatores que determinam a qualidade do algoritmo Tabu Search.

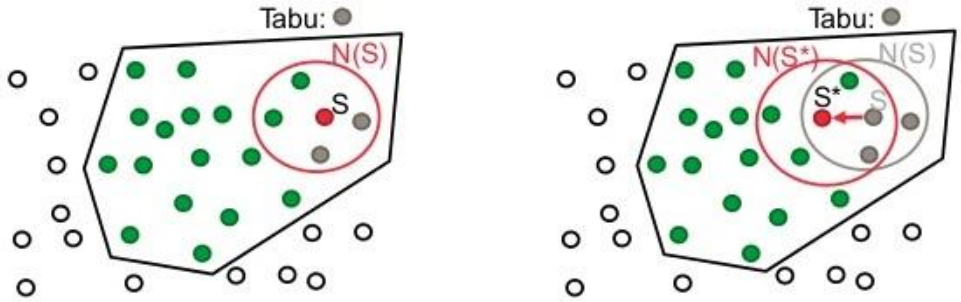

Figura 2.8 - Exemplo de Tabu Search. (“www.slideshare.net,” 2015)

Devido ao uso excessivo de memória, este algoritmo pode-se tornar lento, dependendo do número de iterações as soluções ficam guardadas. No entanto, é bastante eficiente e atinge com facilidade a solução ótima ou próxima da ótima. Em conjunto com outros algoritmos pode-se tornar mais eficiente.

- Algoritmos Genéticos

Algoritmos Genéticos são algoritmos robustos, usados nos mais variados problemas de diferentes domínios. São baseados em procedimentos de seleção natural e de genética (Botassoli, Alberti, & Furtado, 2015).

Este algoritmo possui alguma inteligência na procura de soluções que é feita através de processos iterativos, onde cada iteração é chamada de geração. Durante cada geração, os princípios de seleção e reprodução são aplicados a uma população de candidatos que pode variar, dependendo da complexidade do problema e dos recursos computacionais disponíveis.

O princípio básico do funcionamento dos Algoritmos Genéricos é que um critério de seleção vai fazer com que, depois de muitas gerações, o conjunto inicial de indivíduos gere indivíduos mais aptos. Tal processo passa por várias etapas, como refere (Pereira, Nascimento, & Barbosa, 2012): o cálculo da aptidão, onde é feita uma análise para saber se as soluções respondem bem ao problema proposto; a seleção, onde são escolhidas as soluções para a reprodução (a probabilidade de uma solução dada ser selecionada é proporcional à sua aptidão); o cruzamento, onde as soluções escolhidas são recombinadas, gerando novos indivíduos; e por fim a mutação, onde os indivíduos resultantes do processo de reprodução são alterados, acrescentando assim variedade à população. Tais processos estão exemplificados de um modo simples na figura 2.9.

A maioria dos métodos de seleção são projetados para escolher preferencialmente indivíduos com maiores índices de aptidão, embora não exclusivamente, a fim de manter a diversidade da população.

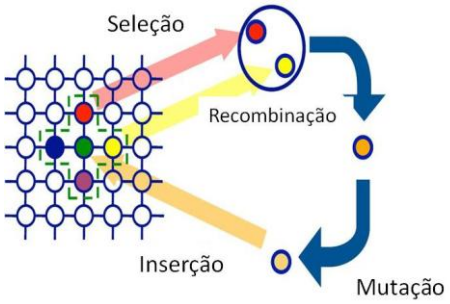

Figura 2.9 - Exemplo de Algoritmos Genéticos. (Botassoli et al., 2015)

Carateriza-se pela sua simplicidade e é eficiente a trabalhar em conjunto com outros algoritmos. Na sua versão simples não é muito competitivo, no entanto a sua versão hibrida pode ser mais eficiente e mais assertivo a encontrar a solução ótima.

- Sistema de Colónia de Formigas

O Sistema de Colónia de Formigas é mais uma metaheurística inspirada em fenômenos da natureza. A ideia principal é a probabilidade de as formigas encontrarem o caminho para obter alimento.

A regra básica em torno de um sistema de colónia de formigas, como refere (Rana, Thulasiraman, & Thulasiram, 2013) é que cada vez que uma formiga encontra uma rota para o alimento, vai libertando a sua feromona pelo caminho. Ao longo do tempo, as outras formigas vão fazendo exatamente o mesmo, originando rotas com diferentes níveis de feromonas. O resultado quando o processo está amadurecido é que um caminho de feromonas prevalecerá e esse será o caminho viável e de menor distância. Este algoritmo está ilustrado na figura 2.10, dando como exemplo o facto de as formigas encontrarem um obstáculo no seu caminho e demonstrando o seu percurso para encontrar o caminho que prevalecerá.

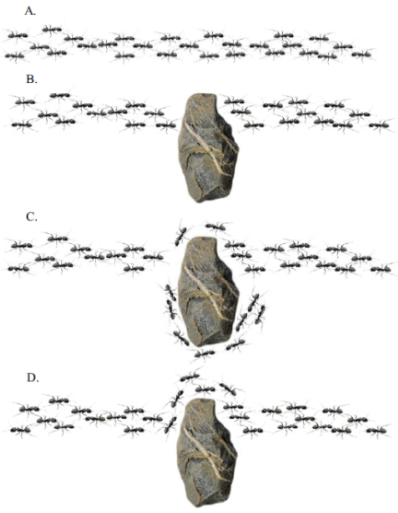

Figura 2.10 - Exemplo do Sistema de Colónia de Formigas. (Lin, Chen, & Chang, 2014)

Este algoritmo para determinados problemas atinge melhores desempenhos que as outras metaheurísticas. No entanto, peca por ser um algoritmo complicado na sua análise teórica e a sua otimização não é direta. Pode ser executado continuamente e adaptar-se a mudanças em tempo real. Não é um algoritmo rápido e nem sempre encontra a solução ótima.

- GRASP (Greedy Randomized Adaptive Search Procedure)

Este algoritmo foi introduzido por (Feo & Resende, 1995) e é uma metaheurística de processos iterativos que gera normalmente soluções perto do ótimo. A ideia principal deste algoritmo é criar soluções novas independentes das anteriores e na sua versão básica divide-se em duas fases em cada iteração, a fase de construção e uma fase de procura local.

A fase de construção consiste em encontrar uma solução viável para o problema. E a fase de procura local é usada para melhorar essa solução encontrada. Estas duas fases ocorrem até encontrarem um critério de interrupção como exemplificado na figura 2.11.

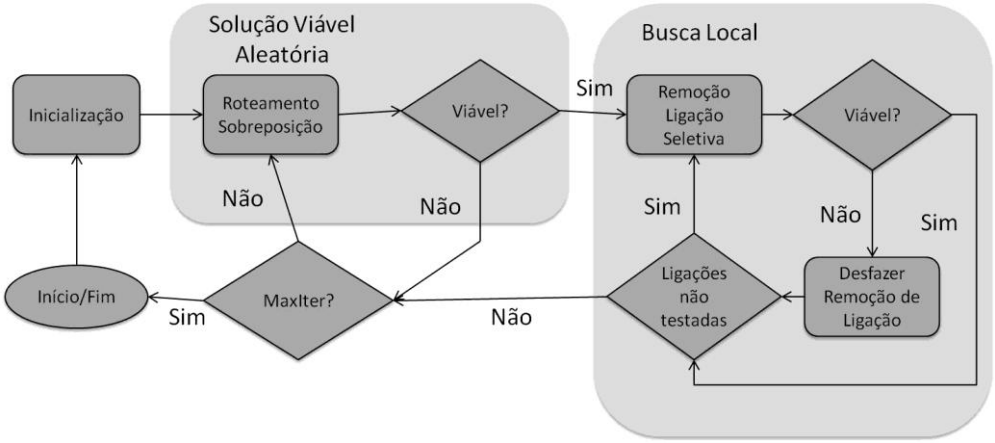

Figura 2.11- Exemplo de Algoritmo GRASP. (Risso, Robledo, & Sartor, 2013)

Durante a fase de construção, em cada iteração, é formada aleatoriamente uma solução viável de cada vez. Mas nem sempre garante que seja formada a melhor solução. Por isso, segue-se a fase de procura local, onde vai tentar melhorar a solução construída. Ou seja, vai substituindo sucessivamente a solução construída por uma melhor na proximidade, até não haver melhorias na solução. Resumidamente, esta metaheurística pode ser considerada um método sem memória, porque cada iteração é independente e não passa nenhuma informação sobre as soluções às outras iterações (Gonc, Ochi, Martins, & Janeiro, 2005).

Este algoritmo carateriza-se por ser rápido e simples. Normalmente encontra soluções boas mas não necessariamente as soluções ótimas. Depende de outros algoritmos no seu processo de construção inicial e tem poucos parâmetros para serem ajustados. Ainda tem uma particularidade que pode ser um algoritmo que dependa de outros algoritmos, ou seja, as suas duas fases de funcionamento podem ser derivadas de outros algoritmos, tanto na fase de construção como na fase de procura local.

2.1.4. Análise e Discussão

A tabela 2.1 vem resumir todas as informações recolhidas das referências estudadas neste capítulo. É feita uma pequena análise sobre alguns aspetos importantes para o funcionamento de um bom algoritmo. Como é visível, estes algoritmos não são perfeitos, visto não serem ótimos em todas as características. No entanto, cada algoritmo apresenta algo que pode ajudar a melhorar o processo de otimização de uma rota. De seguida, são apresentados alguns dos aspetos mais relevantes de cada algoritmo, tanto positivos como negativos, sobre as características mais importantes.

O Branch and Bound pode não ser muito útil devido à velocidade em encontrar a solução ótima, mas para testar ou analisar pequenos subproblemas pode ser uma boa ferramenta de auxílio. Ainda consegue apresentar alguma adaptabilidade a diferentes problemas. Por outro lado, o Branch and Cut ainda pode acrescentar uma base em termos de restrições mas devido à sua lentidão a encontrar soluções ótimas não é dos melhores. Por último, nos métodos exatos o Dijkstra tem a sua utilidade mas peca pela velocidade, no entanto, o facto de encontrar os caminhos mais curtos, poderá ser uma base para iniciar a resolução do problema, visto ser um algoritmo simples e com níveis de rapidez melhores comparativamente com outros algoritmos.

Nas Heurísticas, o algoritmo Savings pode ser um algoritmo um pouco lento em comparação com as metaheurísticas, mas bastante mais eficaz que os métodos exatos. É um algoritmo simples, mas pouco flexível, o que faz com que seja mais utilizado em problemas de roteamento de veículos. Nem sempre encontra a solução ótima, mas sim perto do ótima. Por seu lado, o Nearest Neighbor é um bom algoritmo para o nosso problema dependendo dos

parâmetros ou restrições que usarmos, no entanto funciona melhor em conjunto com outro algoritmo, apresentando características que podem complementar esse algoritmo.

O algoritmo A-Star mostra-se bastante eficiente a trabalhar com outros algoritmos. E sendo um aperfeiçoamento do algoritmo Dijkstra não apresenta os mesmos problemas que os algoritmos dos métodos exatos, a velocidade lenta e a flexibilidade.

Tabela 2.1 - Comparação entre os vários Algoritmos

Método	Algoritmo	Precisão	Velocidade	Simplicidade	Flexibilidade	Compatibilidade
Exato	Dijkstra	X	V	V	X	X
	Branch and Bound	V	X	X	V	X
	Branch and Cut	V	X	X	V	X
Heurística	Savings	X	V	V	X	X
	Nearest Neighbor	X	X	V	V	V
	A-Star	V	X	X	V	V
Metaheurística	Simulated Annealing	V/X	X	V	V	V
	Tabu Search	V	X	-	V	V
	Genéticos	X/V	X/V	V	V	V
	Colônia de Formigas	X	X	X	V	V
	GRASP	X	V	V	V	V

Já nas Metaheurísticas, o Simulated Annealing não tem todos os requisitos num nível elevado, mas é o algoritmo que apresenta melhores resultados a obter a solução ótima. É simples e flexível, o que poderá ser um bom algoritmo para trabalhar individualmente ou em conjunto com outros algoritmos. O Tabu Search é um bom algoritmo, apresentando características boas para o problema mas terá que ser estudado mais aprofundadamente para perceber o quanto o

uso de memória pode afetar a sua eficiência em problemas mais complexos, deixando algumas dúvidas no nível de qualidade das suas características.

Tal como as outras Metaheurísticas, os Algoritmos Genéticos apresentam níveis bons das suas características. No entanto, só é justificável utilizá-lo numa versão híbrida ou mesmo utilizá-lo em conjunto com outro algoritmo, porque na sua versão mais simples peca pela sua rapidez e por obter resultados perto da solução ótima.

O algoritmo de Sistema de Colônia de Formigas devido às suas características de tempo real é dos melhores algoritmos para se utilizar, mas nos outros requisitos atinge níveis pouco satisfatórios comparativamente aos algoritmos do seu nível. Por fim, o algoritmo GRASP é bastante adaptável a outros algoritmos, visto por vezes depender das características dos outros para trabalhar. Com isto, é um algoritmo que apresenta bons níveis de performance, pecando apenas na determinação da solução ótima, o que nem sempre acontece.

2.2 Plataformas de Recolha de Dados

Na otimização de rotas nem sempre dependemos apenas dos algoritmos para determinar o melhor percurso a percorrer. No dia-a-dia deparamo-nos com inúmeros eventos que podem prejudicar ou melhorar o percurso de determinado trajeto. Esses eventos podem ser determinantes e por isso seria ideal saber de antemão que determinada rota vai estar congestionada a dada altura ou que será impossível circular numa rota a determinado horário, entre muitas outras situações.

A informação sobre os eventos pode ter bastante influência na determinação de uma rota, logo, é preciso recolher essa informação e ter acesso a ela. A dificuldade apresenta-se em como recolher esses dados e se a mesma está de acordo com a realidade.

Com a evolução tecnológica deparamo-nos com um grande leque de infraestruturas ou programas que podem facilitar a recolher esses dados. No entanto, nem sempre a informação está disponível para se usar de modo independente. Dependemos de aplicações e programas para saber algumas dessas informações sobre o trânsito.

De seguida, são apresentados algumas plataformas e métodos de recolha de dados do trânsito.

2.2.1. CarWeb

Segundo (Lo, Peng, Chen, Lin, & Lin, 2008), CarWeb é uma plataforma de recolha de dados em tempo real. Essa recolha é feita, periodicamente, através de redes sem fio para os servidores da CarWeb. Esta plataforma é baseada num sistema de cliente-servidor, como ilustrado na arquitetura da figura seguinte.

Figura 2.12 – Arquitetura da CarWeb (Lo et al., 2008)

Os dados GPS são recolhidos para os servidores através dos dados móveis do telemóvel e da rede sem fios. Esses dados são nomeadamente os parâmetros de latitude, longitude e velocidade em milhas por hora e tempo. As bases de dados dos servidores foram feitas a partir de uma “open source” com sistema de informação geográfico denominado PostgreSQL. Por último os mapas foram fornecidos pelo API do Google Maps/UrMap. Englobando todos estes métodos, a CarWeb disponibiliza uma interface Web para mostrar o trânsito em tempo real.

2.2.2. DeCloud4SD

DeCloud4SD é uma plataforma de processamento integrado para dados de sensores de trânsito, tais como dados GPS e RFID, onde recebe, armazena, adquire e computa esses dados numa arquitetura escalável. Assegurando sempre uma recolha de dados em tempo real (Zhao, Fang, Ding, & Wang, 2014).

Esta plataforma vai de encontro a algumas exigências do projeto do Sistema Inteligente de Transportes de Beijing, sendo aplicável em situações reais como: recolha de dados de sensores de trânsito escalável e personalizável; desenvolvimento e implementação rápida de aplicações em tempo real usando o modelo MapReduce; e integração com fontes de dados existentes. Estas situações são mais específicas para detetar comportamentos irregulares dos veículos, monitorizar veículos suspeitos e medir o fluxo do trânsito.

Na figura 2.13 encontra-se a arquitetura apresentada para a plataforma de DeCloud4SD, que se divide em 4 grandes componentes. O “Streaming Data Communicator” é responsável por analisar e validar os dados recolhidos, tal como verificar se os sensores estão a funcionar corretamente. O “Streaming Data Dispatcher” remete os dados fiáveis para as outras componentes da arquitetura, usando um “middleware” de “open source” denominado ActiveMQ, em que garante a receção em tempo real dos dados de fluxo do trânsito. Na componente “Data Proxy” é recolhida toda a informação para ser posteriormente organizada e armazenada numa base de dados, para mais tarde ser consultada. Por fim, o principal componente desta arquitetura é o “Real-time Computing Engine”, onde usa uma “open source” denominada MapReduce para suportar a computação em tempo real do fluxo de trânsito da base de dados em grande escala. Esta arquitetura desempenha um serviço de coordenação distribuída descentralizada para implementar as componentes referidas anteriormente. Isto é possível através da “open source” denominada ZooKeeper.

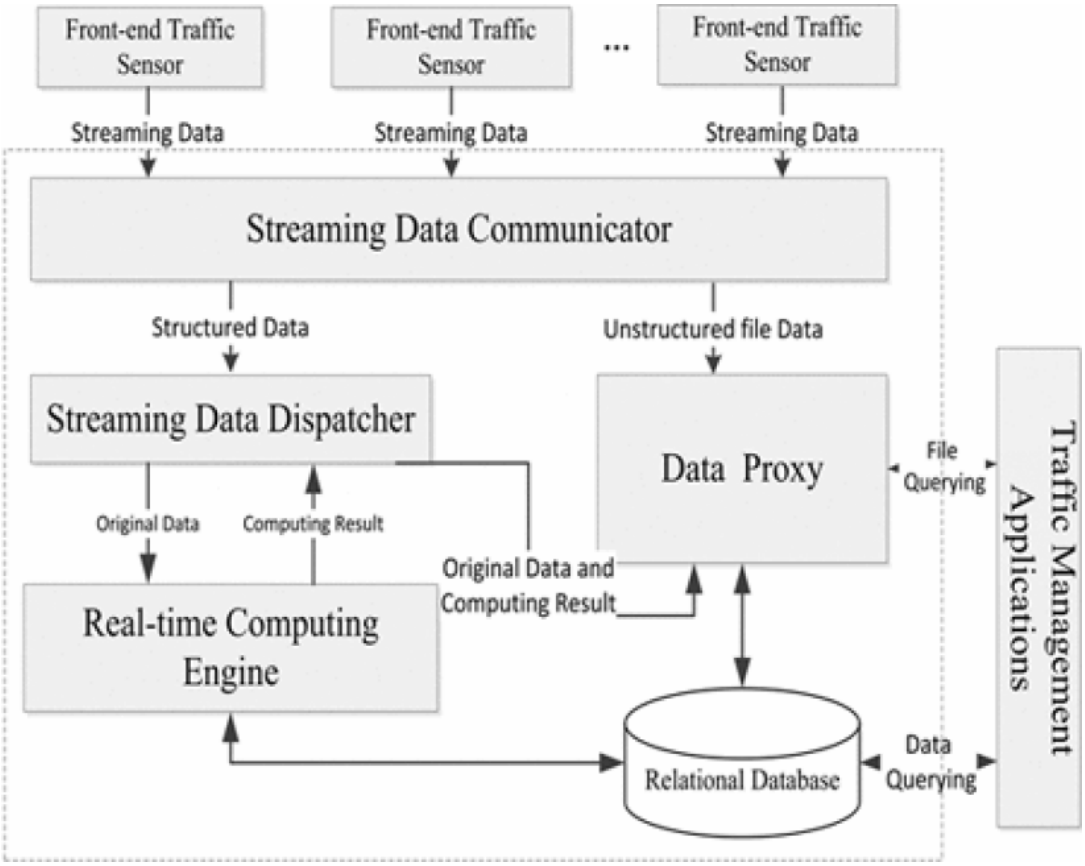

Figura 2.13 – Arquitetura da plataforma DeCloud4SD (Zhao et al., 2014)

Esta plataforma tem serviços que podem auxiliar os nossos objetivos para este trabalho, nomeadamente a utilização de “open sources”, que poderemos usar para melhorar ou completar na nossa implementação prática.

2.2.3. MVTDC

MVTDC é um algoritmo de recolha de dados do trânsito baseado em processamento de vídeo. Através das câmaras de vídeo dispostas nas estradas é feita a recolha de dados do trânsito, onde por vários métodos de processamento de imagem é recolhida a informação do número de veículos, tal como a sua categoria. Ainda é possível contabilizar o fluxo de trânsito e a velocidade dos veículos. Estes métodos estão organizados consoante o seguinte fluxograma (Shuguang, Hongkai, Jingru, & Ke, 2011).

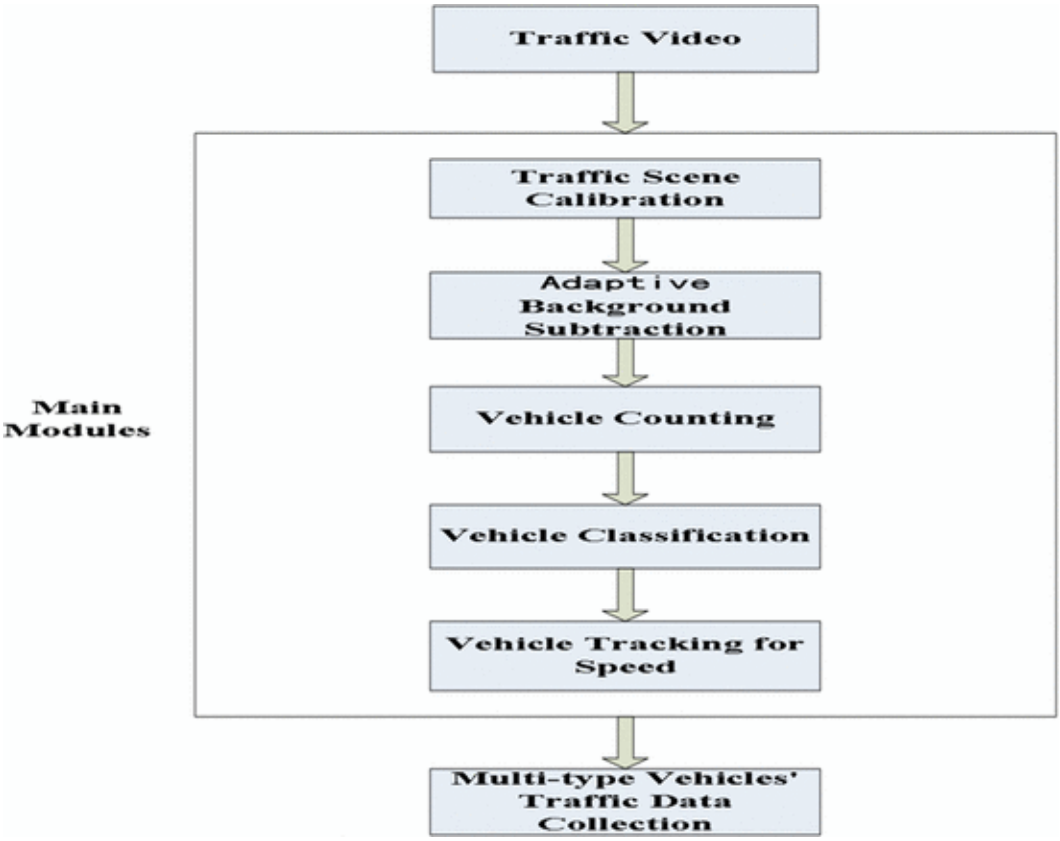

Figura 2.14 - Fluxograma do MVTDC (Shuguang et al., 2011)

Este método não é exímio nas suas funções, apresentando uma eficácia de 90% na contagem dos veículos e ao definir a sua categoria. Por outro lado, também ocorre erros na medição da velocidade na ordem dos 15%. No entanto, estes valores são considerados na maioria dos casos precisos e fiáveis.

Em suma, este método apresenta simplicidade e fiabilidade, mas necessita de trabalho futuro, tal como os problemas com as sombras, a densidade do trânsito e mais alguns parâmetros do fluxo de trânsito.

2.2.4. OLSIMv4

É uma versão atualizada de uma plataforma de informação de trânsito em grande escala, implementada numa situação real de uma rede de autoestradas de North Rhine-Westphalia.

Segundo (Schreckenberg & Luther, 2013), esta plataforma recolhe os dados de trânsito a partir de outros sistemas, quer seja modelos de “open source”, ou sistemas completos e funcionais, até a programas de simulação. Na figura seguinte está representado o fluxograma e os serviços que participam na plataforma OLSIMv4. As linhas a tracejado representam os serviços que estão programados para o futuro.

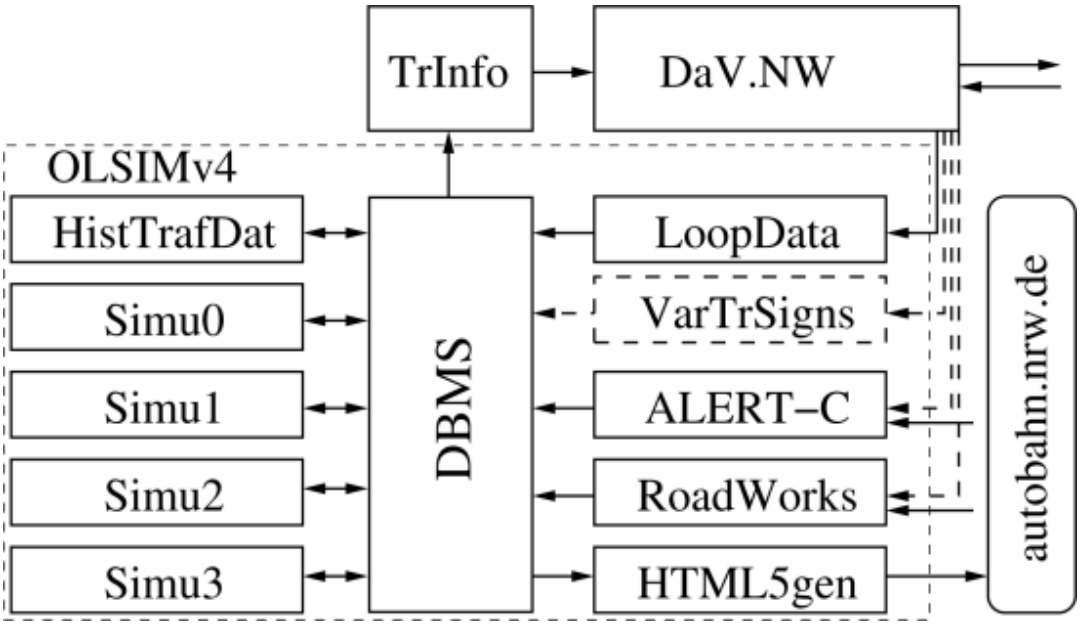

Figura 2.15 - Fluxograma do OLSIMv4 (Schreckenberg & Luther, 2013)

O OLSIMv4 tem como principal foco a recolha de informação por via de outros serviços sobre a rede rodoviária para posteriormente realizar simulações microscópicas, onde antevê o comportamento do fluxo de trânsito. Esta plataforma vai de encontro a uma topologia dinâmica em que tenta minimizar os 5% a 10% de falta de precisão do seu antecessor, uma vez que é implementado um modelo de trânsito mais complexo com uma programação paralela.

Esta plataforma apresenta características bastante relevantes no que toca à recolha de informação do trânsito por parte de outros serviços, o que também será um grande desafio na realização da nossa implementação prática.

2.2.5. STA

STA pode caraterizar-se como um agente de trânsito inteligente, onde de um modo geral o seu foco é informar sobre o trânsito à medida que percorremos determinado percurso e receber informações sobre o estado do trânsito (Lee, Tseng, & Shieh, 2010).

A recolha de dados é feita através de base de dados externos com informação sobre o trânsito atual e passado ao utilizador através de uma interface. Isto é possível pela utilização do GPS, redes sem fios e RDS, como ilustra a figura seguinte.

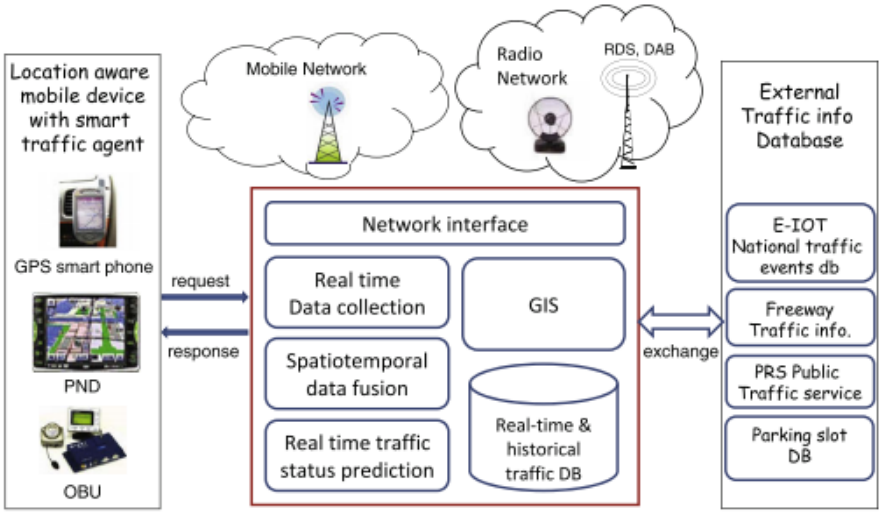

Figura 2.16 - Estrutura da recolha e distribuição de informação de trânsito da STA (Lee et al., 2010)

Este modelo apresenta alguns métodos interessantes, como a comunicação entre utilizador, pois é uma mais-valia para obter informação em tempo real. No entanto, também aborda algumas questões de elevados custos para implementar este tipo de modelo.

2.2.6. CarTel

Segundo (Hull et al., 2006) CarTel é uma plataforma que recolhe os dados de trânsito a partir da informação do GPS. No entanto, para evitar colocar sensores espalhados pelas estradas, a solução apresentada incorpora os sensores de GPS no carro, para assim que houver internet disponível, o sensor comunique com a plataforma para informar sobre o trânsito.

Na figura 2.17 está exemplificada a arquitetura das componentes da plataforma, onde é de salientar que a informação de cada veículo é guardada numa base de dados denominada ICEDB e o CafNet é o método que faz a comunicação entre o veículo e a plataforma.

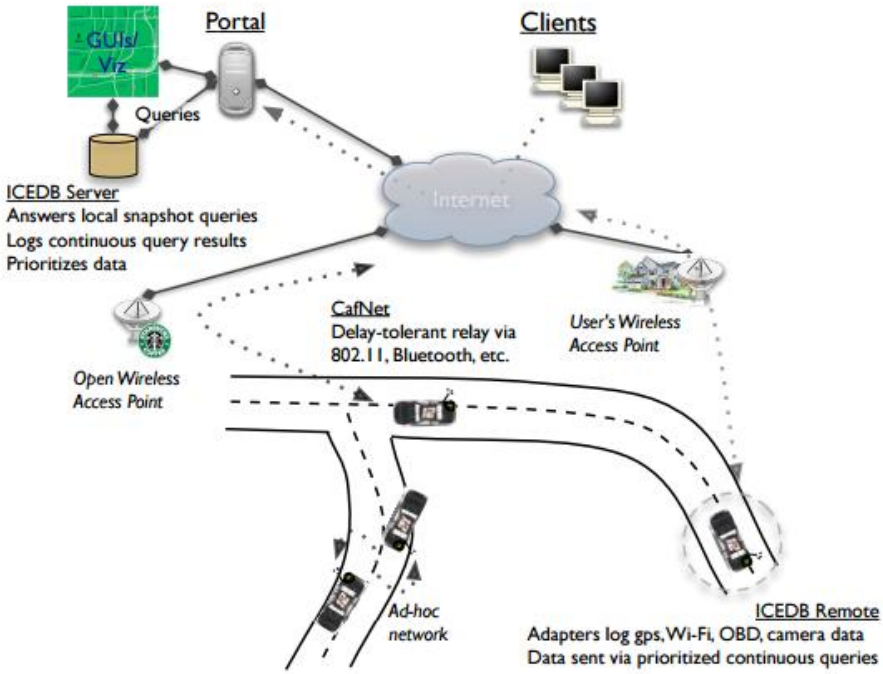

Figura 2.17 - Arquitetura da CarTel (Hull et al., 2006)

Esta plataforma encontra-se limitada no sentido de só ter a recolha de informação de trânsito quando houver ligação à internet disponível. No entanto, é de referir a constante evolução dos "smartphones" e da disponibilidade da ligação à internet, o que pode vir a facilitar a implementação deste tipo de plataformas.

2.2.7. "VNSRTTI"

Esta plataforma realiza a sua recolha de dados de trânsito a partir de vários tipos de informação. Tendo sido implementado em Beijing, existe uma vasta gama de sensores possíveis de fazer essa recolha de dados. Nesta plataforma, estabelece-se a comunicação com os sensores de micro-ondas, de vídeo, sensores implementados nos veículos, sensores que detetam a passagem ilegal de um sinal vermelho, sensores de identificação de matricula ou RFID e através de redes sem fios (Ying & Yang, 2008).

A arquitetura deste sistema é apresentado na figura 2.18. Aqui pode-se verificar que a estrutura do sistema se divide em três partes, a recolha de informação do trânsito, o tratamento dessa informação através de base de dados e serviços e um terminal de navegação que comunica com o utilizador.

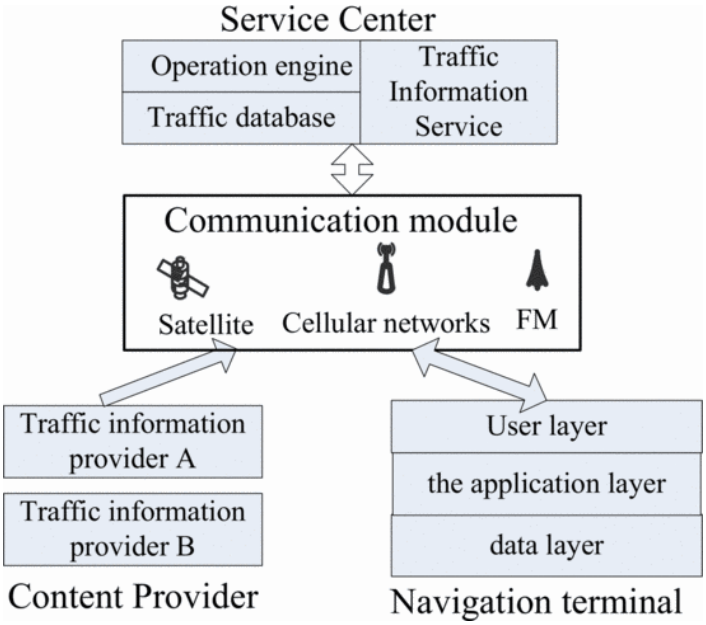

Figura 2.18 - Arquitetura do sistema “VNSRTTI” (Ying & Yang, 2008)

Quando a informação disponível é muito vasta e de fácil acesso, pode-se implementar plataformas com um grau de fiabilidade bastante elevado. No entanto, normalmente essas informações estão restritas às empresas responsáveis pela rede de estradas e não providenciam muita informação. Logo, passando este obstáculo, esta plataforma apresenta métodos fiáveis para este tipo de problemas.

2.2.8. RIMAC

RIMAC é um sistema ainda numa fase piloto, onde se pretende completar o projeto numa aplicação que esteja disponível para todos os utilizadores.

Este sistema foi criado devido ao crescente aumento de veículos na cidade de Lima, o que veio obrigar à necessidade de os condutores terem algo que auxilie a evitar os engarrafamentos e chegar mais rápido aos seus destinos.

Este sistema pretende fazer a recolha de dados em tempo real a partir das câmaras de vídeo presentes nas estradas, no entanto apenas conseguiu fazer essa recolha a partir de um simulador de rede de estradas (Caballero, 2015).

O esquema do sistema é apresentado na figura 2.19. É de salientar como está dividido o sistema, onde apresenta 3 componentes. A recolha de dados feita através das câmaras de vídeo, a interface para os utilizadores fazerem os pedidos de rotas e a base de dados.

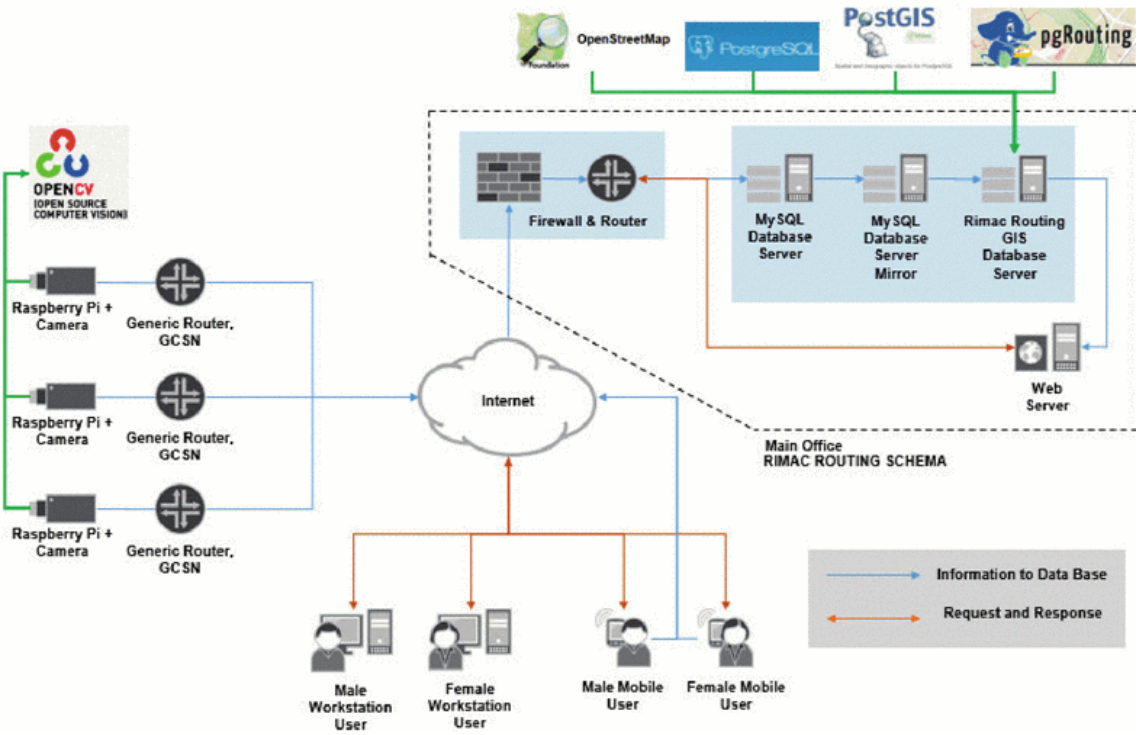

Figura 2.19 - Esquema do sistema RIMAC (Caballero, 2015)

Este sistema foi implementado todo com programas e ferramentas open source e o mapa foi baseado no OpenStreetMap. É possível saber que o algoritmo de otimização usado foi o Dijkstra e os programas usados foram o PostgreSQL, PostGIS e pgRouting, que são bastante referenciados para este tipo de implementações.

Devido às suas implementações open source este sistema é uma boa base para a nossa solução.

2.2.9. Análise e Discussão

Para concluir, é apresentado uma tabela com conclusões sobre algumas das plataformas implementadas no passado. Através da análise da literatura disponível sobre cada plataforma, alguns aspetos sobressaem para realizar uma boa comparação entre elas.

Certamente que a recolha e posterior tratamento dos dados recolhidos têm de ser em quase tempo real, visto ser importante a informação estar sempre atualizada para o caso de algum evento não estar previsto e poder ser prevenido. Outro detalhe que pode ajudar muito quem precise de decidir qual a melhor rota a percorrer, é saber o que ocorreu em outros anos em determinada data, como por exemplo todos os anos ocorre uma maratona e algumas estradas vão estar cortadas. Logo, tendo acesso a essa informação com a realização de uma simulação é algo que pode ajudar bastante na decisão de uma rota.

Por fim, outro aspeto importante é o facto de ser preciso ter mais custos com a instalação de novos sensores, quer sejam espalhados pelos percursos, quer sejam instalados nos veículos. Pois, nem sempre está disponível esse tipo de verbas na implementação de novas plataformas, ou não temos acesso devido à indústria automóvel ser bastante fechada.

Tabela 2.2 - Comparação entre as diferentes plataformas

Plataforma	Tempo Real	Simulação	Comunicação	Novos Sensores
CarWeb	V	X	V	X
DeCloud4SD	V	V	X	V
MVTDC	V	X	X	V
OLSIMv4	V	V	X	X
STA	V	V	V	X
CarTel	V	X	V	V
“VNSRTTI”	V	X	V	X
RIMAC	X	V	V	V

Com isto, podemos analisar na tabela anterior que o requisito mais importante é verificado, ou seja, todas as plataformas analisadas têm como principal foco trabalhar a informação em tempo real, excetuando a RIMAC que ainda não realizou essa implementação, fazendo parte dum trabalho futuro.

A plataforma CarWeb aproveita a tecnologia disponível para realizar a transmissão de dados em tempo real, por isso não precisa de custos adicionais em novos sensores, o que para si já mostra uma grande vantagem. Outra vantagem é vir com uma interface incorporada para fazer a comunicação com o utilizador.

A DeCloud4SD consegue recolher o historial de determinado percurso e fazer uma simulação para prever o que pode ocorrer em determinado dia. Por outro lado, tem a desvantagem de ser preciso implementar novos sensores na rede de estradas.

Numa categoria um pouco diferente temos a MVTDC, que realiza a sua recolha de dados através das câmaras de vídeo. Para além de ser preciso ter muitas câmaras espalhadas pelas

rotas, não se foca na simulação de eventos, apenas tenta contabilizar o número de veículos em determinado percurso, sendo a única comunicação que faz com o utilizador.

A OLSIMv4 é uma plataforma que se foca em juntar um aglomerado de informação sobre o trânsito e realiza simulações para prever o que irá acontecer com o trânsito. Não precisa de implementar nada novo, nem comunica com o utilizador, apenas junta um vasto leque de informação e trabalha-a.

Outra plataforma um pouco diferente das outras é a STA, onde a base de tudo é a comunicação com o utilizador em tempo real. Neste caso, é o utilizador que ajuda a plataforma a recolher os dados de trânsito, informando de acontecimentos que possam dificultar a circulação em determinado percurso. Assim, é possível à plataforma prever o fluxo de trânsito e comunicar ao utilizador. Não precisa de implementar novos sensores, mas sim ter uma boa interface para a comunicação com o utilizador.

A plataforma CarTel tenta fugir à implementação de sensores espalhados pelas rotas e foca-se em coloca-los nos veículos. Não se foca no historial para prever o fluxo de trânsito mas tem uma interface para comunicar com o utilizador.

A “VNSRTTI” foi implementada numa região onde tem disponível bastantes sensores de recolhas de dados, o que por si só já apresenta uma vantagem para não acarretar custos adicionais com novos sensores. Ainda apresenta uma interface para comunicar com o utilizador.

Por fim a RIMAC é um sistema embrionário mas é toda implementada com programas e ferramentas open source. Pretende no futuro fazer a recolha em tempo real através das câmaras de vídeo, mas neste momento apenas realizar simulações. O seu foco é a comunicação com o utilizador e para ter todas as informações sempre corretas precisa de ter muitos sensores de vídeo espalhados pela rede de estradas, o que leva a maiores custos.

Capítulo 3 - Sistema Inteligente para Otimização de Rotas

Na tentativa de responder à pergunta imposta nesta dissertação, criou-se um sistema para ir de encontro à hipótese proposta. Assim, neste capítulo, apresenta-se um sistema proposto pelo autor juntamente com uma extensiva descrição sobre os seus objetivos, os seus pontos fortes e o seu funcionamento.

Este sistema baseia a sua solução num padrão de arquitetura de software denominado MVC, Modelo-Visão-Controlador, que fornece uma maneira de dividir a funcionalidade envolvida na manutenção e apresentação dos dados de um sistema, ou seja, separa a representação da informação da interação com o utilizador.

A arquitetura MVC está dividida em três componentes físicas.

- O modelo, que representa os dados do sistema e as regras. É onde se pode aceder e modificar esses dados, sendo a principal estrutura computacional da arquitetura. Transpondo para a arquitetura do sistema inteligente para otimização de rotas, o modelo equipara-se à base de dados.
- De seguida, tem-se a componente de visão, ou seja, a interface do utilizador. Aqui as informações são apresentadas ao utilizador através de gráficos, textos ou imagens. Esta componente não sabe nada do que se passa nas outras camadas e apenas envia e recebe informações para exibição ao utilizador.
- Por fim, o controlador é onde estão os módulos de toda a arquitetura. Esta camada define o comportamento de todo o sistema e faz a ligação entre o modelo e a componente de visualização. Interpreta as ações do utilizador e atualiza o modelo. Esta comparação está demonstrada na figura 3.1.

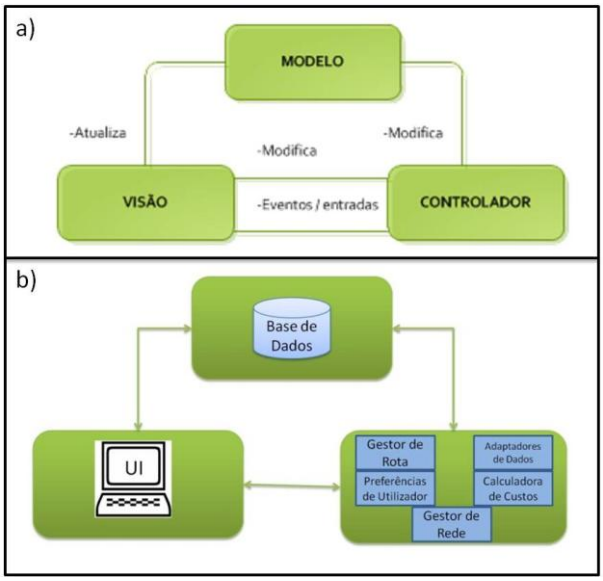

Figura 3.1 - (a) Arquitetura padrão MVC (Modelo-Visão-Controlador) (b) Arquitetura do Sistema Inteligente para Otimização de Rotas

Assim, a solução apresentada tem algumas características que são executadas sequencialmente, a fim de mostrar a rota ideal possível dependendo dos desejos do utilizador, como ilustrado na figura 3.2.

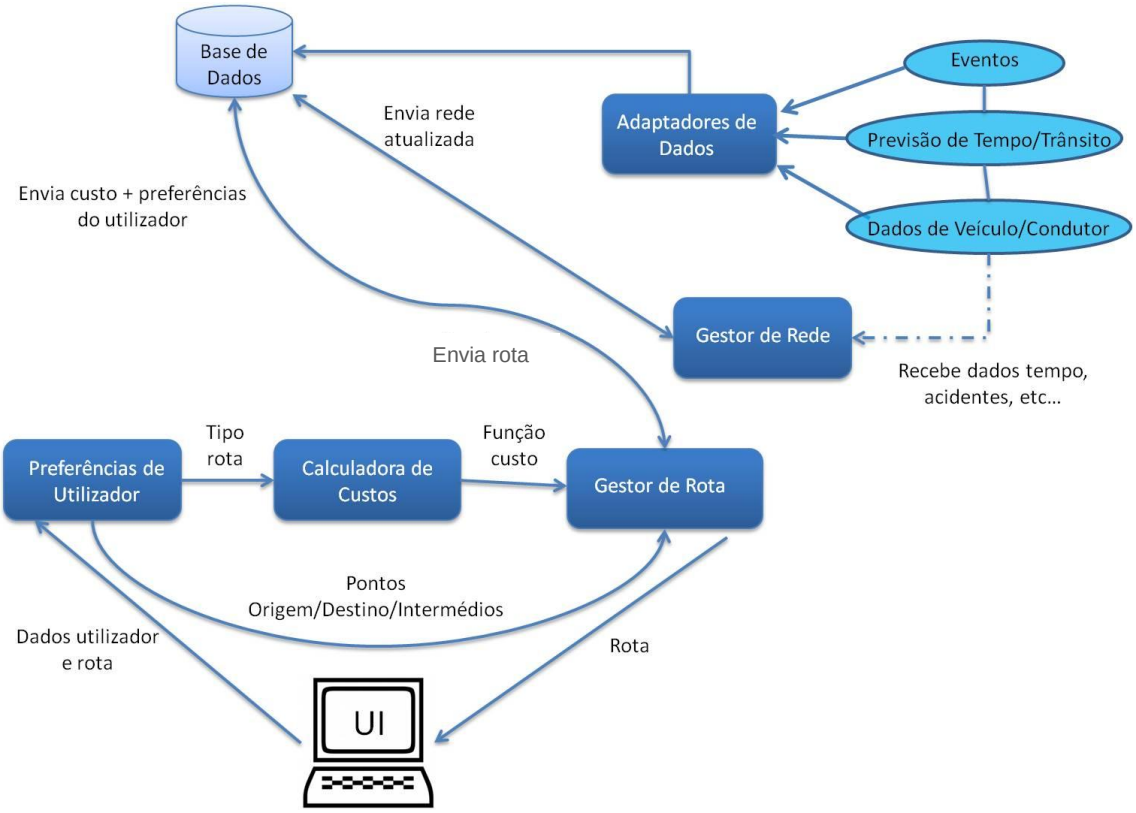

Figura 3.2 - Fluxograma da solução

A partir de uma interface, o utilizador regista as suas preferências para o tipo de rota e quais os pontos relevantes da rota, designando o ponto de partida e de chegada e pode decidir se quer uma rota mais rápida, mais curta ou mais econômica, em termos de consumo de combustível. Além disso, o utilizador pode ainda estabelecer pontos intermédios a visitar, caso seja necessário fazer paragens para outras entregas. Quando todos os parâmetros são inseridos, o sistema analisa potenciais eventos que possam fazer com que essa rota específica não seja ideal, evitando que se fique preso no trânsito ou que seja preciso circular a velocidades mais baixas devido ao fluxo de tráfego elevado.

Finalmente, o sistema usa um algoritmo para otimizar rotas, com o objetivo de traçar a rota ideal para o utilizador.

3.1. Preferências do Utilizador

Este módulo recebe os dados da interface referentes às preferências do utilizador, nomeadamente qual o tipo de rota que o utilizador prefere (curta, rápida ou econômica) e as coordenadas dos pontos importantes para estabelecer uma rota. De seguida, é feita a comunicação à Calculadora de Custos para determinar qual o custo que prevalecerá e será trabalhado para definir a rota.

3.2. Calculadora de Custos

Este módulo recebe os dados provenientes da Preferências do Utilizador e vai-se focar nos dados do tipo de rota para estabelecer os custos a ser trabalhados para definir a rota.

No caso de o utilizador preferir uma rota mais curta, mais rápida ou mais econômica, são estabelecidos custos, esses custos são a velocidade, a distância e os consumos dos carros e preço do combustível. Estes custos têm pesos diferentes na determinação da rota desejada consoante as preferências do utilizador. Por exemplo, um veículo pode circular numa estrada a 120 km/h, mas infelizmente ocorre um acidente originando com que haja o corte da estrada, logo o parâmetro de velocidade para essa estrada passa para 0 km/h ou para um valor muito baixo e faz com que a estrada seja descartada e o sistema tenta arranjar outra rota para contornar o acidente, caso o desejo do utilizador seja a rota mais rápida e conforme está ilustrado na figura 3.3. Assim, este módulo irá comunicar ao Gestor da Rota que o custo da velocidade será o foco para o cálculo da rota ideal e faz com que a rota onde ocorreu o acidente seja descartada. No entanto, se a rota pedida for a mais curta, o sistema não descarta a rota com o acidente, pois continua a ser a rota ideal.

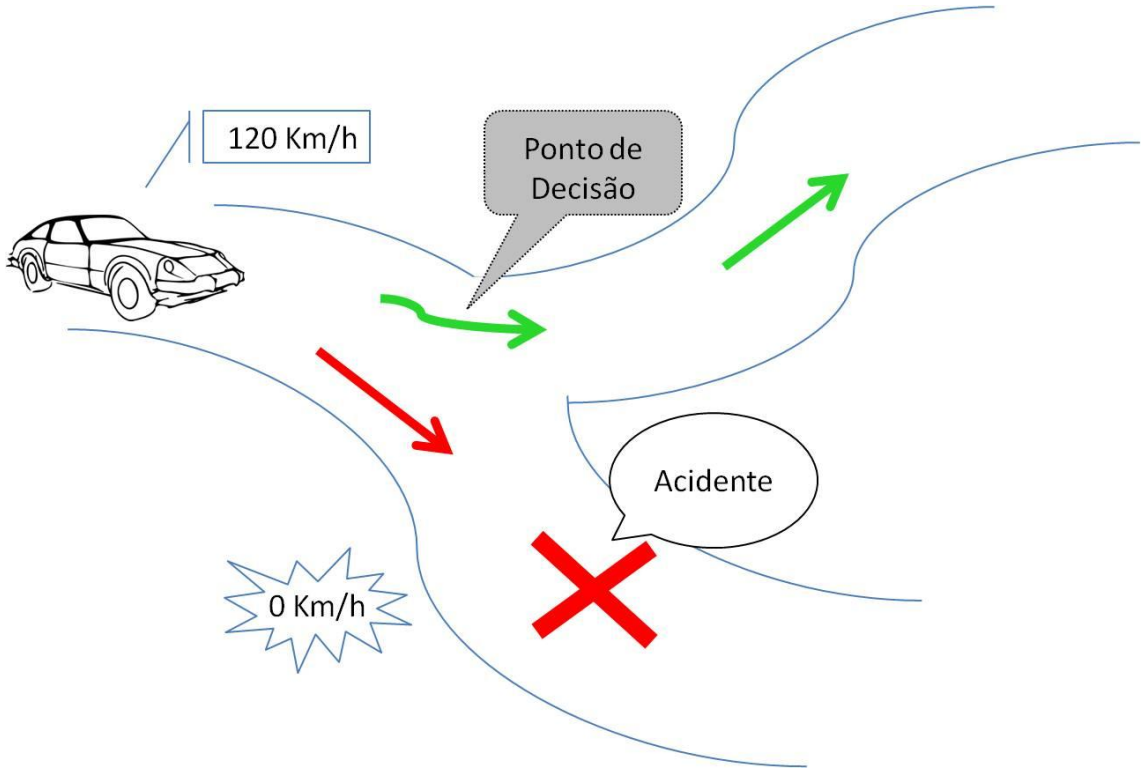

Figura 3.3 – Exemplo de um evento a ocorrer

Sempre que haja vários percursos em que ocorra o corte ao trânsito, a solução encontra quase sempre uma rota alternativa, nem que o único percurso disponível seja a rota mais longa.

3.3. Gestor de Rota

Este módulo trabalha com toda a informação necessária para traçar uma rota ideal consoante as definições do utilizador. Aqui são recolhidas as preferências do utilizador relativamente aos pontos de partida e chegada e pontos intermédios. Também recebe toda a informação do módulo anterior, onde é definido o tipo de rota e a função custo. Esta informação toda é transferida para a base de dados, onde irá ser analisada e trabalhada para definir a rota desejada com todas as preferências do utilizador.

Em sentido inverso e depois do algoritmo Dijkstra otimizar a rota ideal, a base de dados envia todos os nós e estradas para traçar a rota desejada. Esta informação de rota irá ser encaminhada para a Interface do Utilizador para ser visível a ilustração da rota ao utilizador, como podemos ver na Figura 3.4.

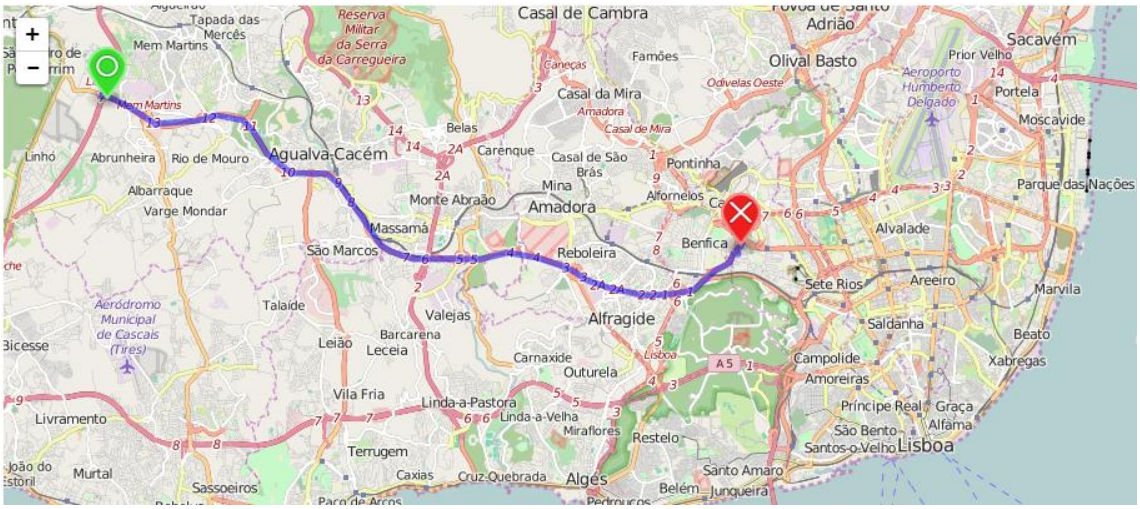

Figura 3.4 - Esboço de uma rota

Cada rota é definida por inúmeros nós, ou seja, cada estrada ou troço tem muitos nós. Assim, a informação que será enviada da base de dados será todos os nós constituintes da rota.

3.4. Adaptadores de Dados

Nem toda a informação relevante para definir uma rota vem do Gestor de Rota, ou seja, é necessário uma recolha de dados sobre alguns aspetos que são importantes na definição de uma rota. Esta recolha de dados pode ser feita previamente ou em alguns casos pode ser em quase tempo real (Agostinho & Jardim-goncalves, 2015). Nesta solução dá-se ênfase a alguns dados a ser recolhidos que podem ser fulcrais, nomeadamente eventos que possam originar algum tráfego nas estradas, como por exemplo acidentes ou outro evento que determine o corte das estradas, tal como manifestações, uma maratona, entre outros.

Outro aspeto importante prende-se com o facto de os dados sobre o tempo e o comportamento do tráfego em determinados dias, poderem não levar ao corte das estradas mas que seja necessário reduzir a velocidade, levando a uma maior demora no trajeto. Por fim, a recolha de dados sobre o veículo e o condutor também pode levar a um ajustamento na decisão da rota, nomeadamente nos seus comportamentos, ou seja, caso um veículo gaste mais ou menos combustível e se um condutor tem uma condução mais leve ou mais agressiva.

Estes dados auxiliares na definição de rotas podem ser adquiridos através de web services ou de API's que disponham dessa informação. Neste módulo, esta informação é recolhida e transformada em formatos que possam ser percetíveis e trabalhados na solução. Ou seja, o utilizador ou administrador pode configurar este módulo para determinados tipos de dados serem percetíveis à solução e que a informação possa chegar corretamente à Base de Dados. Por exemplo, um web service sobre a previsão do tempo envia os dados na escala Fahrenheit,

mas o sistema só reconhece na escala Celsius, logo este módulo faz essa conversão e assim a informação já será reconhecida corretamente.

3.5. Gestor de Rede

Este módulo é responsável por quase todas as interações feitas nos módulos em geral. É aqui que são realizadas as perguntas à Base de Dados, tanto para aproximar os pontos de partida, chegada e intermédios que sejam clicados muito longe de estradas de nós que sejam reconhecidos no mapa, como para criar a coluna dos custos enviados pelo Adaptador de Dados.

A maioria das funções implementadas no sistema estão aqui contempladas, para uma melhor organização sequencial da solução, fazendo com que não haja troca de dados e envio errado de informação de módulo para módulo.

Por fim, realiza-se a pergunta responsável pelo pedido de uma nova rota à Base de Dados e quando estiver definida ativa-se a função de traçar a rota na Interface do Utilizador.

3.6. Base de Dados

O módulo da Base de Dados contem a maior parte da informação importante para a definição das rotas. É aqui que se encontram todos os detalhes do mapa de cada país, tal como as velocidades e distâncias das estradas e os nós pertencentes. Sempre que necessário, é aqui que são inseridas todas as alterações, através de uma coluna nova, que é preciso fazer caso se detete algum evento que possa originar trânsito, como é o caso da figura 3.3, onde o máximo de velocidade na rota é 120 km/h, mas devido ao acidente a velocidade desce para os 0 km/h, logo é um custo que precisa de ser alterado. Neste caso, o módulo Gestor de Rede envia os custos alterados que terão maior peso e é criada uma coluna nova com a informação desses custos, que só é ativada caso haja um acidente nessa estrada.

A base de dados tem a possibilidade de se agregar a outras ferramentas, como é o caso do pgRouting que disponibiliza algoritmos para o cálculo de rotas. Logo, este módulo vem contemplado com o algoritmo que auxilia na definição da rota ideal. Na solução foi usado o algoritmo de otimização Dijkstra.

Sempre que é solicitado um pedido de nova rota, o algoritmo Dijkstra calcula essa rota e envia a informação para o Gestor de Rota. Não são guardados dados sobre rotas que tenham sido calculadas anteriormente, ou seja, está constantemente a ser consultada para calcular a nova rota. A única informação que este módulo guarda é os custos que vêm do Adaptadores de

Dados, relativos aos eventos, previsão de tempo e trânsito e dados sobre o veículo ou condutor.

3.7. Interface do Utilizador

Todos os programas, plataformas e sistemas precisam de uma interface para realizar os pedidos de quem o vai utilizar. Este é o objetivo deste módulo, interligar os pedidos e as preferências do utilizador na escolha de uma rota com o restante sistema, como ilustrado na figura 3.5.

Figura 3.5 - Interface que comunica com o utilizador

Neste módulo, o utilizador, recorre ao mapa disponível, onde terá que clicar para mostrar as coordenadas do ponto de partida e de chegada, para o qual realizará o trajeto. Uma característica importante e que merece algum relevo é o facto de caso ser preciso realizar algumas paragens intermédias, o utilizador pode clicar nesses destinos para a rota ser definida de forma a passar por essas paragens. Existe uma exceção que é caso o utilizador tenha

clicado num sítio muito afastado de estradas, assim o sistema não irá conseguir calcular a rota desejada, mas isso é algo que não depende da solução implementada e caso isso aconteça o utilizador é notificado com essa informação e terá que escolher outro sítio.

Como todas as rotas vão de encontro às preferências do utilizador, é ainda possível neste módulo escolher o tipo de rota a realizar, ou seja, se quer uma rota mais rápida, mais curta ou mais econômica, em termos de consumo de combustível, como podemos ver na parte de baixo da figura 3.5.

Em sentido inverso, a interface recebe a informação do traçado da rota, que vai ser ilustrada no mapa disponível.

Capítulo 4 - Testes e Validação

4.1. Metodologia de Testes

Localizar os erros numa implementação de sistema usando a experimentação também é conhecido como o processo de testes. Este processo pode ser realizado em um ambiente especial, onde o uso normal e excecional do sistema pode ser simulado. No entanto, o teste mostra ao autor do sistema a presença de erros e não a sua ausência, não garantindo a correção completa de uma implementação (Tretmans, 2001).

De acordo com cada campo específico do aplicativo, existem métodos diferentes para testar a adequação das soluções para satisfazer as suas necessidades (Onofre, 2007), fazendo uso do padrão internacional para testes de conformidade de sistemas abertos.

Uma aproximação padrão é usada e ilustrada na figura 4.1. Para testar a especificação da hipótese, deve ser definido um conjunto de testes, mas para eles para ser executada a hipótese deve ser implementada como uma prova de conceito. Esta implementação não precisa ser completamente funcional, mas terá de cumprir os requisitos definidos na arquitetura proposta. Os resultados da execução do teste são observados, levando a um veredito sobre a implementação do sistema com os requisitos iniciais definidos (Tretmans, 2001).

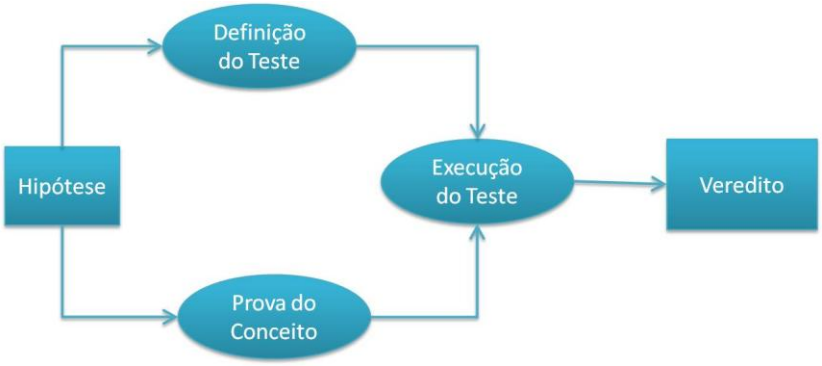

Figura 4.1 – Visão global do processo de testes

Os testes abstratos devem ser especificados com uma notação bem definidos, independente de qualquer implementação. A notação usada é a TTCN-2 onde o comportamento interno do sistema não é relevante, sendo a sequência de eventos determinado através da observação e revelado durante os testes que é o núcleo do conceito a que se refere (Tretmans, 1992).

O TTCN-2 é apresentado em forma de tabela, que mostra as várias partes que define o teste, como uma sequência de eventos sucessivos, um veredicto e um cabeçalho. Cada tabela

possui um cabeçalho, onde está o nome de teste, qual a finalidade e as entradas e saídas são demonstradas. A sequência de eventos sucessivos é indicada pelo aumento da indentação do mesmo e é identificada por um número de linha.

Os eventos que compõem a sequência estão divididos em dois tipos: ações e perguntas. As ações são representadas com um ponto de exclamação ("!") no início do evento, + + define a interação com o sistema. As perguntas, que são representadas com um ponto de interrogação (“?”), definem as respostas esperadas do sistema. A sequência termina com a especificação do veredito que é atribuído quando termina a execução do teste.

O veredito pode produzir três resultados: "Sucesso", "Falha" e "Inconclusivo". Sucesso indica que o teste foi executado com êxito e com o resultado esperado. Falha indica que a implementação não está de acordo com a especificação e Inconclusivo indica que o serviço não está disponível, logo a execução não pode ser testada.

Um exemplo de um teste baseado no TTCN-2 é mostrado na tabela 5.1. Ela exemplifica uma invocação de uma transformação do modelo. O teste apresentado começa com a invocação de uma transformação do modelo para o mecanismo de execução. O próximo passo é verificar se há alguns dados retornados a partir do modelo de execução de transformação. Se houver uma saída de transformação do modelo e o resultado esperado é igual, o veredito do teste é "Sucesso", caso contrário se a saída da transformação não é igual à produção esperada então o veredito do teste é "Falha". No caso de a convocação ser feita e não houver nenhuma saída do teste o resultado é "INCONCLUSIVO", uma vez que existem várias razões para esse comportamento.

Tabela 4.1 - Exemplo de teste TTCN em tabela

Multiplicação de 2 Serviços de Invocação		
Nome do Teste:	Teste de Multiplicação por 2 Serviços	
Propósito:	Confirma se o serviço está disponível e fornece o resultado da operação	
Entradas:	I1: Número a ser multiplicado, I2: Resultado Esperado	
Saídas:	O1: Resultado da multiplicação por 2 da entrada	
Número Linha	**Comportamento**	**Veredito**
1	! Invoca o Serviço com os parâmetros (I1)	
2	? Retorna dados como Resultado do Serviço (O1) assim Serviço Disponível	
3	? Resultado (O1) é igual ao Esperado (I2)	SUCESSO
4	?Resultado (O1) é diferente ao Esperado (I2)	FALHA
5	? Nenhum Resultado do Serviço	INCONCLUSIVO

A fim de apresentar um exemplo de caso de teste TTCN-2 com base em tabela definido na tabela 4.1, foi criado a tabela 4.2. Os parâmetros da tabela são as entradas para o teste, os resultados esperados e os resultados obtidos durante o teste. Cada linha da tabela 5.2 representa um caso de teste específico. Com esta abordagem mais do que um caso de teste pode ser representado para cada teste abstrato.

Tabela 4.2 - Exemplo de um caso de teste

Teste	Entrada		Saída	Resultado (Número de Linha)	
	I1: Número	I2: Resultado Esperado	O1: Resultado	Esperado	Atual
1	60	120	120	(3)	(3)
2	8	16	120	(3)	(4)
3	0	0	Sem Dados	(3)	(5)

4.2. Prova de Conceito da Implementação

Para poder aplicar a abordagem descrita, várias tecnologias e ferramentas foram usadas nos módulos da solução que ajudam a definir uma rota. Como se pode ver na figura 4.2 e explicado de seguida.

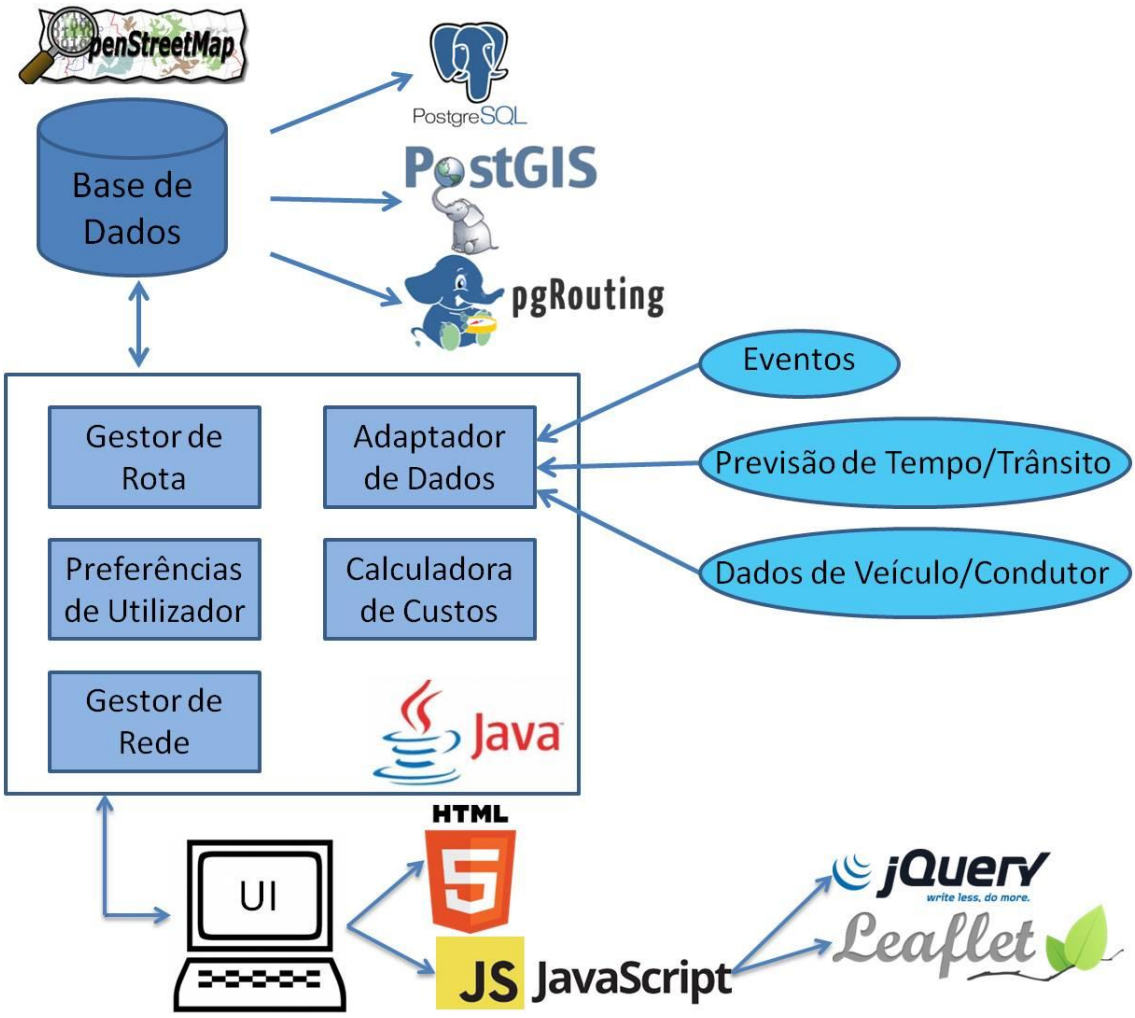

Figura 4.2 - Arquitectura do sistema com as tecnologias e ferramentas usadas

O módulo da Base de Dados consiste na tecnologia PostgreSQL[1], que é um sistema de base de dados objeto-relacional de open source. Este módulo é composto ainda pelas ferramentas de PostGIS[2], que são responsáveis por armazenar na base de dados um sistema de informação geográfica e onde se realiza queries para recolher a informação que está disponível nos mapas dos países disponíveis no OpenStreetMap[3]. Finalmente, é composto pelo pgRouting[4] que é uma ferramenta open source que auxilia o PostGIS e a base de dados PostgreSQL para fornecer funcionalidades de roteamento geoespacial. O pgRouting oferece ainda alguns algoritmos de otimização para calcular as rotas. Estas tecnologias e ferramentas utilizadas na base de dados têm bastantes semelhanças e vão de encontro às especificações do nosso problema. Além disso, são usadas como um módulo, o que facilita a tarefa de implementar na nossa solução e é a razão pela qual é usada.

Todos os módulos responsáveis pela manipulação das preferências do utilizador e da recolha de dados é feita na linguagem de programação orientada a objetos Java[5] e o programa Eclipse[6]. Aqui estão contemplados os módulos, Preferências do utilizador, Calculadora de Custos, Gestor de Rota, Adaptadores de Dados e Gestor de Rede.

Finalmente, na Interface do Utilizador são usados o HTML5, uma linguagem para a estruturação e apresentação de conteúdos e JavaScript[7], uma linguagem de programação para navegadores da web. É nesta última linguagem que recorre-se às bibliotecas jQuery[8] para simplificar os scripts que interagem com o utilizador e o Leaflet[9], que é usado para construir aplicações de mapas na web.

Neste momento há um grande leque de tecnologias e ferramentas para se poder usar neste tipo de sistemas, mas muitas têm os direitos de autores e não disponibilizam a informação ou o código necessário para poder usar e trabalhar com elas. Na solução as tecnologias e ferramentas são open source, o que torna bastante fácil a sua implementação.

4.3. Definição do Teste

Ao longo desta secção será apresentado tabelas TTCN que formalizam os testes executados para validar a implementação do sistema proposto. Estes testes vão ao encontro de verificar se a informação está correta para determinar uma rota. Visto o sistema ser composto por vários

[1] “PostgreSQL Global Development Group,” 2016
[2] “PostGIS Project Steering Committee, ‘PostGIS,’” 2016
[3] “WordPress, ‘OpenStreetMap,’” 2015
[4] “pgRouting Community, ‘pgRouting Project,’” 2016
[5] “Oracle Corporation, ‘Java,’” 2010
[6] “The Eclipse Foundation, ‘Eclipse,’” 2016
[7] “JavaScript.com, ‘JavaScript,’” 2016
[8] “The jQuery Foundation, ‘jQuery,’” 2016
[9] “GitHub, Inc., ‘Leaflet,’” 2015

módulos e a informação ter que seguir uma determinada sequência, é bastante possível que possa haver perda total ou parcial dessa informação. Assim, serão testados alguns cálculos de rotas com diferentes variantes, para validar a implementação do sistema.

- Teste de Cálculo de uma Rota

Na tabela 4.3 está um caso de teste para verificar se o sistema fornece rotas, com as preferências do utilizador, que é a característica principal do sistema.

Tabela 4.3 – Teste de Cálculo de uma Rota

Calcular uma Rota		
Nome do Teste:	Teste de Cálculo de um Rota	
Propósito:	Confirma se há rota disponível e fornece a rota	
Entradas:	I1: Preferências do Utilizador, I2: Resultado Esperado	
Saídas:	O1: Resultado do cálculo da rota	
Número Linha	**Comportamento**	**Veredito**
1	! Invoca o Serviço com os parâmetros (I1)	
2	? Retorna dados como Resultado do Serviço (O1) assim Serviço Disponível	
3	? Resultado (O1) é igual ao Esperado (I2)	SUCESSO
4	?Resultado (O1) é diferente ao Esperado (I2)	FALHA
5	? Nenhum Resultado do Serviço	INCONCLUSIVO

Este caso de teste foi executado com sucesso e a maioria das rotas foram validadas. Mesmo assim, ainda é possível que o caso de teste apresente alguns resultados inesperados, devido ao facto de o mapa não reconhecer determinados pontos, mas em nada influencia a implementação do sistema.

- Teste de Cálculo de uma Rota com um Ponto Intermédio

Na tabela 4.4 será realizado o teste de caso para verificar se o cálculo de uma rota é realizado contendo o ponto intermédio inserido pelo utilizador. Neste caso, os resultados inesperados que podem ocorrer no teste é a rota estar a ser calculada normalmente sem o ponto intermédio inserido. Também é possível que aconteça que o ponto intermédio seja reconhecido como um ponto de chegada e a rota apenas é calculada do ponto de partida até ao ponto intermédio.

Tabela 4.4 – Teste de Cálculo de uma Rota com um Ponto Intermédio

Calcular uma Rota com um Ponto Intermédio		
Nome do Teste:	Teste de Cálculo de um Rota com um Ponto Intermédio	
Propósito:	Confirma se há rota disponível e fornece a rota	
Entradas:	I1: Preferências do Utilizador, I2: Resultado Esperado	
Saídas:	O1: Resultado do cálculo da rota	
Número Linha	**Comportamento**	**Veredito**
1	! Invoca o Serviço com os parâmetros (I1)	
2	? Retorna dados como Resultado do Serviço (O1) assim Serviço Disponível	
3	? Resultado (O1) é igual ao Esperado (I2)	SUCESSO
4	?Resultado (O1) é diferente ao Esperado (I2)	FALHA
5	? Nenhum Resultado do Serviço	INCONCLUSIVO

Também neste caso de teste os resultados foram adquiridos com sucesso, sendo que as rotas estão a ser calculadas contendo o ponto intermédio inserido.

- Teste de Cálculo de uma Rota com vários Pontos Intermédios

Visto ser importante o teste de caso com um ponto intermédio, também convém analisar se a solução está pronta para fazer o cálculo das rotas com vários pontos intermédios, assim é realizado o caso de teste ilustrado na tabela 4.5.

Tabela 4.5 – Teste de Cálculo de uma Rota com vários Pontos Intermédios

Calcular uma Rota com vários Pontos Intermédios		
Nome do Teste:	Teste de Cálculo de um Rota com vários Pontos Intermédios	
Propósito:	Confirma se há rota disponível e fornece a rota	
Entradas:	I1: Preferências do Utilizador, I2: Resultado Esperado	
Saídas:	O1: Resultado do cálculo da rota	
Número Linha	**Comportamento**	**Veredito**
1	! Invoca o Serviço com os parâmetros (I1)	
2	? Retorna dados como Resultado do Serviço (O1) assim Serviço Disponível	
3	? Resultado (O1) é igual ao Esperado (I2)	SUCESSO
4	?Resultado (O1) é diferente ao Esperado (I2)	FALHA
5	? Nenhum Resultado do Serviço	INCONCLUSIVO

Mais uma vez o teste de caso foi adquirido com sucesso, mostrando que a solução está pronta para receber os pontos intermédios que o utilizador desejar. Convém realçar que a solução segue a ordem que o utilizador inseriu os pontos intermédios.

- Teste de Cálculo de uma Rota com um Evento de Acidente

Por outro lado, temos os eventos de acidentes. Neste caso, já não convém que estes pontos venham contemplados nas rotas mas sim que se evitem rotas que tenham estes eventos. Logo, é importante verificar se o cálculo da rota engloba estes pontos ou se evita no cálculo da rota.

Tabela 4.6 – Teste de Cálculo de uma Rota com um Evento de Acidente

Calcular uma Rota com um Evento de Acidente		
Nome do Teste:	Teste de Cálculo de um Rota com um Evento de Acidente	
Propósito:	Confirma se há rota disponível e fornece a rota	
Entradas:	I1: Preferências do Utilizador, I2: Resultado Esperado	
Saídas:	O1: Resultado do cálculo da rota	
Número Linha	**Comportamento**	**Veredito**
1	! Invoca o Serviço com os parâmetros (I1)	
2	? Retorna dados como Resultado do Serviço (O1) assim Serviço Disponível	
3	? Resultado (O1) é igual ao Esperado (I2)	SUCESSO
4	?Resultado (O1) é diferente ao Esperado (I2)	FALHA
5	? Nenhum Resultado do Serviço	INCONCLUSIVO

Nos casos de testes anteriores foi verificado se os pontos intermédios vinham contemplados no cálculo das rotas. Agora é verificado com sucesso que o cálculo da rota evita as estradas com os eventos de acidentes.

4.4. Veredito

Através dos testes efetuados é possível tirar algumas conclusões. A primeira conclusão é que a prova de conceito implementada passou nos testes com sucesso. Todos os testes de casos estão com resultados positivos. Ou seja, a informação dentro do sistema está a ser enviada corretamente. Pois, os pontos intermédios e os eventos de acidentes não se tão a confundir e o cálculo das rotas é realizado com sucesso. Assim, o utilizador está apto a receber uma rota com as suas preferências e desejos.

Tabela 4.7 - Percentagem de Sucesso dos Testes

Caso de Teste	Número de Testes		Terminados com Resultado Esperado	Percentagem de Sucesso
	Realizados	Incompletos		
1	50	3	42	84%
2	50	0	46	92%
3	50	0	46	92%
4	50	2	45	90%

Como é visível na tabela 4.7, todos os testes realizados obtiveram uma taxa de sucesso acima dos 80%. No entanto, alguns testes tiveram um resultado de Incompleto. Este fato acontece, por exemplo no caso de teste 1, por pontos de partida ou chegada não se encontrarem perto de estradas reconhecidas nos mapas. Ou então, como no caso de teste 4, o cálculo da rota é feito antes do acidente terminar e a resposta é dada ao utilizador depois do acidente terminar.

Concluindo, os testes definidos e executados provam que a hipótese formulada no início da dissertação é uma hipótese válida e a arquitetura do sistema criada é capaz de lidar com os problemas impostos.

4.5. Cenário de Validação Industrial

O piloto industrial de C2NET Metalworking Networked para PME (pequenas e médias empresas) será usado como cenário de validação industrial deste trabalho. O objetivo do projeto C2NET é acrescentar valor na melhoria da competitividade das PMEs em geral, na inovação e adaptabilidade no caso de um consórcio de empresas em rede. Proporcionando intercâmbios entre países e interempresas, na construção de empresas em rede que são suportados por esquemas de relações estáveis e de modernas cooperações e de coordenações paradigmas de negócios. Isto irá conceder vantagens aos consumidores finais, principalmente em termos de redução dos produtos a serem lançados no mercado e dos seus custos.

A rede é composta por duas PME que trabalham nas indústrias transformadoras em que compram aço a fornecedores e transformam em produtos finais, prontos para vender no mercado. Uma das especializações é na produção de tubos de aço, e a outra é a experiência que está relacionada com a criação de prateleiras de aço. Ambas as empresas querem reduzir os custos de aquisição de matérias-primas dos fornecedores de aço, tanto em termos de material como no transporte do mesmo.

Ambas as empresas utilizam a metodologia de trabalho *Make-to-order*[10], onde apenas produzem quando têm encomendas por parte dos clientes. Isto faz com que as mesmas não tenham interesse em possuir *stock* de matéria-prima, uma vez que ter material em armazém parado, traduz-se num grande investimento inicial, incomportável por uma PME sem garantias que o mesmo irá ser utilizado. Para tal, a cada nova encomenda de clientes têm de reavaliar também elas uma encomenda ao fornecedor (Figueiras, P., Melo, R., Costa, R., Agostinho, C., Lima, C., & Jardim-Goncalves, 2015).

O tempo e custo de produção nestas PMEs é também elevado, a cada nova produção é necessário reconfigurar toda a maquinaria. Logo, as empresas têm de planear cuidadosamente a maneira de obter matéria-prima para poder começar a produzir. Atrasos inesperados nesta aquisição levam a aumentos nos custos de produção, como é ilustrado na figura 4.3 (Ferreira et al., 2016).

Um exemplo concreto é o fato de o transporte da matéria-prima estar programado para determinado dia e chegar a determinada hora, mas devido a algum evento, por exemplo um grande acidente ou algumas estradas cortadas devido ao mau tempo, esse transporte terá que fazer um grande desvio e vai acrescentar mais um dia ao que estava estipulado, logo toda a produção irá estar comprometida e será necessário fazer os devidos ajustamentos para não haver mais atrasos. Assim, o planeamento é muito importante com recurso à solução implementada.

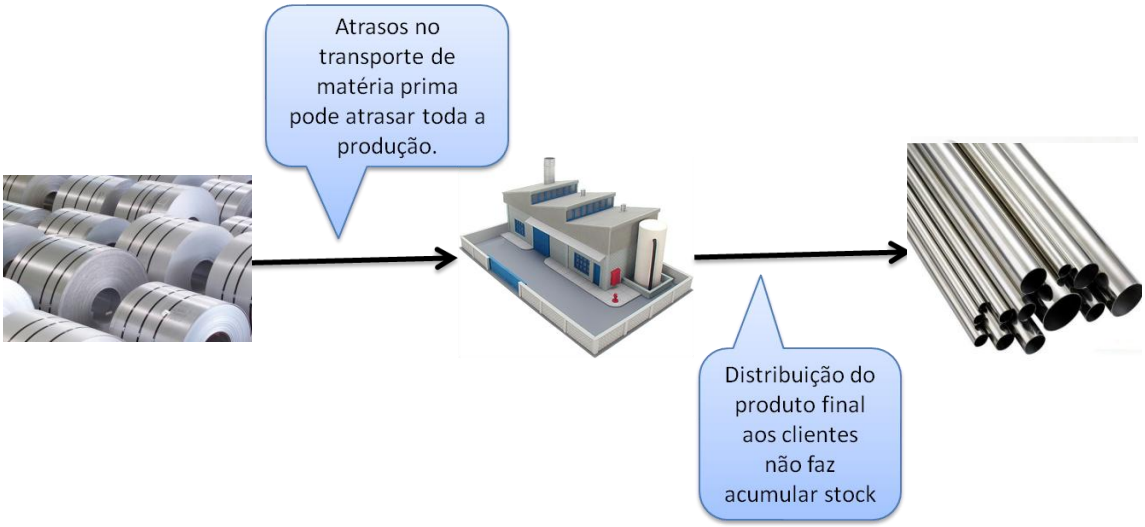

Figura 4.3 - Objetivo do Sistema no Cenário Industrial

Em conformidade com este estudo e as exigências expostas, para realizarem a contratação do transporte, a plataforma sugere a melhor rota de entrega dos produtos consoante a localização das empresas e para as datas pretendidas.

[10] Processo de manufatura, em que os bens são produzidos depois de receber a encomenda.

Capítulo 5 - Conclusões e Trabalho Futuro

Existem muitos programas e sistemas para definir rotas, consoante os desejos do utilizador, no entanto, esses serviços têm muitas restrições e não é facultado muita informação sobre as suas funções e a recolha de dados efetuada.

Dada a diversidade e a complexidade de definir uma rota ideal e posteriormente obter a otimização perfeita da mesma, existe uma variedade de métodos para resolver esse problema, denominados algoritmos de otimização. Estes algoritmos podem-se dividir em três grandes grupos.

- Métodos exatos, que normalmente encontram a solução ideal mas demoram muito tempo a realizar essa operação.
- Heurísticas, que são algoritmos que encontram a solução ideal ou perto dela e realizam a procura pela rota ideal um pouco através pela intuição.
- E por fim, as Metaheurísticas que tentam evitar os erros dos dois grupos anteriores e encontra a solução num tempo razoável.

Para uma melhor fiabilidade na determinação da rota ideal, é de salientar também a informação prévia sobre o comportamento do trânsito e se ocorre determinado evento que possa levar ao corte das estradas. Assim, uma recolha de dados em tempo real sobre o fluxo do tráfego ajuda a determinar com maior exatidão uma rota ideal.

Por isso, o objetivo deste trabalho passou por criar uma solução computacional que auxilie na decisão de definir uma boa rota para realizar uma deslocação. Esta solução apresenta flexibilidade não só a definir a rota como a recolher dados em tempo real do fluxo de trânsito e de ocorrência de eventos que possam de certa forma alterar o fluxo normal de tráfego. Como é referido neste trabalho, foi realizada uma análise em tempo real de eventos, tais como acidentes ou cortes de estradas, para evitar rotas que apresentem um acréscimo de fluxo de veículos originando demoras na realização das rotas. Esta análise também está preparada para analisar as condições meteorológicas e o comportamento do trânsito, para evitar rotas com um deslocamento mais lento. No final, foi realizada uma pesquisa a partir de um algoritmo de otimização de rotas, algoritmo Dijkstra, para definir qual a rota ideal consoante as preferências do utilizador.

Alguns aspetos fortes da solução implementada é a inserção dos pontos intermédios, pois muitas soluções do nosso dia-a-dia têm mas não estão devidamente a funcionar ou ainda estão numa fase inicial. Em relação à recolha de dados em tempo real, a solução foi implementada para receber uma grande variedade de informação. Parte dessa informação já se encontra a funcionar mas outra apenas está preparada para receber e não foi testada.

A solução apresentada só foi possível ser executada devido aos inúmeros programas, tecnologias e ferramentas disponíveis em open source. Certamente são utensílios de grande ajuda e que facilitam não só o trabalho realizado no desenvolvimento da solução, como futuramente pode ajudar outras pessoas que queiram iniciar por esta área ou mesmo melhorar programas que tenham sido desenvolvidos.

O cenário de validação industrial forneceu resultados satisfatórios como esperado ao longo do desenvolvimento da solução proposta. Na verdade, pode-se concluir que o trabalho desenvolvido no decorrer desta dissertação apresentou um resultado positivo. O problema inicial com todas as suas características foi resolvido com sucesso, e foi provado que a hipótese proposta é correta.

Apesar dos desafios que a recolha de dados em tempo real apresenta, como trabalho futuro, é possível aumentar a quantidade de dados para recolha e verificar melhor a sua veracidade se houver parcerias com as entidades responsáveis por esse departamento. A disponibilidade da maioria desses dados facilitaria uma melhor implementação da solução, apresentando resultados mais credíveis.

Outro melhoramento possível deve-se à performance da solução, pois é possível diminuir os tempos de cálculos de rotas e de recolha de dados, fazendo com que os resultados sejam instantâneos. Por fim, um aspeto que podia ser interessante abordar é o facto de em vez de inserir as preferências do utilizador manualmente, seria inserir verbalmente. Já existe algumas tecnologias sobre o assunto e seria bastante inovador neste tipo de sistemas.

A solução computacional apresentada neste trabalho está num estado embrionário, no entanto, realiza a maioria das funcionalidades com resultados credíveis, levando o seu autor a concluir que é possível realizar uma solução funcional e completa.

Capítulo 6 - Referências

Agostinho, C., & Jardim-goncalves, R. (2015). Annual Reviews in Control Sustaining interoperability of networked liquid-sensing enterprises : A complex systems perspective. *Annual Reviews in Control*, *39*, 128–143. http://doi.org/10.1016/j.arcontrol.2015.03.012

Benslimane, M. T. (2014). Exact method for the multi-region vehicle routing problem in large quantities by a heterogeneous fleet of vehicles. *2014 International Conference on Logistics Operations Management*.

Botassoli, G. T., Alberti, R. A., & Furtado, J. C. (2015). Simulação computacional para otimização de filas em processos. *GEINTEC-Gestão, Inovação E Tecnologias*, *5*, 2121–2135.

Botinˇ, M. (2006). Anomalies in Distributed Branch-and-Cut Solving of the Capacitated Vehicle Routing Problem. *28th International Conference on Information Technology Interfaces*, 677–682.

Caballero, L. C. (2015). RIMAC Project : Open urban routing information system fed by real time reliable sources . *Smart Cities Conference (ISC2), 2015 IEEE First International. IEEE*.

Camarinha-matos, L. M., & Terminology, B. (2016). SCIENTIFIC RESEARCH Unit 2 : SCIENTIFIC METHOD, 2009–2016.

César, H., Oliveira, B. De, Vasconcelos, G. C., & Bastos, G. (2006). A Multi-Start Simulated Annealing Algorithm for the Vehicle Routing Problem with Time Windows. *2006 Ninth Brazilian Symposium on Neural Networks (SBRN'06). IEEE*, 1–6.

Chao, Y. (2010). A Developed Dijkstra Algorithm and Simulation of Urban Path Search. *Computer Science and Education (ICCSE), 2010 5th International Conference On. IEEE*, 1164–1167.

Charbonneau, N., & Vokkarane, V. M. (2010). Tabu Search Meta-Heuristic for Static Manycast Routing and Wavelength Assignment over Wavelength-Routed Optical WDM Networks. *Communications (ICC), 2010 IEEE International Conference On. IEEE*.

Feo, T. A., & Resende, M. G. C. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization 6.2*, 109–133.

Ferreira, J., Sarraipa, J., Ferro-beca, M., Agostinho, C., Jardim-goncalves, R., & Costa, R. (2016). End-to-end manufacturing in factories of the future. *International Journal of Computer Integrated Manufacturing*, *3052*(May). http://doi.org/10.1080/0951192X.2016.1185155

Figueiras, P., Melo, R., Costa, R., Agostinho, C., Lima, C., & Jardim-Goncalves, R. (2015). A

Semantic Enrichment Approach Based on the Vector Space Model Supporting Collaboration in the Manufacturing Domain. *ASME 2015 International Mechanical Engineering Congress and Exposition. American Society of Mechanical Engineers*, 1–9.

Filipe, A., Guerreiro, M., Prof, V., Ferreira, P., Barbosa, D., Carlos, P., … Figueiredo, M. (2009). Construção de uma Metaheurística de Optimização de Rotas de Veículos. *Diss. Tese de Mestrado. Instituto Superior Técnico. Universidade Técnica de Lisboa*. Retrieved from https://fenix.tecnico.ulisboa.pt/downloadFile/395139478936/Disserta%C3%A7%C3%A3o_Andr%C3%A9_Guerreiro_53316.pdf

Gheisari, S., & Haghighat, A. T. (2008). SA-Mobicast : A Simulated Annealing-Based Mobicast Routing Protocol for Wireless Sensor Networks. *3rd International Symposium On. IEEE*, 529–534.

Gonc, L. B., Ochi, L. S., Martins, S. L., & Janeiro, R. De. (2005). A GRASP with Adaptive Memory for a Period Vehicle Routing Problem. *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06). Vol. 1. IEEE*.

Guan, C. (2010). Tabu Search Algorithm for Solving the Vehicle Routing Problem. *Information Processing (ISIP), 2010 Third International Symposium On. IEEE*.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 100–107. http://doi.org/10.1109/TSSC.1968.300136

http://passosapsicologia.blogspot.pt/. (2015). Retrieved February 12, 2016, from http://www.maestrogestao.com.br/wp-content/uploads/2013/08/qual-o-melhor-caminho.jpg

Hull, B., Bychkovsky, V., Zhang, Y., Chen, K., Goraczko, M., Miu, A., … Madden, S. (2006). CarTel : A Distributed Mobile Sensor Computing System. *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems. ACM*.

Jasika, N., Alispahic, N., Elma, A., Ilvana, K., Elma, L., & Nosovic, N. (2012). Dijkstra ' s shortest path algorithm serial and parallel execution performance analysis. *MIPRO, 2012 Proceedings of the 35th International Convention. IEEE*, 1811–1815.

Ji, X., Liu, L., Zhao, P., & Wang, D. (2015). A-Star Algorithm Based On-Demand Routing Protocol for Hierarchical LEO / MEO Satellite Networks. *Big Data (Big Data), 2015 IEEE International Conference On. IEEE*, 1545–1549.

Kawamura, M. S. (2006). Aplicação Do Método Branch-and-Bound Na Programação De Tarefas Em Uma Única Máquina Com Data De Entrega Comum Sob Penalidades de Adiantamento e Atraso. *Diss. Universidade de São Paulo*.

kuniga.wordpress.com. (2010). Retrieved January 20, 2016, from

https://kuniga.wordpress.com/2010/10/15/algoritmo-de-branch-and-cut/

Lee, W., Tseng, S., & Shieh, W. (2010). Collaborative real-time traffic information generation and sharing framework for the intelligent transportation system. *Information Sciences*, *180*(1), 62–70. http://doi.org/10.1016/j.ins.2009.09.004

Lin, T., Chen, Y., & Chang, H. (2014). Performance Evaluations of an Ant Colony Optimization Routing Algorithm for Wireless Sensor Networks. *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2014 Tenth International Conference On. IEEE*. http://doi.org/10.1109/IIH-MSP.2014.178

Ling-hui, C. (2010). Optimization of the VRP with Single Depot Based on Vehicle Coordination Strategy. *Intelligent Computation Technology and Automation (ICICTA), 2010 International Conference On. Vol. 2. IEEE*. http://doi.org/10.1109/ICICTA.2010.393

Lo, C., Peng, W., Chen, C., Lin, T., & Lin, C. (2008). CarWeb : A Traffic Data Collection Platform. *The Ninth International Conference on Mobile Data Management (Mdm 2008). IEEE*, 221–222. http://doi.org/10.1109/MDM.2008.26

Lysgaard, J. (1997). Clarke & Wright ' s Savings Algorithm. *Department of Management Science and Logistics, The Aarhus School of Business 44*.

Malaquias, N. (2006). Uso dos algoritmos genéticos para a otimização de rotas de distribuição. *Unpublished*. Retrieved from http://repositorio.ufu.br/handle/123456789/358

Onofre, S. M. (2007). Plataforma para testes de conformidade de sistemas baseados em módulos conceptuais STEP. *Diss. FCT-UNL*, 1–102.

Pereira, R. A., Nascimento, R. A., & Barbosa, S. (2012). Algoritmos Genéticos para Roteamento em Redes. *Unpublished*. Retrieved from http://www.rotadefault.com.br/docs/AG_Roteamento.pdf

Rana, H., Thulasiraman, P., & Thulasiram, R. K. (2013). MAZACORNET : Mobility Aware Zone based Ant Colony Optimization Routing for VANET. *2013 IEEE Congress on Evolutionary Computation. IEEE*, 2948–2955.

Risso, C., Robledo, F., & Sartor, P. (2013). Optimal Design of a Multi-Layer Network an IP/MPLS Over DWDM Application Case. Retrieved from http://www.intechopen.com/source/html/45119/media/3.jpg

Roque, I., & Junior, S. (2006). Comparação entre Métodos Exatos e Heurísticos para tratar o Problema de Roteamento de Veículos em um Ambiente Fabril. *Unpublished*, 1–9.

Schreckenberg, M., & Luther, W. (2013). Real-time Traffic Information System Using Microscopic Traffic Simulation. *Modelling and Simulation (EUROSIM), 2013 8th EUROSIM Congress On. IEEE*. http://doi.org/10.1109/EUROSIM.2013.83

Shuguang, L., Hongkai, Y., Jingru, Z., & Ke, Y. (2011). Multi-type Vehicles' Traffic Data

Collection Using Video Processing. *Intelligent Control and Information Processing (ICICIP), 2011 2nd International Conference On. Vol. 1. IEEE*.

SILVA, B. D. C. H. (2013). Otimização De Rotas Utilizando Abordagens Heurísticas Em Um Ambiente Georreferenciado. *Unpublished*.

slideplayer.com.br. (2007). Retrieved January 20, 2016, from http://slideplayer.com.br/slide/1733801/

Souza, C. C. De. (2013). UTILIZAÇÃO DOS ALGORITMOS GENÉTICOS COMO FERRAMENTA DE OTIMIZAÇÃO EM PROBLEMAS DE ROTEIRIZAÇÃO. *FACEF Pesquisa-Desenvolvimento E Gestão 15.3*, 285–297.

Tretmans, J. (1992). A formal approach to conformance testing. *Unpublished*.

Tretmans, J. (2001). An Overview of OSI Conformance Testing. *Samson, Editor, Conformance Testen, in Handboek Telematica 2*, 1–14.

www.jtscm.co.za. (2013). Retrieved February 11, 2016, from http://www.jtscm.co.za/index.php/jtscm/article/view/90/89

www.professeurs.polymtl.ca. (2000). Retrieved January 20, 2016, from http://www.professeurs.polymtl.ca/michel.gagnon/Disciplinas/Bac/IA/ResolProb/resproblema.html

www.slideshare.net. (2015). Retrieved January 20, 2016, from http://www.slideshare.net/oulasvir/modelbased-user-interface-optimization-part-ii-letter-assignment-at-sicsa-summer-school-on-computational-interaction-2015

www.theprojectspot.com. (2013). Retrieved January 20, 2016, from http://www.theprojectspot.com/tutorial-post/simulated-annealing-algorithm-for-beginners/6

Ying, S., & Yang, Y. (2008). Study on Vehicle Navigation System with Real-time Traffic Information. *Computer Science and Software Engineering, 2008 International Conference On. Vol. 4. IEEE*, 1079–1082. http://doi.org/10.1109/CSSE.2008.1447

Zhao, Z., Fang, J., Ding, W., & Wang, J. (2014). An Integrated Processing Platform for Traffic Sensor Data and Its Applications in Intelligent Transportation Systems. *2014 IEEE World Congress on Services. IEEE*. http://doi.org/10.1109/SERVICES.2014.38