INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Algoritmos para geração de rotas com base na análise de contexto

RICARDO EMANUEL CAPELAS PINTO
Outubro de 2019

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Algoritmos para geração de rotas com base na análise de contexto

Ricardo Emanuel Capelas Pinto

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Computacionais

Orientador: Doutora Maria Goreti Carvalho Marreiros

Co-Orientador: Mestre Luís Manuel Silva Conceição

Júri:

Presidente:

[Nome do Presidente, Categoria, Escola]

Vogais:

[Nome do Vogal1, Categoria, Escola]

[Nome do Vogal2, Categoria, Escola]

Porto, outubro 2019

À minha família e amigos que me apoiaram neste percurso e que nunca me deixaram desistir nos momentos mais difíceis.

Resumo

A crescente afluência turística que se tem vindo a registar nos últimos anos em Portugal, cria a necessidade de valorização dos nossos recursos culturais e promoção de novos percursos e itinerários. Com estas ideias em vista, a utilização de sistemas inteligentes capazes de combinar diferentes capacidades científicas e tecnológicas, em áreas como a gestão, marketing e TIC, tem vindo a ganhar maior importância no mercado. Regra geral, antes de visitar um país, os turistas procuram identificar os seus principais destinos e pontos de interesse. Indo de encontro a este processo de pesquisa e planeamento existem sistemas de recomendação para o turismo, que fornecem sugestões tendo em conta as preferências e interesses do utilizador.

O projeto TheRoute *(Tourism and Heritage Routes including Ambient Intelligence with Visitants' Profile Adaptation and Context Awareness)* tem como objetivo principal a investigação e experimentação no âmbito da geração automática de rotas para turistas e visitantes de pontos de interesse, considerando o conhecimento dos domínios das rotas, do perfil do visitante e ainda a adequação ao contexto. O contexto da viagem pode ser captado através de diversas fontes de informação sensorial como a localização do turista, o tempo da visita, as condições meteorológicas, assim como aspetos e características relevantes da atividade do utilizador. Entre estes aspetos costumam ser considerados o perfil e personalidade do utilizador, estado emocional e condição física. O trabalho desenvolvido nesta dissertação enquadra-se no projeto TheRoute, sendo o seu objetivo principal o desenvolvimento de um módulo de geração de rotas que considere o contexto do utilizador da aplicação e as condicionantes do meio.

Com o objetivo de resolver o problema proposto foram desenvolvidas duas soluções algorítmicas. Uma consiste na adaptação do algoritmo A* com cortes, enquanto outra é baseada no *Ant Colony Optimization*, um algoritmo de *Swarm Intelligence*. Para a primeira implementação foram ainda experimentadas duas heurísticas diferentes. A primeira consiste na seleção do caminho com a maior soma de pontuações de cada ponto de interesse. A segunda utiliza uma abordagem semelhante ao *Simulated Annealing* para a seleção de cada ponto de interesse a integrar no trajeto.

Os resultados obtidos nas experiências realizadas permitiram concluir que o algoritmo A* com cortes, orientado à heurística de seleção do caminho com maior pontuação, é o que obtém a conjugação de resultados mais favorável para as métricas de satisfação definidas.

Palavras-chave: Sistema de Recomendação, Turismo, Geração de Rotas, Análise de Contexto, Pontos de Interesse, Perfil do Utilizador

Abstract

The growth of tourism activity that has been registered in recent years in Portugal, has created a need of valuing our cultural resources and promote new routes and itineraries. With these ideas in mind, the use of intelligent systems capable of managing different scientific and technological capacities, in areas such as management, marketing and IT, has gained a greater relevance in the market. In general, before visiting a country, tourists try to identify their main destinations and points of interest. Meeting this research and planning process, there are tourism recommendation systems that provide suggestions based on user's preferences and interests.

The TheRoute project (Tourism and Heritage Routes including Ambient Intelligence with Visitants' Profile Adaptation and Context Awareness) has as its main goal the investigation and experimentation on automatic generation of routes for tourists and visitors of the points of interest, considering the knowledge of the routes, the profile of the visitor and the context awareness. The context of a trip can be taken through various sources of sensor information, such as the location of the tourist, the time of the visit, the weather conditions, as well as relevant aspects and characteristics of the user's activity. Among other aspects, there are considered the profile and personality of the user, emotional state and physical conditions. The work developed in this dissertation is part of the TheRoute project, and its main goal is the development of a route generation module that considers the context of the tourist and the environmental constraints.

In order to solve the proposed problem, two algorithmic solutions were developed. One is an adaption of the A* algorithm with cuts, while the other is based on Ant Colony Optimization, a Swarm Intelligence algorithm. For the first implementation two different heuristics were also tried. The first one selects the path with the highest sum of scores for each point of interest. The second one uses a similar approach to Simulated Annealing on the selection of each point of interest that is considered into the path.

The results from the experiments allowed to conclude that the A* with cuts, oriented to the heuristic for the path with the highest score, is the one that obtains the best conjugation of results for the defined satisfaction metrics.

Keywords: Recommendation System, Tourism, Route Generation, Context-Awareness, Points of Interest, User Profile

Agradecimentos

Agradeço de uma maneira geral a todos aqueles que, de forma direta ou indireta, contribuíram para tornar possível a concretização desta dissertação.

Expresso a minha gratidão à professora Goreti Marreiros e ao professor Luís Conceição por me terem proposto a elaboração desta dissertação e por me terem orientado ao longo do desenvolvimento do projeto.

Agradeço ainda à minha família e amigos, em especial aos meus pais e avós, que sempre me apoiaram e motivaram nos momentos mais difíceis e que compreenderam as minhas ausências e falta de disponibilidade ao longo destes últimos anos.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Acrónimos e Símbolos

Lista de Acrónimos

ACO	*Ant Colony Optimization*
API	*Application Programming Interface*
B&B	*Branch-and-Bound*
BFS	*Breadth-First-Search*
CSV	*Comma-Separated Values*
DAL	*Data Access Layer*
DAO	*Data Access Object*
FFE	*Fuzzy Front End*
HC	*Hill-Climbing*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
NCD	*New Concept Development*
PSO	*Particle Swarm Optimization*
QA	*Quality Assurance*
SA	*Simulated Annealing*
TheRoute	*Tourism and Heritage Routes including Ambient Intelligence with Visitants' Profile Adaptation and Context Awareness*
TIC	Tecnologias da informação e comunicação
TSP	*Travelling Salesman Problem*

Lista de Símbolos

$\sum$	Somatório
e	Número de Euler
$\sin$	Seno
$\cos$	Cosseno

1 Introdução

O presente documento relata as diversas fases de desenvolvimento da dissertação elaborada no âmbito da unidade curricular TESE/DISSERTAÇÃO/ESTÁGIO (TMDEI) do Mestrado em Engenharia Informática do Instituto Superior de Engenharia do Porto.

Este capítulo consiste numa breve apresentação do tema abordado "Algoritmos para geração de rotas com base na análise de contexto". Na secção 1.1 é introduzido o tema mencionado e exposto o seu enquadramento. Na secção 1.2 são descritos os objetivos para a concretização deste projeto, assim como uma breve descrição do trabalho realizado. Por fim, na secção 1.3 é apresentada a organização estrutural adotada para a dissertação.

1.1 Enquadramento

Nos últimos anos tem-se verificado um grande aumento na afluência turística em Portugal. De acordo com dados disponibilizados pelo Instituto Nacional de Estatística (INE), no ano de 2018 foram registados nos estabelecimentos hoteleiros portugueses 25.2 milhões de hóspedes e 67.7 milhões de dormidas. Estes valores representam aumentos anuais de 5.1% e 3.1%, respetivamente ("Portal do INE," 2019). O INE estima ainda que em 2017, o valor acrescentado bruto (VAB) proveniente da atividade turística tenha crescido 13.6% superando os de 6.6% de 2016 e atingindo 7.5% do VAB total da economia nacional. O setor representou 13.7% do PIB em 2017, tendo-se registado um aumento de 14.5% face aos resultados de 2016 ("Portal do INE," 2018). Estes valores são informação de dezembro de 2018 e reportam-se ao ano de 2017, não tendo sido ainda publicados os números referentes a 2018.

Segundo uma pesquisa do Instituto de Planeamento e Desenvolvimento do Turismo, baseado no número de chegadas aos aeroportos portugueses e na média de dias de permanência no nosso país, algumas das nossas cidades têm já mais turistas do que residentes. O estudo concluiu que em Lisboa existem 9 turistas por cada residente, no Porto 8 e em Albufeira, que lidera a lista, existem 39 turistas por cada morador (PÚBLICO, 2018).

Aleada a esta evolução da procura turística, existe uma crescente necessidade de valorização dos nossos recursos culturais e da promoção de percursos e itinerários que se revelem interessantes para quem nos visita. Da mesma forma, o uso de sistemas inteligentes capazes de utilizar diferentes capacidades científicas e tecnológicas, em áreas como a de gestão, marketing e TIC, tem vindo a ganhar maior importância no mercado.

O projeto TheRoute (*Tourism and Heritage Routes including Ambient Intelligence with Visitants' Profile Adaptation and Context Awareness*) tem como objetivo principal a investigação e experimentação no âmbito da geração automática de rotas para turistas e visitantes de pontos de interesse, considerando o conhecimento dos domínios das rotas, do perfil do visitante e ainda a adequação ao contexto *(Context-Awareness)* (Ramos et al., 2019). O trabalho desenvolvido nesta dissertação enquadra-se no projeto TheRoute, sendo o seu objetivo principal o desenvolvimento de um módulo de geração de rotas que considere o contexto do utilizador da aplicação e as condicionantes do meio. A aplicação existente é constituída por um *backend* em Java, que gere e disponibiliza os dados do sistema. O projeto contempla ainda um *site* para criadores de conteúdo. Este conteúdo é direcionado aos agentes turísticos que podem inserir dados relevantes para os pontos de interesse, como são exemplos os seus detalhes (nome, descrição, localização, etc.), categorias e itinerários fixos (Ramos et al., 2019). Existe também uma aplicação móvel desenvolvida em Ionic (Ramos et al., 2019), compatível com os principais sistemas operativos móveis Android e iOS, capaz de capturar *inputs* implícitos e explícitos do utilizador e do meio.

O TheRoute conta com uma equipa diversificada, ao combinar profissionais das áreas das TIC e tecnologias operacionais com uma diversidade de investigadores de ciências sociais, artes e humanidades. O projeto é assegurado por vários parceiros que, apesar de estarem simultaneamente envolvidos em todas as atividades, repartem entre si a sua coordenação. O Instituto Politécnico do Porto assume a estruturação da arquitetura do sistema, baseada em inteligência do ambiente, mobilidade, saúde, bem-estar e acessibilidade nas rotas. Cuida também da gestão do projeto, assim como da sua organização, divulgação e valorização. Por sua vez, o Instituto Superior de Engenharia do Porto coordena as atividades relativas à modelação de pontos de interesse, perfil dos visitantes e adequação ao contexto, com o desenvolvimento de algoritmos para geração de rotas. As atividades de aquisição de conhecimento em rotas gerais e específicas são executadas pelo Instituto Politécnico de Viana do Castelo. A Douro Azul, na sua condição de operador turístico, entra com o conhecimento relativo ao turismo e património para a região norte de Portugal, sendo igualmente responsável pela coordenação da atividade de teste e experimentação dos componentes do sistema TheRoute.

Regra geral, antes de visitar um país, os turistas procuram identificar os seus principais destinos e pontos de interesse, de modo a poderem planear antecipadamente a sua viagem. A informação sobre pontos turísticos, hospedaria e restaurantes, entre outros, pode ser obtida de diversas fontes, principalmente através de vários *sites* na Internet. Esta opção apresenta alguns problemas, nomeadamente quanto à diversidade de fontes de informação e à atualidade da informação exibida. O tempo que se perde na organização e planeamento da

viagem é um dos fatores mais negativos com que se depara quem vai viajar (Borràs, Moreno, & Valls, 2014). Para auxiliar este processo de decisão e planeamento existem sistemas de recomendação para o turismo, que fornecem sugestões tendo em conta as preferências e interesses do utilizador (Ricci, Rokach, & Shapira, 2015). Em muitos casos estes sistemas também podem ter em consideração o contexto dinâmico da viagem (Borràs et al., 2014).

O contexto da viagem pode ser captado através de diversas fontes de informação sensorial como a localização do turista, o tempo da visita, as condições meteorológicas (Borràs et al., 2014), assim como aspetos e características relevantes da atividade do utilizador. Entre estes aspetos costumam ser considerados o perfil do utilizador, estado emocional, condição física, atividades do quotidiano (Champiri, Shahamiri, & Salim, 2015), etc. Os dados recolhidos podem ser processados por um sistema que comunica ativamente com o utilizador, sugerindo-lhe ações calculadas através da análise desses *inputs*. Na área do turismo, mais especificamente, um sistema dotado com tais capacidades consegue, por exemplo, ajustar um plano de visita caso o visitante demore mais tempo do que o planeado num dos pontos de interesse ou, caso essa reorganização do trajeto seja impraticável, avisar com antecedência o utilizador sobre esse atraso.

1.2 Objetivos e Breve Descrição do Trabalho Realizado

O trabalho realizado para a concretização desta dissertação enquadra-se em duas áreas principais: algoritmia e adequação ao contexto. Como objetivo principal pretende-se o desenvolvimento de um módulo capaz de gerar rotas, tendo em conta diversos *inputs* com origens distintas. Entre estes *inputs* destacam-se os pontos de interesse a visitar, o seu horário de funcionamento, as previsões meteorológicas para cada local, o tempo total dedicado para completar o trajeto e até mesmo algumas características do perfil do utilizador.

Para atingir este objetivo principal, o trabalho é divido nos seguintes pontos inerentes à sua concretização:

1. Investigação e respetiva realização do estado da arte nas áreas de adequação ao contexto e as principais abordagens algorítmicas para geração de rotas;

2. Identificação das abordagens mais relevantes e apropriadas para o problema em questão;

3. Desenho e conceção do módulo de geração de rotas tendo em conta modelos de análise de contexto, assim como os algoritmos de geração de rotas;

4. Experimentação e avaliação do módulo de geração de rotas;

5. Integração do módulo desenvolvido no protótipo do TheRoute;

6. Conceção e experimentação de cenários reais para o teste e validação do sistema TheRoute.

A validação do trabalho é suportada através da avaliação do algoritmo de geração de rotas, tendo em conta as três hipóteses seguintes:

1. O algoritmo escolhido é eficiente, permitindo obter soluções válidas para os diversos parâmetros de entrada num curto espaço de tempo. O tempo limite considerado para o processo de criação de uma rota personalizada é 30 segundos. Este tempo pode variar, dependendo das condições da ligação e da carga do sistema, não devendo ultrapassar os 35 segundos;

2. É possível a recomendação de um trajeto que agrade ao turista, tendo em conta vários aspetos pessoais do seu perfil e condicionantes do contexto envolvente, sem que o utilizador necessite de perder tempo a configurar esse trajeto;

3. A rota sugerida deve ocupar tempos próximos aos solicitados pelo turista, de forma a preencher praticamente a totalidade do horário dedicado ao passeio.

É expectável que os utilizadores da aplicação desistam de a usar se cada pedido de rota demorar muito tempo a ser concretizado. Desta forma, o tempo limite definido tem em conta o fator utilizador e as limitações computacionais que a criação da rota acarreta, criando um compromisso de sustentabilidade entre a aplicação e o turista.

1.3 Estrutura

A presente dissertação foi organizada de forma a proporcionar aos seus leitores uma visão alargada das fases de elaboração do módulo de geração de rotas para o projeto TheRoute.

Durante o capítulo atual (Introdução) pretende-se fazer uma breve apresentação do projeto, de forma a familiarizar o leitor com o seu âmbito e oferecer uma base de conhecimento para um melhor entendimento dos capítulos seguintes.

O capítulo que se segue, Estado da Arte, tem como principal objetivo aprofundar o contexto do projeto desenvolvido. No Estado da Arte podem ser consultados aspetos relevantes tais como a análise de valor de negócio, uma investigação sobre algoritmos para geração de rotas e um estudo das soluções já existentes para a recomendação de turismo.

O capítulo 3, Análise e Design, apresenta algumas das decisões tomadas relativamente à arquitetura adotada e às interações a efetuar entre os componentes do sistema. Essas decisões surgem após uma análise detalhada ao problema que se pretende resolver nesta dissertação.

O capítulo 4, Implementação da Solução, é o responsável pela explicação das etapas necessárias para o desenvolvimento da solução idealizada, tendo em conta os pressupostos definidos no capítulo da Análise e Design.

No capítulo 5, Experimentação e Avaliação, são expostos os resultados dos testes realizados ao módulo desenvolvido, assim como as experiências elaboradas para atingir a solução final.

Por fim, é apresentado o capítulo 6 dedicado às Conclusões, no qual são expostas várias reflexões críticas alusivas aos resultados obtidos e às metodologias adotadas. Também são abordadas as limitações do trabalho realizado, bem como algumas sugestões para um possível trabalho futuro.

2 Estado da Arte

Neste capítulo é inicialmente apresentada a análise de valor de negócio para a qual o módulo desenvolvido irá contribuir, sendo de seguida enumeradas algumas das soluções já disponíveis no mercado com características idênticas às do sistema que se pretende desenvolver. Por fim são ainda exibidas as pesquisas efetuadas sobre alguns dos algoritmos existentes para geração de rotas.

2.1 Análise de Valor de Negócio

Como já foi referido no capítulo 1 (Introdução) a afluência turística a Portugal tem evoluído de forma crescente e continuada. Como tal, a criação de um *software* de recomendação turística pode-se revelar uma boa aposta de negócio, facilitar o planeamento de uma viagem por parte do turista, impulsionar a atividade turística e ajudar a promover o património português.

O principal objetivo de uma análise de valor é avaliar a melhor forma de aumentar o valor de um produto ou serviço, minimizando os custos e a perda de qualidade (Monczka, Handfield, & Giunipero, 2008). A proposta de valor levada a cabo no decorrer desta dissertação passa pelo desenvolvimento de um módulo de geração de rotas a integrar no sistema TheRoute. Desta forma, este módulo representa um componente de um sistema já em desenvolvimento, tendo um papel fundamental no seu funcionamento e, como tal, na sua geração de valor. Uma vez que o desenvolvimento deste componente tem o objetivo de integração no produto TheRoute, não sendo adquirido separadamente, a análise de valor incidirá simultaneamente sobre o módulo desenvolvido e sobre o produto final a disponibilizar ao utilizador.

De acordo com o modelo *New Concept Development* (NCD), desenvolvido por Peter Koen, o processo de inovação encontra-se dividido em três partes: a linha da frente da inovação (*the front end of innovation* ou *fuzzy front end* (FFE)), o processo de desenvolvimento do novo produto (*the new product development process*) e a comercialização do produto. Segundo os autores, o FFE é um componente de extrema importância no processo de inovação, uma vez

que as decisões tomadas nesta fase influenciarão as opções tomadas no processo de desenvolvimento do produto e na sua comercialização (Koen, Bertels, & Kleinschmidt, 2014).

O NCD é constituído por três partes (Figura 1):

- O Motor (*engine*), que representa a liderança, cultura e estratégia da organização que impulsiona os cinco principais elementos controlados pela entidade empreendedora;

- A área interna, responsável por definir os cinco elementos controláveis da atividade do FFE (identificação da oportunidade, análise da oportunidade, criação e enriquecimento da ideia, seleção da ideia e definição de conceito);

- Os fatores de influência, que consistem nas capacidades organizacionais, no mundo externo (canais de distribuição, leis, políticas governamentais, clientes, concorrentes e clima político-económico) e as ciências (internas e externas) que podem estar envolvidas. Estes fatores não são controláveis pela organização, ao contrário dos elementos identificados na área interna (Koen et al., 2002).

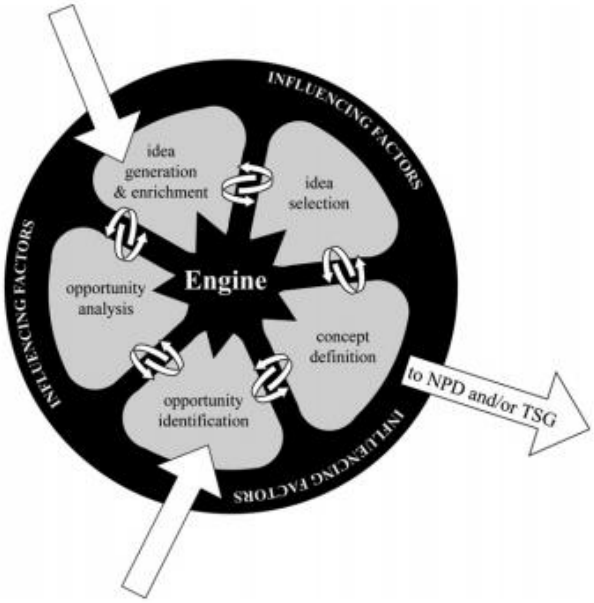


Figura 1 – Modelo NCD
(Koen et al., 2002)

A contínua evolução tecnológica tem potenciado a modernização de vários processos do quotidiano humano. Nos dias de hoje o acesso à Internet e aos *smartphones* está facilmente ao alcance da maioria dos cidadãos residentes nos chamados países desenvolvidos. A procura na área do turismo tem vindo a aumentar nos últimos anos, também potenciada por esta evolução tecnológica e pela enorme capacidade de propagação de informação que estes meios proporcionam. É certo que já praticamente ninguém parte à descoberta de um novo local sem antes consultar na Internet os seus principais pontos de interesse, locais de estadia, restaurantes, etc. A par da definição do local de destino da viagem é também normalmente

estipulado o período de duração, dividido pelos diversos pontos de passagem. Esta definição, para além de ser morosa, pode ser bastante irrealista, porque o viajante ou o turista desconhece muitos outros fatores que podem arruinar por completo o seu plano de viagem. A simples consulta de informação através de fontes não fidedignas, as condições ambientais, os atrasos no transporte e nos tempos das visitas, são apenas alguns exemplos que podem interferir e provocar desvios no planeamento de um itinerário de viagem.

Com o objetivo de resolver ou tentar atenuar alguns dos problemas acima identificados e aproveitando a **oportunidade** criada pela crescente afluência de turismo no norte do país, assim como a necessidade de promover itinerários como forma de aproveitamento das principais infraestruturas de entrada de visitantes no nosso país, o GECAD propôs-se a desenvolver um sistema inteligente de recomendação turística. Apesar de já existirem algumas alternativas, como as apresentadas na secção 2.2, nenhuma aparenta preencher por completo os requisitos e as ambições pretendidas, abrindo assim espaço e oportunidade à criação de novas alternativas. Algumas aplicações presentes no mercado já utilizam métodos bastante avançados para a o planeamento da rota. No entanto, em determinadas situações revelam alguma falta de conjugação de critérios e pouco detalhe e assertividade no que diz respeito a informação de pontos de interesse em Portugal.

A **fase de criação e enriquecimento da ideia** culminou com a decisão de criar um sistema de recomendação turístico que, além de gerar automaticamente rotas tendo em conta o perfil individualizado do utilizador e o contexto meio envolvente, é capaz de minimizar a perspetiva de custo total do tempo de viagem e de espera.

A **fase de seleção da ideia** é um dos objetivos desta dissertação e passa por estudar, experimentar e decidir qual a abordagem algorítmica mais apropriada e capaz de oferecer mais garantias para uma melhor resolução do problema em questão.

Por fim e para concluir a análise dos elementos chave do NCD, a **fase de definição do conceito** passa pela implementação, teste e integração do módulo de geração de rotas no sistema TheRoute. Para apoiar esta definição foram estabelecidas algumas parcerias estratégicas com algumas entidades, como por exemplo a Douro Azul, com o intuito de ajudar os proponentes e o autor desta dissertação na melhoria de processos externos às áreas de competência tecnológica. Um dos exemplos desse apoio foi a delegação da gestão de conteúdos dos pontos de interesse a uma entidade externa, que em virtude da sua larga experiência no ramo possui um amplo conhecimento na área do turismo, sendo o parceiro ideal para a obtenção dos melhores resultados.

2.1.1 Valor de Negócio

O conceito de valor é definido em diferentes contextos como a necessidade, desejo ou interesse (Nicola, Ferreira, & Ferreira, 2012) na aquisição ou usufruto de um produto ou serviço. A criação de valor é fundamental para qualquer negócio, uma vez que qualquer atividade comercial envolve a troca de algum bem ou serviço tangível e/ou intangível (Nicola

et al., 2012). De acordo com (Zeithaml, 1988) o valor percebido "é a avaliação geral do consumidor sobre a utilidade de um produto com base nas perceções do que é recebido e do que é dado". O valor para o cliente é, segundo (Woodall, 2003), qualquer perceção pessoal na vantagem que pode surgir através de uma associação entre a procura por parte de um cliente e a oferta disponibilizada por uma organização. Este valor deve ser resultado do equilíbrio entre os benefícios e os sacrifícios, sendo que deve contar com a redução de sacrifícios para proporcionar um maior valor para o cliente.

Como já foi referido no início da secção 2.1.1, a proposta de valor levada a cabo no decorrer desta dissertação passa pelo desenvolvimento de um módulo de geração de rotas a integrar num sistema que já se encontra em desenvolvimento, o TheRoute. Uma vez que as funcionalidades que este módulo tem para oferecer ao sistema são fulcrais para o seu funcionamento, o seu contributo para o valor do produto final será muito elevado. Com a integração deste módulo pretende-se oferecer ao utilizador a possibilidade de gerar um trajeto para uma visita a vários pontos de interesse num determinado período de tempo, tendo em consideração aspetos como a análise do contexto e os gostos e perfil do turista. Com a disponibilização deste produto espera-se que os seus utilizadores percam menos tempo no planeamento e organização da viagem, podendo desfrutar mais do seu passeio.

O valor que o módulo desenvolvido pode acrescentar para os seus utilizadores, assim como o produto final no qual ele se insere, encontra-se representado na Tabela 1 sob a forma de uma proposta longitudinal de valor. Os benefícios para o cliente são concretizados pela oferta de um produto de qualidade e utilidade, que se propõe a facilitar a tarefa de pesquisa e programação de um passeio turístico.

Numa fase inicial, no estado de Pré-Compra, os benefícios para o cliente estão relacionados com a possibilidade de redução no tempo despendido para o planeamento e organização do itinerário da visita. O cliente pretende uma solução que lhe proporcione um planeamento de qualidade, tendo em conta os seus gostos. Por essa razão, o utilizador quer uma solução que lhe permita obter resultados ajustáveis ao seu perfil, ou seja, um produto personalizado. Por outro lado, poderá existir um sacrifício relacionado com o tempo que o cliente perde até encontrar esta solução de *software* e com o tempo que demora até o decidir adquirir.

Na fase de Compra espera-se que o cliente tenha facilidade na aquisição do produto. O utilizador terá à sua disposição um bom suporte de serviço, com acesso a informação fidedigna e atualizada e um produto capaz de lhe oferecer flexibilidade na geração das rotas, tendo em conta fatores provenientes do contexto envolvente. O sacrifício que terá nesta fase está relacionado com a necessidade de ligação à Internet para troca de informação com o servidor, com o consequente consumo de dados móveis. No entanto, este ponto é minimizado pela cada vez maior disponibilidade de pontos de acesso Wi-Fi grátis que proliferam um pouco por todo o lado.

Para a fase de Pós-Compra espera-se uma efetiva redução no tempo de procura de pontos de interesse e planeamento da visita, por parte do utilizador. É expectável que o produto se adeque aos gostos pessoais do turista. O inconveniente encontrado consiste na possibilidade

de o algoritmo de geração de rotas poder não encontrar soluções ótimas para todas as situações. Por outras palavras, devido a condicionantes do meio e ao período dedicado à visita, pode não ser possível arranjar uma solução que satisfaça todos as vontades do utilizador.

Por fim, no estado de Pós-Utilização é expectável que o cliente se sinta satisfeito com a qualidade e utilidade do produto, usando-o novamente para as suas deslocações e futuras visitas turísticas.

Tabela 1 – Proposta longitudinal de valor

	Benefícios	**Sacrifícios**
Pré-Compra	Redução do tempo despendido no planeamento e organização de um trajeto; Qualidade no planeamento da rota; Personalização do produto	Perda de tempo na procura do *software*
Compra	Fácil aquisição; Informação fidedigna e atualizada.	Necessária ligação à Internet e consequente gasto de dados móveis
Pós-Compra	Efetiva redução no tempo de planeamento da visita; Efetiva redução no tempo de procura de pontos de interesse; Adequação do produto aos gostos pessoais	O algoritmo pode não obter soluções ótimas para todas as situações
Pós-Utilização	Satisfação com o produto; Utilidade do produto	

2.1.2 Modelo de Negócio

O modelo de negócio idealizado para o sistema proposto nesta dissertação encontra-se representado na Figura 2, sob a forma de um *Business Model Canvas*.

A proposta de valor levada a cabo no decorrer desta dissertação passa pelo desenvolvimento de um módulo de geração de rotas a integrar num sistema de recomendação de turismo, o TheRoute. Este sistema de recomendação é o produto final a comercializar e visa simplificar e reduzir o tempo de planeamento de uma viagem ou passeio turístico.

As atividades-chave para a concretização desta dissertação passam pela investigação sobre os diversos modelos de análise de contexto e algoritmos de geração de rotas; pela criação e desenvolvimento de um módulo de geração de rotas, tendo por base uma arquitetura de *software* sólida e consistente; pela conceção e experimentação de cenários reais de teste para provar a robustez, a fiabilidade e o elevado desempenho do *software* desenvolvido.

Os recursos-chave identificados foram a equipa de investigação e desenvolvimento, a equipa de gestão de conteúdos e a equipa de ajuda e suporte. A primeira equipa é responsável pelo desenvolvimento e integração do módulo no sistema já existente, assim como por testar a solução desenvolvida e reparar algum problema reportado pela equipa de ajuda e suporte. A equipa de gestão de conteúdos tem como papel principal o registo de dados relativos aos pontos de interesse turísticos e outras informações importantes a considerar para um melhor funcionamento do sistema.

Como parceiros-chave foram identificadas as agências de viagem, os centros de turismo, os municípios e a empresa de cruzeiros Douro Azul. Todas estas entidades beneficiam com o turismo e podem também sair favorecidas com o lançamento deste produto, uma vez que um dos seus objetivos é promover o país e potenciar a atividade turística. A cooperação poderá ser estendida ao nível do intercambio de informações úteis a apresentar na aplicação, assim como patrocinar anúncios publicitários e ajudar a difundir a aplicação pelos seus clientes.

Uma vez que a finalidade do produto se enquadra na área do turismo, a segmentação de clientes é constituída por turistas, viajantes e peregrinos. A diferença entre estes três tipos de segmento é a objetivo da visita, sendo que é o turista é o indivíduo que comummente viaja por lazer, o viajante aquele que se encontra de passagem pelo local e o peregrino o que se desloca num determinado trajeto em prol da sua região (ex. rota de Santiago).

Os canais de distribuição escolhidos foram as conferências científicas, as feiras tecnológicas, a divulgação promocional na Internet e a publicação da aplicação móvel nas lojas das principais plataformas móveis (Android e iOS). Numa fase inicial e devido ao foro científico do trabalho, pretende-se difundir o produto desenvolvido pela comunidade científica, de modo a ganhar alguma credibilidade e notabilidade por parte dos entendidos na tecnologia. Dessa forma, as conferências científicas serão um bom meio para divulgar as características mais técnicas da solução final. Através da presença em feiras tecnológicas, será mais fácil chegar perto de potenciais clientes, uma vez que neste tipo de eventos participam pessoas que procuram soluções tecnológicas para diversas áreas de interesse. Com a publicação da aplicação móvel na Apple App Store e na Google Play, consegue-se também uma maior visibilidade junto do segmento de clientes que procura uma aplicação para telemóvel com as potencialidades de sugestão e geração automática de rotas turísticas.

Para cimentar uma boa relação com o cliente, os dois produtos disponibilizados ao público-alvo, a aplicação móvel e o *website*, terão um serviço de ajuda e suporte, para lidar com as sugestões e dificuldades encontradas pelos seus utilizadores. Em cada lançamento de uma nova versão do *software* serão divulgadas as novidades implementas, de forma a dar conhecer ao utilizador as novas potencialidades do produto. Como a opinião dos utilizadores da aplicação é importante para o valor do produto, será disponibilizado em cada um dos produtos um formulário que possibilita o registo de opiniões e sugestões para uma contínua melhoria do sistema. Adicionalmente, existe um serviço de ajuda e suporte, cujo responsável está incumbido de filtrar e direcionar sugestões e problemas reportados por utilizadores da *app* e do *website*.

A estrutura de custos a suportar está relacionada com o vencimento dos recursos humanos necessários ao desenvolvimento e manutenção do produto e a contratação de serviços *web* para a alojamento do *website*, armazenamento de dados e entrega de conteúdo.

As principais fontes de receita advêm da disponibilização do *software* como um serviço (*Software as a Service*), tanto na disponibilização do *website* como na distribuição da aplicação móvel pelas lojas *online* das principais plataformas. Pretende-se também arrecadar receitas através da disponibilização de publicidade de terceiros nos produtos comercializados. Poderá também ser possível obter rendimentos através da venda do *software* a uma ou mais empresas, numa fase de maior maturação do produto.

Figura 2 – Modelo de negócio Canvas

2.2 Soluções e Abordagens Semelhantes em Sistemas de Recomendação para Turismo

Tradicionalmente, a maioria dos sistemas de recomendação para turismo, à semelhança dos restantes sistemas de recomendação, concentra-se em fazer sugestões individualizadas para o utilizador (Adomavicius & Tuzhilin, 2009). Estes sistemas têm em consideração múltiplas características pessoais e gostos do indivíduo, mas a grande maioria não tem em conta informação contextual, como local, horário, condições atmosféricas e até sobre a companhia de outras pessoas (Adomavicius & Tuzhilin, 2009), ex. viajar sozinho ou em grupo. No entanto, a inclusão da variante contextual pode influenciar bastante os resultados sugeridos pelo sistema, podendo contribuir para a experiência do utilizador. Para o caso do turismo não seria apropriado, por exemplo, recomendar a visita a um local descoberto num dia de chuva intensa. Também não seria, de todo, agradável recomendar a visita a um local num período do ano ou horário do dia no qual este estivesse fechado. Por estas e outras situações, o envolvimento das variáveis de contexto e a sua análise podem ser determinantes para a geração de bons resultados.

A constante evolução tecnológica, nomeadamente nos sistemas de informação e dispositivos móveis, tem potenciado a procura e utilização deste tipo de sistemas de recomendação por parte dos turistas. As secções seguintes apresentam alguns exemplos de produtos ou protótipos desenvolvidos nos últimos anos para recomendação de pontos e rotas turísticas.

2.2.1 iTravel

O iTravel foi desenvolvido no âmbito de uma pesquisa com o objetivo de facilitar a tarefa de recomendação de lugares, ao explorar as classificações atribuídas por outros turistas após as suas visitas (W.-S. Yang & Hwang, 2013). Com o recurso a comunicações *peer-to-peer* móveis, como o Bluetooth ou o Wi-Fi, os autores deste estudo pretendiam fornecer aos turistas meios práticos e baratos para a troca de avaliações armazenadas nos seus dispositivos móveis (W.-S. Yang & Hwang, 2013). Ao utilizar dispositivos móveis e comunicações *peer-to-peer*, um turista pode detetar continuamente outros turistas que lhe estejam próximos e trocar informações diretamente com eles sem a necessidade de recorrer a infraestruturas adicionais (W.-S. Yang & Hwang, 2013). Essa arquitetura encontra-se ilustrada na Figura 3. Cada dispositivo móvel possui uma base de dados própria para armazenar as classificações do utilizador acerca do lugar visitado. Esses dados são posteriormente analisados por um componente cuja função é decidir pontuações para as atrações não visitadas. Existe um gestor de posicionamentos responsável por detetar a localização do utilizador. A comunicação entre dispositivos é assegurada por um gestor de comunicação, que troca dados de classificação através de Wi-Fi e Bluetooth. O gestor de dados gere as avaliações do utilizador e as classificações recebidas de outros intervenientes. O gestor de interface tem a responsabilidade de apresentar as informações ao utilizador do dispositivo e de receber as suas classificações sobre os locais.

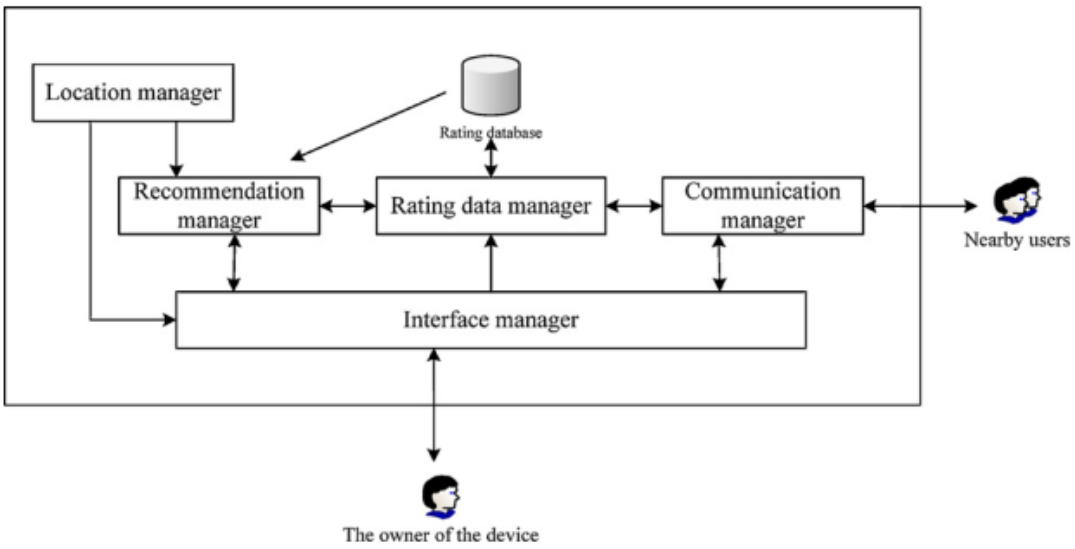


Figura 3 – Arquitetura do sistema iTravel
(W.-S. Yang & Hwang, 2013)

Com a realização deste estudo os autores tencionavam demonstrar que utilizadores que visitam os mesmos lugares têm uma maior probabilidade de partilhar gostos semelhantes (W.-S. Yang & Hwang, 2013). Os testes efetuados com o protótipo desenvolvido revelaram que o sistema desenvolvido conseguia trocar e mostrar informações úteis na hora e local certos, ajudando o utilizador a tomar decisões no decorrer da viagem.

2.2.2 CT-Planner

O CT-Planner é um sistema de planeamento de viagens assistido por computador que utiliza como base o perfil do utilizador, a avaliação dos pontos de interesse e a participação do utilizador (Kurata & Hara, 2013). Para evitar perda de tempo na definição do perfil do utilizador, o CT-Planner faz apenas duas perguntas antes de gerar a recomendação: o destino da viagem e o estilo de viagem favorito (Kurata & Hara, 2013). A aplicação *web* desenvolvida suporta cinco tipos diferentes de estilos de viagem: visitar as várias atrações do local, passeio pela cidade, passeio em locais relaxantes, conhecer as localidades e caminhar com crianças (Kurata & Hara, 2013). Através das escolhas do turista o sistema determina o valor inicial do perfil do utilizador. O processo que se segue é a avaliação personalizada do ponto de interesse, que tem em consideração fatores inerentes ao local como é o caso da análise de contexto. Os autores dão como exemplo a ideia de que o mesmo turista pode atribuir diferentes avaliações ao mesmo ponto de interesse dependendo da estação do ano, condições meteorológicas ou até mesmo pelo facto do turista fazer essa visita sozinho ou acompanhado (Kurata & Hara, 2013). Por estas razões, o CT-Planner adotou uma abordagem cíclica para a sugestão do trajeto a efetuar na viagem. A aplicação propõe um plano ao utilizador que, por sua vez, o examina e devolve a sua opinião. O sistema volta a examinar os dados recolhidos e revê o plano em concordância com os mesmos, tornando o processo iterativo ao não forçar o

utilizador a introduzir os seus interesses de uma só vez (Kurata & Hara, 2013). Como resultado é apresentado um mapa como o da Figura 4, ilustrado com o auxílio da API do Google Maps, com o planeamento da viagem.

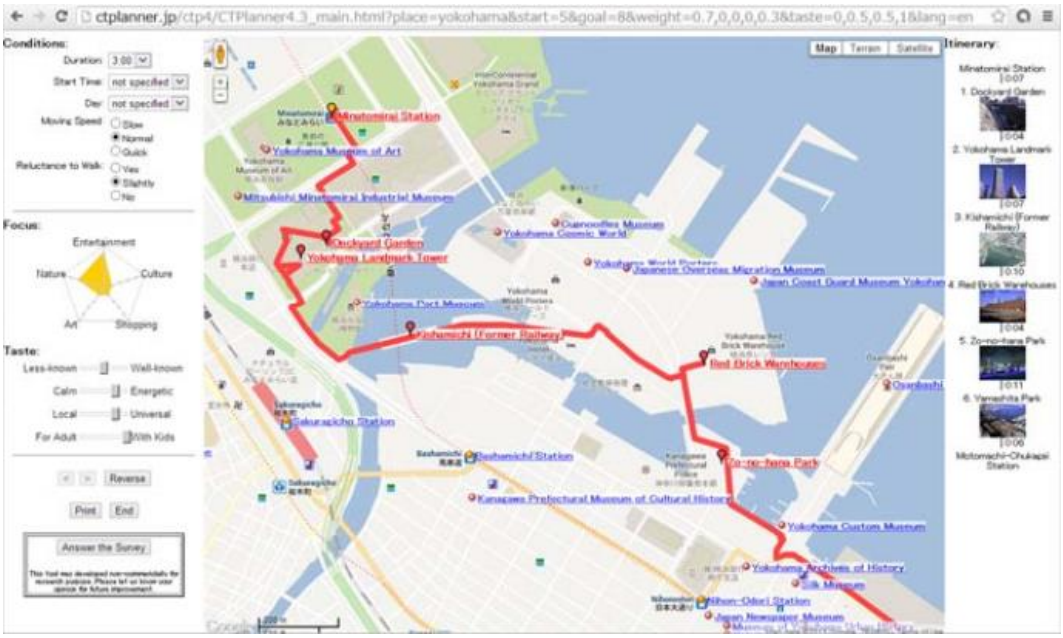


Figura 4 – Exemplo de um trajeto em Yokohama (Japão) proposto pelo CT-Planner (Kurata & Hara, 2013)

Na imagem é também possível verificar os atributos personalizáveis para a criação do trajeto. As condições da viagem têm em consideração cinco variáveis: duração, hora de início, dia da semana, velocidade de caminhada e relutância a caminhar. O perfil do utilizador é definido por duas partes: focos e gostos. Os seus valores são pré-determinados com base na seleção inicial de estilos de viagem favoritos (Kurata & Hara, 2013).

2.2.3 City Trip Planner

O City Trip Planner é um sistema especializado em turismo que oferece a possibilidade de planear trajetos em cinco cidades belgas (Vansteenwegen, Souffriau, Berghe, & Oudheusden, 2011). Este sistema propõe viagens tendo em consideração os interesses pessoais do utilizador e o contexto da viagem. Além disso, a aplicação, desenvolvida em tecnologias *web*, possui algumas funcionalidades peculiares, como o agendamento de intervalos para o almoço e a sugestão de pontos de interesse por parte das agências de turismo da cidade (Vansteenwegen et al., 2011).

O City Trip Planner sugere viagens personalizadas, tendo por base um pequeno questionário inicial feito ao utilizador com vista a aferir os seus gostos e preferências. A partir das respostas recolhidas, são geradas pontuações para cada ponto de interesse que um turista possa vir a visitar, dentro das restrições criadas pelo contexto da própria viagem. De seguida, um algoritmo heurístico atua sobre os dados recolhidos, resultando na proposta de uma viagem

personalizada, adaptada aos interesses do utilizador, à sua localização atual, ao destino, à sua disponibilidade de horário e ao horário de funcionamento dos pontos de interesse (Vansteenwegen et al., 2011). A arquitetura descrita encontra-se representada na Figura 5.

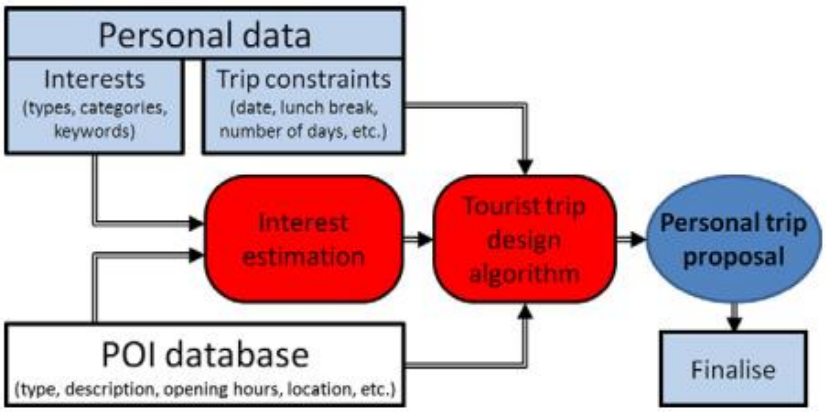


Figura 5 – Arquitetura do sistema City Trip Planner
(Vansteenwegen et al., 2011)

O trajeto gerado é apresentado sobre a forma de uma tabela (Figura 6) ou mapa e pode ser impresso ou carregado para um dispositivo de navegação GPS. Para cada ponto sugerido o utilizador pode também escolher se pretende ou não visitar esse local.

	Attraction	Arrival			
1	**Starting point: Railway Station**	**11:00**			
	Concert Hall - inside ★★★	11:08	5	☐	☐
	't Zand - square ★★★	11:14	5	☐	☐
	Concert Hall - outside ★★★	11:20	5	☐	☐
	Saint Saviour's Cathedral - outside ★★★	11:28	5	☐	☐
2	**Beguinage - inside** ★★★	**11:40**	**15**	☑	☐
	De Vos - almshouses ★★★	11:56	5	☐	☐
	Bogaerdenschool - outside ★★★	12:02	5	☐	☐
	Saint Joseph - De Meulenaere - alms... ★★★	12:09	5	☐	☐
	Hospitaalmuseum - Memling in Sint-J... ★★★	12:18	5	☐	☐
3	**BREAK**	**12:38**	**60**		
	Bruggemuseum - Welcome Church of Ou... ★★★	13:38	5	☑	☐
	Episcopal palace - outside ★★★	13:43	5	☐	☐
	Bruggemuseum - Gruuthuse - outside ★★★	13:49	5	☐	☐
	Hof van Beveren (former) - outside ★★★	13:56	5	☐	☐
	Hof van Watervliet - outside ★★★	14:01	5	☐	☐
4	**House Perez de Malvenda -outside** ★★★	**14:09**	**10**	☐	☐

Figura 6 – Exemplo de um trajeto em Bruges (Bélgica) proposto pelo City Trip Planner
(Vansteenwegen et al., 2011)

2.2.4 Visitacity

O Visitacity é uma plataforma para recomendação de turismo disponível através de um *website* e de uma aplicação móvel, compatível com os sistemas Android e iOS. Este produto fornece uma variada quantidade de funcionalidades para o planeamento de uma viagem. Após escolher o destino a visitar, o utilizador pode ver o mapa turístico do local, contendo os seus principais pontos de interesse. Existe também a possibilidade de se poder consultar algumas sugestões sobre aspetos culturais, produtos e atividades regionais a conhecer. Consegue-se igualmente listar as principais atrações turísticas organizadas por categoria. Por último, mas não menos importante, é possível gerar itinerários para períodos entre um a três dias. A Figura 7 ilustra um exemplo de planeamento de um itinerário para o primeiro de três dias na cidade do Porto.

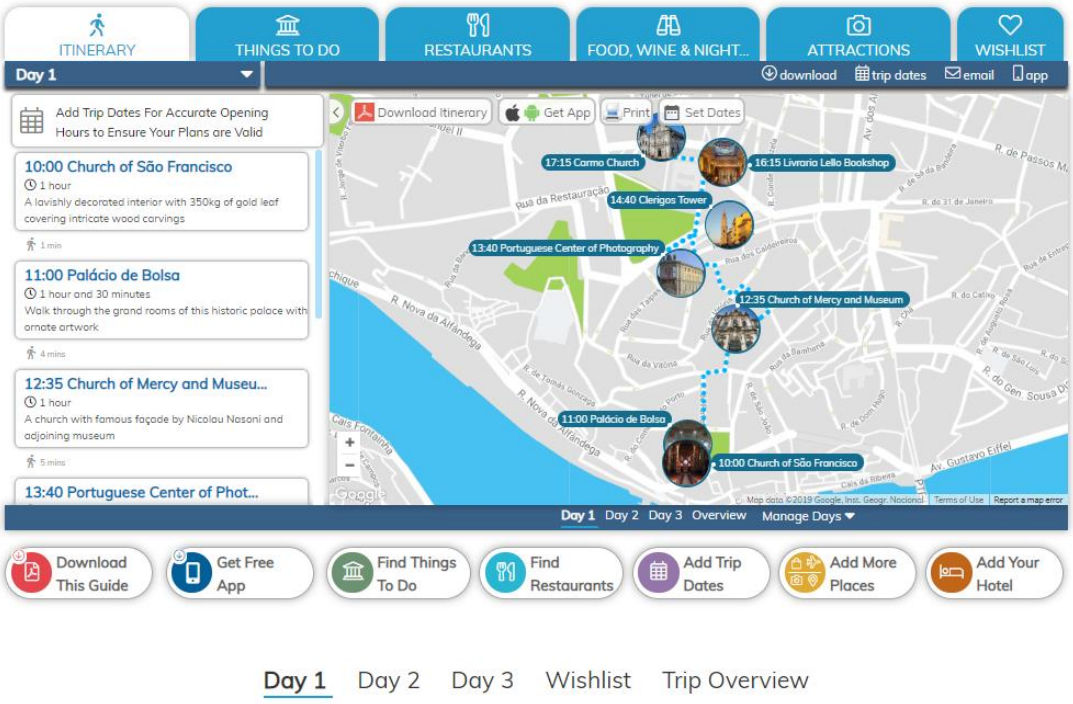


Figura 7 – Planeamento de itinerário no *website* do Visitacity
(Visitacity, n.d.)

O comportamento da aplicação móvel é muito semelhante ao do *site*, fornecendo assim a mesma variedade de funcionalidades. A Figura 8 ilustra um exemplo de um planeamento turístico para um dia na cidade de Paris. Na apresentação do mapa é possível visualizar os pontos a visitar, assim como as respetivas horas de visita. A vista que permite ver o plano detalhado da viagem apresenta, de forma cronológica, os locais a visitar. Entre cada elemento da lista é exibida a distância necessária a percorrer para alcançar o ponto seguinte. É também possível consultar para cada ponto de interesse a duração de visita estimada, o horário de funcionamento e as direções para o local.

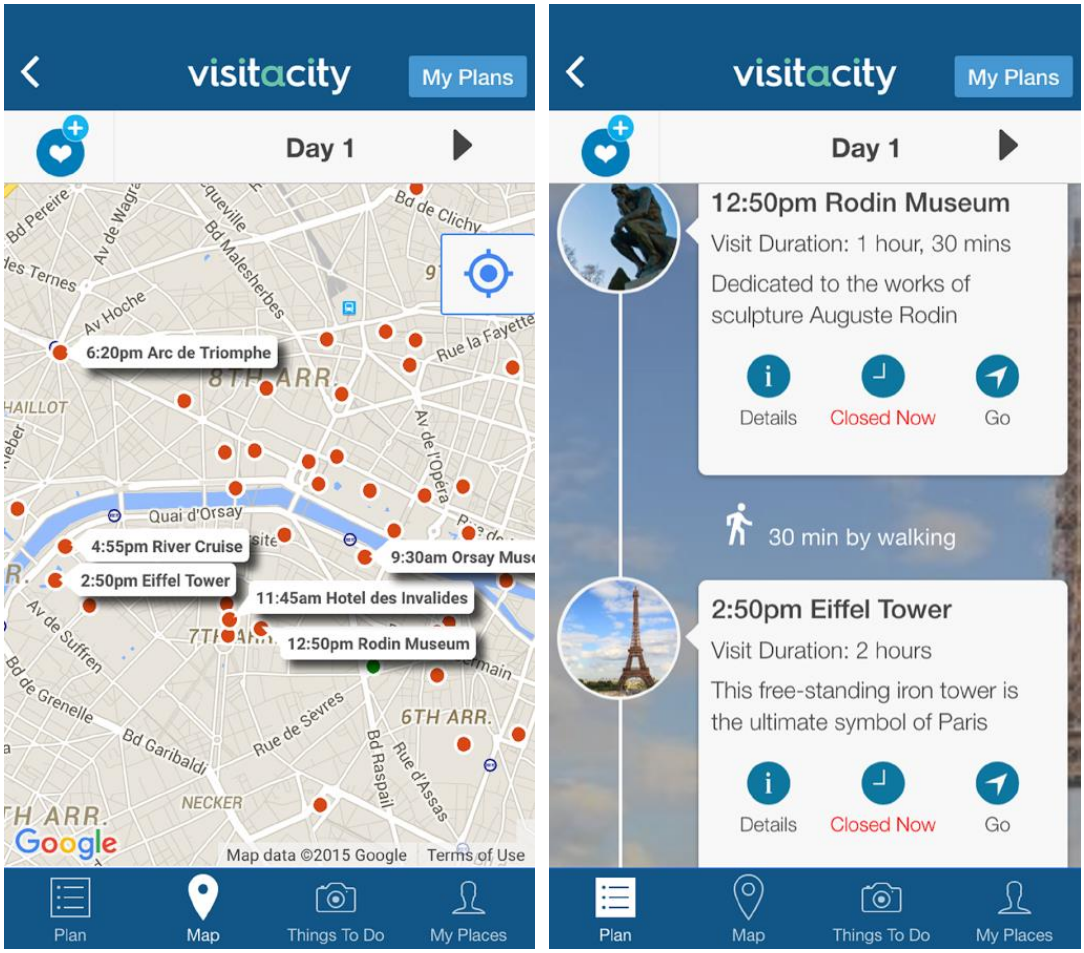


Figura 8 – Planeamento de itinerário na aplicação móvel do Visitacity em Android (Visitacity, n.d.)

2.2.5 Comparação das Soluções

Atualmente existem inúmeros trabalhos de pesquisa e produtos na área da recomendação turística. Pela pesquisa efetuada na literatura, o autor desta dissertação decidiu referir os quatro trabalhos apresentados nas secções anteriores. Alguns deles, à semelhança deste trabalho, resultam de investigações efetuadas no âmbito de sistemas de recomendação de turismo. Para estes casos, da pesquisa resultaram alguns protótipos que serviram de prova de conceito para fundamentar as conclusões dos seus autores. Cada uma das investigações teve uma abordagem diferente, tendo esses pontos de vista sido expostos em artigos científicos. No caso do Visitacity, trata-se de um produto final que se encontra disponível no mercado de forma gratuita para o utilizador. Por essa razão, os pormenores de implementação e as abordagens tomadas não foram revelados, sendo apenas alguns aspetos percetíveis pela análise do funcionamento do sistema. No entanto, como produto o Visitacity revela-se muito útil e apelativo para o fim a que se destina.

O City Trip Planner revela alguns aspetos interessantes a considerar para esta dissertação, nomeadamente a metodologia utilizada para o algoritmo de geração de rotas e a arquitetura idealizada, muito próxima da esperada para o sistema a desenvolver. Este sistema revela também outros aspetos semelhantes ao TheRoute como as parcerias realizadas com agências turísticas locais e o foco turístico apenas numa pequena área de turismo. No caso do City Trip Planner apenas eram consideradas como alvo turístico cinco cidades da Bélgica. O TheRoute tem como alvo, numa fase inicial do projeto, a zona norte de Portugal.

Das soluções encontradas, a que mais se equipara ao produto final a desenvolver é a Visitacity. Esta aplicação é bastante completa e possui muitas funcionalidades semelhantes às que se pretendem fornecer no TheRoute. No entanto, no que diz respeito à criação do itinerário a visitar, analisa poucos aspetos do contexto envolvente e não tem em consideração parâmetros como a personalidade e os gostos do utilizador. O planeamento é estático e limitado a três dias no máximo. Por abranger uma grande diversidade de centros turísticos em todo o mundo, esta aplicação não consegue expor com elevado rigor e detalhe informações atualizadas sobre todos os locais que apresenta. Um dos desafios do TheRoute é possuir a informação mais relevante para cada local, e sendo o mais atual possível. Com estas variáveis asseguradas serão também mais precisos os cálculos e ponderações utilizados para a geração e sugestão do trajeto a realizar.

2.3 Algoritmos para a Geração de Rotas

A escolha do algoritmo ou algoritmos a utilizar para a criação dos trajetos turísticos é um passo fundamental para a implementação do módulo proposto nesta dissertação. A geração de rotas tendo em conta diferentes restrições do meio, os atributos das atrações (horários de abertura e encerramento, duração da visita, valor de entrada, etc.) e os tempos de viagem pode ser um processo moroso se não forem utilizadas as técnicas e métodos mais adequados. Para resolver estes problemas de otimização são normalmente utilizados algoritmos baseados em heurísticas. O ideal para maximizar a satisfação dos turistas é a criação em tempo real do trajeto a ser realizado. As heurísticas são procedimentos que determinam soluções boas ou quase ótimas para um problema de otimização (Eiselt & Sandblom, 2000). Dado um problema computacional, uma heurística é essencialmente um algoritmo de pesquisa que procura a melhor solução para um problema de entre as possíveis hipóteses de resolução (Porumbel, 2012). Estes algoritmos usam recursos razoáveis e são capazes de produzir soluções aceitáveis, mas sem qualquer garantia teórica (Porumbel, 2012).

Atualmente existe uma enorme variedade de algoritmos baseados em heurísticas capazes de gerar caminhos entre diversos pontos. Alguns dos exemplos mais conhecidos são, entre outros, o *Branch-and-Bound*, *Hill-Climbing*, Pesquisa Tabu, *Simulated Annealing*, *Swarm Intelligence* e Algoritmos Genéticos. As secções seguintes apresentam uma breve introdução de cada um dos exemplos mencionados anteriormente, assim como as conclusões retiradas da análise de outros artigos científicos.

2.3.1 Branch-and-Bound

O *Branch-and-Bound* (B&B) é um método de pesquisa que enumera implicitamente todas as soluções possíveis para o problema em questão, armazenando as diversas soluções parciais numa estrutura em árvore (Morrison, Jacobson, Sauppe, & Sewell, 2016). Os nós que não são explorados na árvore geram filhos que dividem o espaço da solução em regiões menores que podem ser percorridas recursivamente (*branching*) (Morrison et al., 2016). São utilizadas regras para remover regiões do espaço de pesquisa que são comprovadamente abaixo do ideal (*bounding*) (Morrison et al., 2016). Depois de toda a árvore ter sido explorada, a melhor solução encontrada na pesquisa é retornada (Morrison et al., 2016).

A estratégia de pesquisa num algoritmo B&B determina a ordem na qual os nós não explorados são selecionados para visita (Morrison et al., 2016). A escolha da estratégia tem consequências potencialmente significativas na quantidade de tempo de computação necessária para a execução do algoritmo, bem como na quantidade de memória usada (Morrison et al., 2016). A pesquisa pode ser efetuada recorrendo a algoritmos como o primeiro em profundidade (*Depth-first search*), o primeiro em largura (*Breadth-first search*) e o primeiro o melhor (*Best-first search*), comummente conhecido como A*, cujas estratégias de exploração são exemplificadas na Figura 9. O nó tracejado é considerado a solução ótima e os números que se encontram na parte externa de cada nó indicam a sua ordem de exploração.

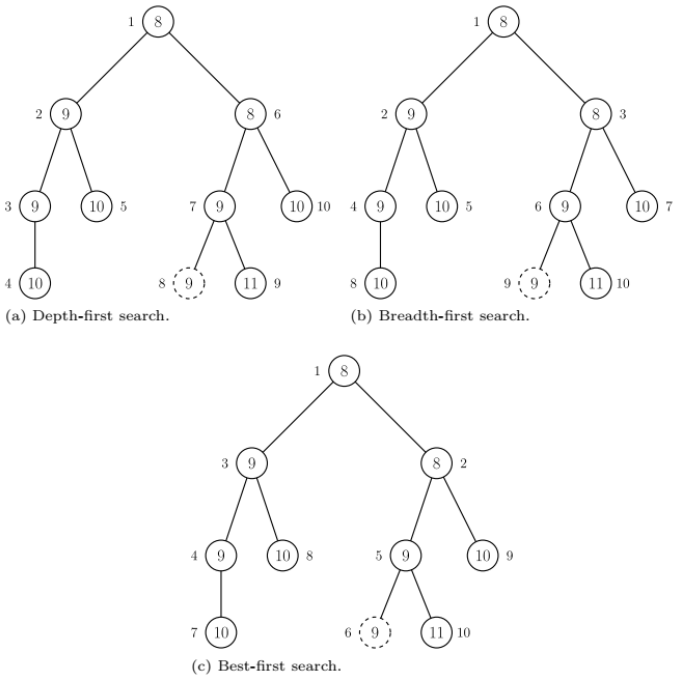


Figura 9 – Diferentes estratégias de pesquisa para o B&B
(Morrison et al., 2016)

Esta técnica é bastante eficaz, mas a sua complexidade de tempo geralmente é muito alta, podendo ser inaceitável para a resolução de alguns problemas (Kokash, 2005).

2.3.2 Hill-Climbing

O *Hill-Climbing* (HC) (Figura 10) é um algoritmo de pesquisa local que consiste num *loop* que se move continuamente na direção em que o valor aumenta (Russell & Norvig, 2009), daí a analogia com a subida de uma colina. O algoritmo tem como objetivo encontrar o melhor estado de acordo com uma função objetiva e termina quando atinge um pico com valor superior a qualquer um dos vizinhos. Se a elevação corresponder a um custo, o objetivo é encontrar o mínimo global, representado pelo vale mais baixo do gráfico; se a elevação corresponder a uma função objetiva, o objetivo é encontrar o pico mais alto, correspondente ao máximo global do gráfico (Russell & Norvig, 2009). À semelhança dos restantes algoritmos de pesquisa local, o HC não armazena uma árvore de pesquisa, ao deslocar-se apenas de um nó atual para os seus vizinhos. A estrutura de dados para o nó atual unicamente regista o estado e o valor da função objetivo para esse nó, não antecipando os vizinhos imediatos do estado atual (Russell & Norvig, 2009). Por estas razões, não é possível retroceder no algoritmo, ou seja, não se consegue voltar para o nó anterior.

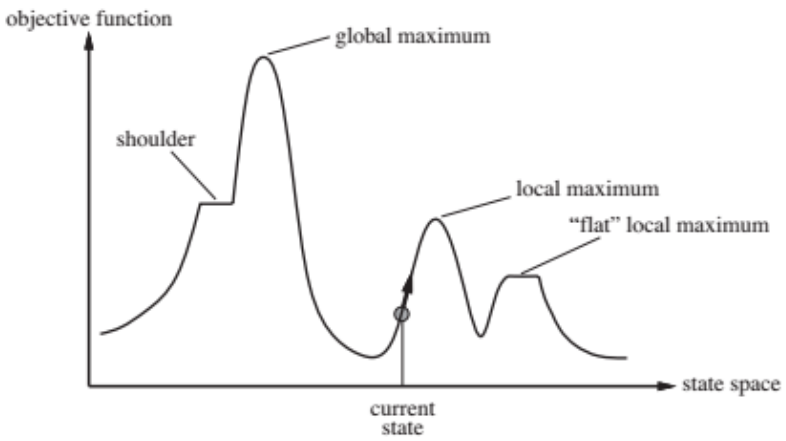


Figura 10 – Exemplo de uma execução de um algoritmo Hill-Climbing
(Russell & Norvig, 2009)

O HC consegue ser eficaz e ter um bom desempenho (Kokash, 2005). Por ser considerado uma pesquisa local gananciosa, encontra sempre os ótimos locais mais próximos sem precisar saber antecipadamente onde ir a seguir (Kokash, 2005). Os algoritmos de pesquisa local têm duas vantagens principais, nomeadamente quanto ao pouco e quase constante espaço de memória utilizado durante a sua execução; e o encontrar frequentemente soluções razoáveis em problemas que considerem um espaço de estados grande (Russell & Norvig, 2009). Por outro lado, o algoritmo HC pode ficar frequentemente preso ao encontrar:

- Máximos locais: um máximo local é um pico que é maior que cada um dos estados vizinhos, mas menor que o máximo global. O método tem tendência para ser atraído para um máximo local, em direção ao pico, mas depois fica preso sem conseguir navegar para outro local (Russell & Norvig, 2009);

- Cristas: uma crista resulta de uma sequência de máximos locais tornando muito difícil a navegação ao longo do espaço de estados (Russell & Norvig, 2009);

- Planaltos: um planalto é uma área plana representada ao longo do espaço de estados. Existem dois tipos de planaltos, um deles é o máximo local plano, a partir do qual não existe uma subida, o outro permite continuar a navegação e é conhecido por ombro (Russell & Norvig, 2009).

Ao longo dos anos têm sido desenvolvidas algumas variantes deste algoritmo, como é o caso do *Stochastic Hill-Climbing*, do *First-choice Hill-Climbing* e do *Random-restart Hill-Climbing* (Russell & Norvig, 2009). No entanto, nenhuma das suas variantes descritas se tem revelado completa porque frequentemente podem ficar presos nos máximos locais e não conseguirem encontrar uma solução (Russell & Norvig, 2009). Para uma execução bem-sucedida do algoritmo contribuem fatores como a existência de poucos máximos e planaltos locais. Outra variante do HC que é bastante popular no universo dos algoritmos de pesquisa é a Pesquisa Tabu (Russell & Norvig, 2009).

2.3.3 Pesquisa Tabu

A pesquisa Tabu é uma estratégia de pesquisa local concebida para evitar ótimos locais com o auxílio de estruturas de memória (Kokash, 2005). Esta abordagem utiliza uma estratégia de proibição para determinados movimentos ou estados, classificando-os em conformidade, de forma a impedir possíveis entradas em ciclo (Glover, 1986). Os movimentos que são marcados como "tabu" são geralmente uma pequena parte comparativamente a todos os possíveis, podendo torna-se disponíveis novamente após um período de tempo relativamente pequeno (Glover, 1986).

Este algoritmo armazena numa ou mais listas, denominadas listas tabu, os k movimentos efetuados nas sucessivas iterações que não podem voltar a ser visitados (Russell & Norvig, 2009). A função destas listas não é impedir que um movimento seja repetido, mas impedir que ele seja revertido (Glover, 1986). Desta forma consegue-se melhorar a eficiência na pesquisa, permitindo que sejam evitados alguns mínimos locais. Os elementos de uma lista tabu são geridos em função da ordem na qual são registados. Cada vez que um novo elemento é adicionado à lista, o elemento mais antigo da lista é removido. Assim, as listas tabu são geridas como listas circulares (Glover, 1986). O método procura o melhor movimento possível em cada etapa. Por esse motivo, o procedimento é inicialmente direcionado para um ótimo local (Glover, 1986). No entanto, esta condição não leva à interrupção da pesquisa, uma vez que o seu objetivo continua a ser a identificação do melhor movimento disponível (Glover, 1986).

2.3.4 Simulated Annealing

O *Simulated Annealing* (SA) é baseado na analogia entre a simulação do recozimento de sólidos na metalurgia e a dificuldade de resolver grandes problemas de otimização combinatória (van Laarhoven & Aarts, 1987). O recozimento é o processo usado para temperar ou endurecer metais e vidros. Neste processo o material é aquecido até atingir uma temperatura elevada e em seguida, é arrefecido gradualmente, o que lhe permite atingir um estado de baixa energia (Russell & Norvig, 2009). O SA resulta da combinação de um algoritmo HC com a possibilidade de efetuar transições entre estados de forma aleatória. Esta última particularidade contrasta com o HC que procura sempre o melhor movimento, em detrimento de um movimento aleatório (Russell & Norvig, 2009). Com esta possibilidade, o algoritmo evita ficar preso em máximos ou mínimos locais. Para simular o processo de arrefecimento, parte-se de um valor de temperatura alto, seguindo-se uma fase de arrefecimento lenta à medida que o algoritmo é executado. Enquanto a temperatura é elevada, o algoritmo permite com maior frequência maus movimentos, ou seja, aceita movimentar-se para uma solução pior que a atual (Russell & Norvig, 2009). Com a redução da temperatura o algoritmo vai-se tornando mais seletivo ao diminuir a probabilidade de ocorrência de maus movimentos. Um exemplo de execução do SA pode ser visualizado na Figura 11.

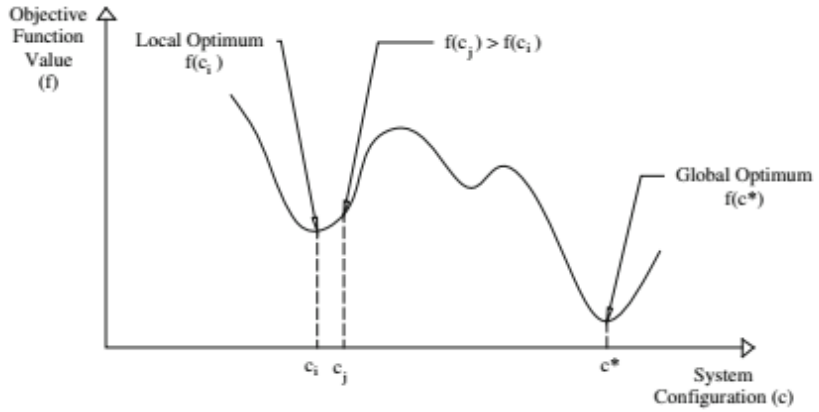


Figura 11 – Exemplo de uma execução do algoritmo *Simulated Annealing*
(Costa, Cunha, Coelho, & Einstein, 2016)

2.3.5 Swarm Intelligence

Os algoritmos de *Swarm Intelligence*, como o próprio nome indica, baseiam-se na análise da inteligência de enxames. São um conjunto de metodologias e técnicas de Inteligência Artificial sustentados pelo estudo do comportamento coletivo ou de grupos em sistemas descentralizados e auto-organizados (Kokash, 2005). "As unidades de processamento de informação que constituem um enxame podem ser animadas, mecânicas, computacionais ou matemáticas; podem ser insetos, pássaros ou seres humanos; podem ser elementos de uma matriz, robôs ou estações de trabalho independentes; podem ser reais ou imaginários"

(Kennedy, 2006). A principal vantagem das técnicas de inteligência de enxames é que são impressionantemente resistentes ao problema do ótimo local (Kokash, 2005). Os dois tipos mais populares de *Swarm Intelligence* são a otimização por enxame de partículas e a otimização por colónia de formigas, e são brevemente apresentados nas secções 2.3.5.1 e 2.3.5.2, respetivamente.

2.3.5.1 Otimização por Enxame de Partículas

A otimização por enxame de partículas, do inglês *Particle Swarm Optimization* (PSO), é uma metodologia baseada em padrões observados em comportamentos animais. Exemplos desses padrões são os movimentos dos bandos de pássaros e dos cardumes de peixes que, apesar de serem sincronizados, não possuem um mecanismo de controlo central. Por outras palavras, estes movimentos não são comandados por uma entidade central mas dependem antes de um grupo de indivíduos que interagem socialmente em função de um objetivo comum (Raamesh, 2013). Cada um dos indivíduos do grupo ou enxame é tratado como um ponto que se move num espaço de pesquisa, com o objetivo de encontrar a melhor solução para um problema. Inicialmente são espalhadas várias partículas com uma velocidade aleatória pelo espaço de pesquisa. Iterativamente, cada partícula ajusta o seu movimento tendo em conta o conhecimento da sua melhor posição e da melhor posição atual entre as restantes partículas (Raamesh, 2013). Com esta consciência sobre o posicionamento individual e do grupo, o movimento do grupo tende a encontrar o ótimo global no espaço de pesquisa para o problema definido. A Figura 12 exemplifica o processo de otimização por enxame de partículas para um bando de pássaros.

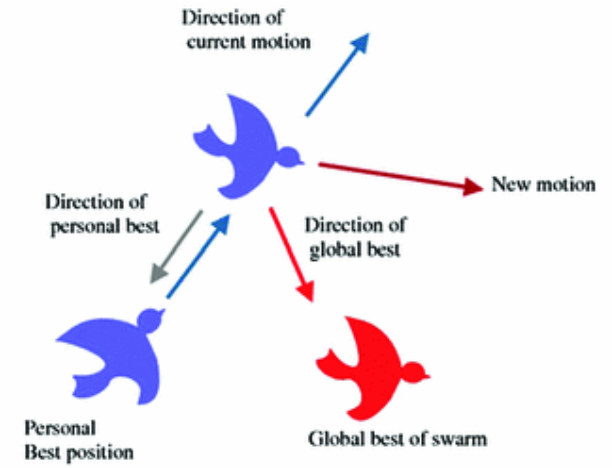


Figura 12 – Representação esquemática da atualização do movimento de uma partícula (Esmin, Coelho, & Matwin, 2015)

2.3.5.2 Otimização por Colónia de Formigas

A otimização por colónia de formigas, do inglês *Ant Colony Optimization* (ACO), é uma metodologia inspirada no comportamento de algumas espécies de formigas ao saírem do seu formigueiro para encontrarem alimento. No decorrer do percurso efetuado, as formigas

depositam feromonas no solo para marcar um caminho favorável que deve ser seguido pelos restantes membros da colónia (Dorigo, Birattari, & Stützle, 2006). Ao sentirem a presença das feromonas libertadas, as formigas tendem a seguir os caminhos onde essa concentração hormonal é mais elevada. Através deste mecanismo, as formigas são capazes de transportar alimentos para o formigueiro de uma forma extraordinariamente eficaz (Dorigo et al., 2006). Inicialmente, cada formiga escolhe aleatoriamente um dos caminhos para superar o obstáculo. No entanto, ao fim de algumas escolhas aleatórias, um dos caminhos começa a apresentar uma maior concentração de feromonas que os restantes, tornando-o mais atraente. As formigas que optam pelo trajeto mais curto são as que mais rapidamente fazem o circuito, depositando mais feromona, fazendo a colónia optar por esse caminho.

O processo ACO anteriormente explicado pode ser consultado na Figura 13, onde são ilustradas as fases que ocorrem desde a saída do formigueiro até à chegada à fonte de alimento. Numa primeira fase (a), as formigas seguem um caminho entre o formigueiro e uma fonte de alimento, até que aparece um obstáculo no caminho (b). As formigas podem optar por seguir pela esquerda ou para a direita com igual probabilidade. A feromona é depositada mais rapidamente no caminho mais curto (c), fazendo com que todas as formigas acabem por escolher essa solução (d) (Yousefikhoshbakht, Didehvar, & Rahmati, 2013).

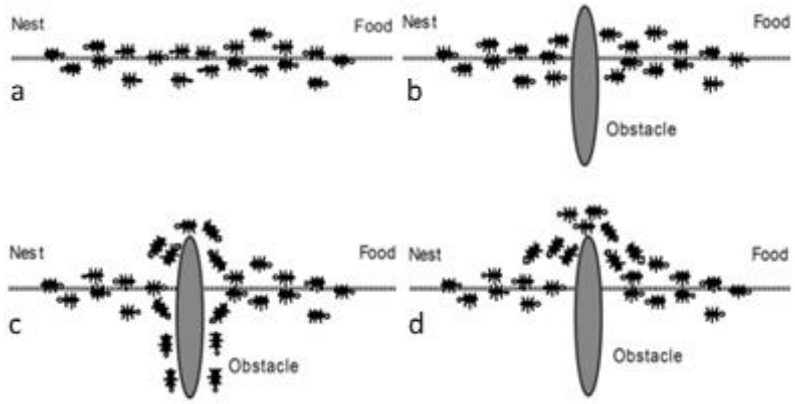


Figura 13 – Comportamento das formigas na pesquisa de alimento
(Yousefikhoshbakht et al., 2013)

2.3.6 Algoritmos Genéticos

Os algoritmos genéticos têm como objetivo desenvolver uma população de soluções candidatas a um determinado problema, ao utilizar operadores inspirados na variação genética e na seleção natural (Mitchell, 1996). Com esta metodologia, estes algoritmos comportam-se como uma abstração da evolução das espécies que ocorre na natureza, ao selecionar os estados mais adequados ao problema em questão.

O conjunto de fases e respetivos estados na execução de um algoritmo genético podem ser visualizados no exemplo ilustrado na Figura 14, sendo as diversas fases descritas em seguida. O processo de evolução começa a partir de um determinado conjunto de estados gerados

aleatoriamente (a), chamados de população (Russell & Norvig, 2009). Cada estado ou indivíduo, é representado por um conjunto de símbolos pertencentes a um alfabeto finito (Russell & Norvig, 2009), geralmente por combinações de *bits* (0 e 1). Cada individuo é avaliado pela função de aptidão (b), que deve retornar valores mais altos para melhores estados (Russell & Norvig, 2009). Esta avaliação proporciona uma medida de desempenho em relação a um conjunto particular de parâmetros, sendo independente para cada indivíduo. A probabilidade de um indivíduo ser selecionado para reprodução (c) é diretamente proporcional à sua pontuação de aptidão, em relação à pontuação dos restantes indivíduos da população. No exemplo apresentado, foram selecionados aleatoriamente dois pares para reprodução, tendo em conta as probabilidades previamente calculadas. Em cada par de progenitores a ser cruzado é escolhido, de forma aleatória, um ponto de corte nos seus cromossomas. Em (c), os pontos de corte são aplicados após o terceiro dígito (gene) no primeiro par de progenitores e após o quinto dígito no segundo par.

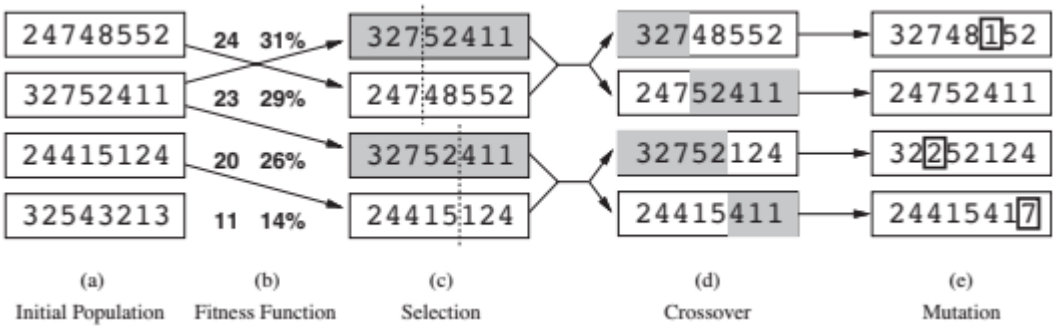


Figura 14 – Fases de um algoritmo genético
(Russell & Norvig, 2009)

Os descendentes são criados através de cruzamentos (d) entre sequências de genes dos progenitores, efetuados no ponto de corte (Russell & Norvig, 2009). No exemplo apresentado, o primeiro descendente do primeiro par de progenitores obtém os três primeiros genes do primeiro progenitor e os restantes do segundo progenitor. Por sua vez, o segundo descendente obtém os três primeiros genes do segundo progenitor e os restantes do primeiro. Por fim, cada gene fica sujeito a uma mutação aleatória (e), tendo uma pequena probabilidade independente associada (Russell & Norvig, 2009). Como é possível verificar na Figura 14, ocorreram mutações em três dos quatro descendentes criados.

A operação de cruzamento revela-se bastante vantajosa na criação de indivíduos com melhores aptidões, fugindo desse modo a máximos locais do problema. Por outras palavras, com o conceito de mutação é possível introduzir um fator adicional de diversidade na população.

2.3.7 A*

O A*, pronunciado A Star, é um algoritmo de pesquisa informada usualmente utilizado para executar a procura do caminho com o menor custo entre um nó de partida e um nó de

destino (Zeng & L. Church, 2009). É uma combinação de diferentes heurísticas como a pesquisa em largura *Breadth-First-Search* (BFS) e o algoritmo de Dijkstra[1]. Adicionalmente, este algoritmo faz uso de uma função heurística que avalia o custo de uma transição entre o estado atual e os estados vizinhos até chegar ao estado de destino. Esta função tem uma grande influência na qualidade da solução encontrada, uma vez que é a responsável por encaminhar a pesquisa na direção mais adequada ao problema (Yao, Lin, Xie, Wang, & Hung, 2010).

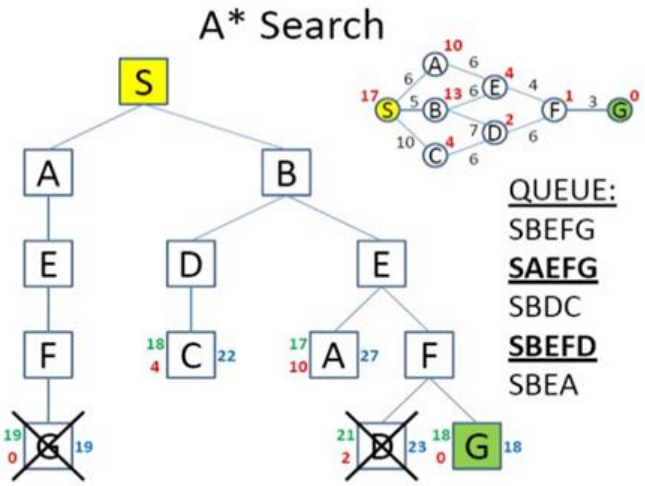


Figura 15 – Pesquisa A*
(Kashyap, 2017)

Na ilustração acima apresentada é possível observar um exemplo muito simples do algoritmo A*. O grafo é constituído por oito nós e pretende-se encontrar o caminho mais curto entre o nó S e o nó G. Ao longo da execução do algoritmo é utilizada uma fila de prioridades (*priority queue*) para armazenamento dos nós candidatos (nós não visitados) para a expansão da pesquisa, sendo por isso conhecida como "*open set*" (Zeng & L. Church, 2009). A cada iteração do algoritmo, o nó com o valor mais baixo é removido da fila, sendo posteriormente explorados os seus nós vizinhos. Caso o valor do novo nó a explorar seja favorável à função heurística utilizada, esse nó é adicionado à fila para que a sua vizinhança seja explorada em futuras iterações. O algoritmo continua a sua exploração até que encontra o nó de destino que tenha um custo inferior a qualquer nó na fila, ou até que já não existam mais nós na fila para explorar.

O A* é um algoritmo versátil, uma vez que pode ser utilizado para resolver uma grande variedade de problemas com diferentes graus de complexidade. No entanto, dependendo do âmbito e da dimensão do problema, o algoritmo A* pode apresentar uma pesquisa lenta. A função heurística utilizada também pode condicionar a velocidade de execução do algoritmo.

[1] O algoritmo de Dijkstra é utilizado para encontrar o caminho mais curto entre dois nós ou vértices num grafo cujas arestas tenham peso maior ou igual a zero (Mehlhorn & Sanders, 2008).

2.3.8 Outras Abordagens

As abordagens tradicionais apresentadas ao longo da secção 2.3 são amplamente utilizadas em diversos sistemas cujo objetivo é encontrar o caminho que melhor satisfaça as condicionantes para um determinado problema. Como resultado da pesquisa efetuada, foram encontrados alguns artigos científicos que apresentam soluções algorítmicas interessantes para problemas relacionados com a recomendação de turismo. Em seguida, são apresentados dois desses trabalhos, que escolheram abordagens diferentes na definição do(s) algoritmo(s) a utilizar nos seus sistemas de recomendação de turismo.

Os investigadores (Garcia, Arbelaitz, Linaza, Vansteenwegen, & Souffriau, 2010) optaram por aplicar uma abordagem híbrida ao combinar duas heurísticas diferentes. No seu trabalho havia a necessidade de gerar rotas em tempo real e incluir a possibilidade de utilização de transportes públicos, o que complicou a abordagem ao problema. Com esta condicionante, cada cálculo para o caminho mais curto entre dois pontos de interesse passou a depender do fator tempo. Um pequeno atraso na saída de um ponto de interesse pode causar um aumento significativo no tempo de chegada ao seguinte. Por exemplo, quando um turista perde o autocarro e decide esperar pelo próximo ou caminhar para o próximo ponto de interesse (Garcia et al., 2010). Foi implementado um algoritmo de Dijkstra modificado para encontrar o caminho mais curto dependente do tempo, de forma a suportar o requisito da utilização dos transportes públicos. Os tempos médios de viagem calculados são depois armazenados numa base de dados para poderem ser posteriormente reutilizados sem ser necessário um novo cálculo. A outra heurística utilizada foi a pesquisa local iterada, que tem por objetivo inserir novas visitas a uma rota e atribuir-lhe uma pontuação. Essa pontuação depende diretamente do grau de satisfação do turista em relação a um ponto de interesse, assim como do tempo necessário para o poder visitar. Entre esses pontos, a heurística seleciona aqueles com a maior pontuação, repetindo o processo até que não seja possível inserir mais locais (Garcia et al., 2010).

Outro artigo encontrado na literatura foi o dos investigadores (Sun & Lee, 2004), onde é proposto um sistema de recomendação de turismo baseado na integração de funções de análise espacial com um algoritmo heurístico. Na sua solução, Sun e Lee utilizam um modelo baseado em vetores, que é usado para representar os interesses pessoais dos turistas, assim como os recursos turísticos disponíveis (Sun & Lee, 2004). Ao contrário dos autores apresentados anteriormente, estes investigadores não utilizaram o algoritmo de Dijkstra, por entenderem que o caminho mais curto nem sempre é o que o turista pretende. Em vez disso, decidiram dar prioridade aos gostos do turista, tendo em conta distância e tempo que este pretende gastar no trajeto. Estes fatores são normalmente expressos como um limite superior que não pode ser excedido, mas que devem ser utilizados até valores próximos desse limite (Sun & Lee, 2004), ou seja, a rota gerada deve aproximar-se o mais possível do tempo e da distância máximos que o turista impôs. Para superar estas imposições, foi adotado uma variante da pesquisa Tabu, conhecido como algoritmo de extensão e colapso (Sun & Lee, 2004). É um método de pesquisa heurística que seleciona cada ponto turístico com base no

seu valor de atração para o turista e no seu valor de custo, de forma a maximizar o tempo e distância impostos pelo utilizador.

2.3.9 Considerações sobre as Abordagens Estudadas

Durante a pesquisa e o estudo efetuado aos diversos algoritmos abordados, foram encontradas algumas particularidades que podem ser esclarecedoras quanto à viabilidade da sua utilização. De seguida são apresentados alguns dos principais tópicos constatados:

- O *Branch-and-Bound* revela-se eficaz para problemas simples, quando o espaço de pesquisa não é muito extenso. Por outro lado, possui uma complexidade temporal elevada (Kokash, 2005), podendo ser inaceitável para a resolução de problemas que exijam uma resposta rápida;

- O *Hill-Climbing* consegue ser eficaz e ter um bom desempenho (Kokash, 2005). Geralmente utiliza de forma constante pouca memória durante a sua execução e encontra frequentemente soluções razoáveis em problemas com um vasto espaço de estados (Russell & Norvig, 2009). Tem o inconveniente de poder ficar preso em máximos locais, cristas e planaltos, e não conseguir encontrar uma solução (Russell & Norvig, 2009);

- A Pesquisa Tabu consegue escapar a ótimos locais ao permitir avanços para estados menos favoráveis (Glover, Martí, & Laguna, 2007). Por estar dependente da definição de um vasto conjunto de parâmetros, a sua má configuração pode levar à não obtenção do ótimo global (Abdmouleh, Gastli, Ben-Brahim, Haouari, & Al-Emadi, 2017). O número de iterações efetuadas para encontrar uma solução pode ser muito elevado (Abdmouleh et al., 2017), o que regra geral implica mais tempo de execução;

- O *Simulated Annealing* pode fornecer boas soluções para muitos problemas combinatórios e consegue evitar ficar preso em máximos ou mínimos locais (Russell & Norvig, 2009). No entanto, é necessário ter especial cuidado com a definição dos seus parâmetros. Uma má configuração inicial pode levar a que a pesquisa termine prematuramente ou se estenda por tempo indeterminado;

- Os algoritmos baseados em *Swarm Intelligence* têm a vantagem de conseguir efetuar pesquisa em paralelo num espaço de pesquisa (Abdmouleh et al., 2017). Os vários agentes que constituem o "enxame" são indivíduos independentes e conseguem contribuir em simultâneo na procura de uma solução. Em algumas implementações a distribuição de probabilidades que avalia a mudança entre estados pode alterar a cada iteração (Abdmouleh et al., 2017), o que dificulta uma análise posterior às decisões tomadas;

- Os algoritmos genéricos são adequados para problemas complexos e menos bem definidos (Abdmouleh et al., 2017). Por outro lado, o seu tempo de execução pode ser elevado e não é garantida a qualidade da solução (Abdmouleh et al., 2017);

- O algoritmo A* é versátil e pode ser utilizado numa vasta gama de problemas de pesquisa. Durante a sua exploração, expande um nó apenas se for promissor para a heurística considerada. Pode facilmente utilizar heurísticas de outros algoritmos de pesquisa e alterar a maneira como avalia a transição entre cada nó. Possui algumas desvantagens ao ser exigente em termos de memória e processamento. Como mantém algumas listas em memória, estas podem atingir tamanhos excessivamente grandes e causar problemas na gestão de memória.

As considerações realizadas para cada uma das abordagens estudadas servem para ter uma ideia das garantias e dos problemas que cada abordagem algorítmica pode trazer. No entanto, só é possível perceber a que mais se adequa ao problema em questão por via de tentativas de adaptação e experimentação das soluções obtidas.

3 Análise e Design

Neste capítulo são descritas as fases de análise e design para o objeto desta dissertação. A secção 3.1, é a responsável pela apresentação de uma análise mais técnica ao problema relacionado com a criação de rotas turísticas tendo em conta o contexto. Na seção 3.2 são expostos os diagramas representativos da arquitetura do sistema existente, assim como da arquitetura do módulo a desenvolver. Por fim, a secção 3.3 revela a realização do caso de uso para a criação de uma rota personalizada de acordo com os gostos do utilizador e com o contexto da viagem.

3.1 Análise ao Problema

Como já mencionado anteriormente, a área do turismo tem sofrido enormes evoluções ao longo das últimas décadas. O crescimento desta área de negócio tem sido fortemente potenciado pelo desenvolvimento tecnológico. A necessidade de oferecer soluções tecnológicas direcionadas ao turismo tem também fomentado a investigação nas áreas do planeamento e criação de rotas turísticas.

Ao longo desta secção é analisado o problema a resolver nesta dissertação, recorrendo a um ponto de vista mais técnico. Inicialmente é introduzido o problema do caixeiro viajante e as suas semelhanças com a criação de rotas turísticas. Seguidamente é apresentado o método escolhido para a criação e seleção do perfil do utilizador, que possui um papel bastante importante na seleção dos pontos de interesse a apresentar ao turista. São também apresentados os conceitos de negócio relevantes ao projeto TheRoute, assim como algumas das novas ideias a implementar. Estes conceitos são representados e explicados com recurso a um modelo de domínio.

3.1.1 Problema do Caixeiro Viajante

O problema do caixeiro viajante, do inglês *Travelling Salesman Problem* (TSP), é um conhecido problema de otimização combinatória estudado em áreas como as ciências da computação, algoritmia e pesquisa operacional (Alhanjouri & Alfarra, 2011). O TSP é um problema polinomial difícil (NP-hard) não determinístico (Mukhairez & Maghari, 2015), que consiste numa visita por um determinado conjunto de cidades, por parte do caixeiro viajante. Cada deslocação entre cidades tem associada uma distância ou peso. O caixeiro é obrigado a passar por todas as cidades e a voltar à cidade de onde iniciou a viagem (Mukhairez & Maghari, 2015). Como o caixeiro não pretende gastar muito tempo na viagem, tenta encontrar a sequência de cidades que lhe permita minimizar a distância percorrida (Alhanjouri & Alfarra, 2011). Por outras palavras, é pretendido o caminho mais curto que possibilite visitar todas as cidades e voltar à cidade de partida. As cidades e os respetivos custos para as visitar podem ser representados através de um grafo não direcionado (caminhos bidirecionais), como é exemplo o grafo da Figura 16.

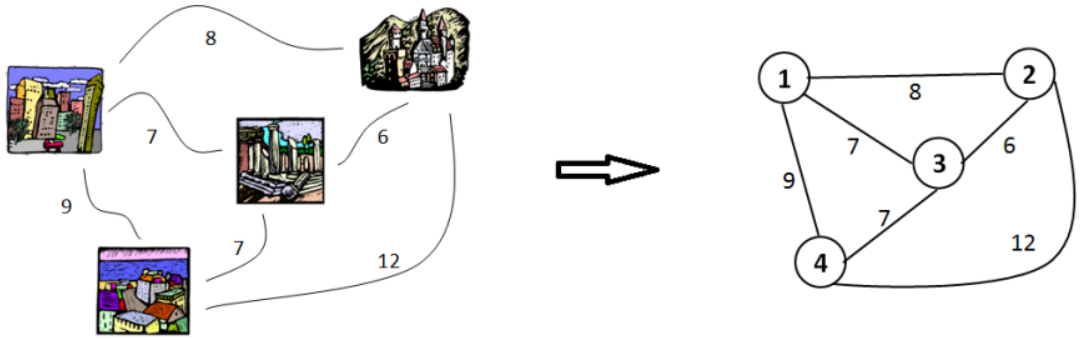


Figura 16 – Problema do caixeiro viajante
(Hong Kong, n.d.)

O problema em estudo nesta dissertação é bastante semelhante ao do caixeiro viajante, com a diferença de existirem algumas restrições que diminuem o número de combinações possíveis. Em vez de cidades, são considerados pontos de interesse, o que aumenta ainda mais o número de combinações possíveis, uma vez que cada cidade tem um conjunto de pontos de interesse. O ponto de interesse inicial é o mais próximo da localização atual do turista, ou da posição escolhida geograficamente pelo mesmo. De seguida, é necessária uma pesquisa exploratória no grafo até atingir o fim do horário de visita definido pelo utilizador. Para descobrir o caminho que mais se adequa aos gostos do turista tem de se maximizar a soma dos interesses dos pontos de interesse a visitar, não sendo, por isso, obrigatório que o melhor caminho seja o mais curto. Se existirem N pontos de interesse e se quiser calcular todos os caminhos possíveis entre eles, o número de combinações possíveis é $N!$ (N fatorial). Por exemplo, para apenas 100 pontos de interesse existem 100! combinações, o que se traduz em aproximadamente $9{,}33*10^{157}$ possibilidades. Este número de combinações é bastante elevado, mesmo para a capacidade de processamento que existe nos dias de hoje. Dependendo da complexidade do problema, cálculos deste género podem demorar muitos

anos até se conseguir calcular todas as sequências possíveis. Por estas razões, o problema a abordar é NP-hard, sendo impossível chegar a uma solução ideal em tempo razoável para um elevado número de pontos de interesse. No entanto, com adaptações das heurísticas mencionadas no estado da arte juntamente com algumas técnicas de otimização é possível chegar a uma solução próxima da ótima.

3.1.2 Perfil de Utilizador

Um dos principais objetivos do TheRoute é fornecer ao turista um produto tecnológico capaz de lhe proporcionar uma experiência de lazer compatível com os seus gostos. Desta forma, um dos aspetos importantes ao contexto da rota é o perfil e a personalidade do seu utilizador.

Para aferir a personalidade e os gostos do turista, após a sua primeira autenticação com sucesso na aplicação móvel, este deve responder a um questionário constituído por dez perguntas. Esse questionário é baseado num conjunto de imagens que o utilizador tem de classificar o quão se adequam ao seu perfil, numa escala de zero a cinco. Adicionalmente, o utilizador deve classificar algumas imagens relativas a determinada categoria turística, de modo a ser possível complementar a aferição das suas preferências. As suas respostas permitem a construção de um perfil de utilizador, tendo em conta o *Big Five Personality Inventory* (Ramos et al., 2019). Trata-se de um modelo que, como o próprio nome indica, tem em consideração cinco aspetos ou dimensões psicológicas da personalidade humana:

- **Abertura à Experiência** (*openness to experience*) – indica o grau de abertura de uma pessoa relativamente à experiência de novas realidades. Pessoas com grande abertura para a novidade são geralmente mais criativas, aventureiras, curiosas e artísticas. Por outro lado, os que possuem um baixo grau de abertura são considerados mais convencionais e encontram maior conforto no ambiente familiar (Zopiatis & Constanti, 2012);

- **Conscienciosidade** (*conscientiousness*) – é uma medida de confiabilidade, na qual um individuo muito consciente tem uma personalidade responsável, organizada, confiável e persistente. Quem tem uma pontuação baixa nesta dimensão revela ser desorganizado, distraído e não confiável (Zopiatis & Constanti, 2012);

- **Extroversão** (*extraversion*) – indica o nível de conforto relativamente ao relacionamento interpessoal. Os extrovertidos tendem a ser mais assertivos e sociais, enquanto que os introvertidos tendem a ser mais tímidos, quietos e reservados (Zopiatis & Constanti, 2012);

- **Neuroticismo ou Estabilidade Emocional** (*neuroticism*) – mede a capacidade de uma pessoa suportar situações de elevada pressão e de desenvolver emoções negativas. Indivíduos com uma estabilidade emocional positiva tendem a ser calmos, autoconfiantes e seguros. Pelo contrário, indivíduos com estabilidade emocional baixa tendem a ser nervosos, ansiosos, deprimidos e inseguros (Zopiatis & Constanti, 2012);

- **Agradabilidade** (*agreeableness*) – identifica a capacidade que um indivíduo tem em cooperar com os outros. Aqueles que demonstram uma alta pontuação nesta dimensão são cooperativos, afetuosos e confiantes. Os que possuem uma pontuação mais baixa têm tendência a ser frios e desagradáveis (Zopiatis & Constanti, 2012).

Tendo em conta as dimensões psicológicas apresentadas, é possível conjugar algumas correspondências entre o comportamento humano e as atividades de lazer que mais se adequam ao seu perfil. Por exemplo, quem possui uma elevada abertura a novas experiências poderá estar mais inclinado a testar desportos radicais ou até mesmo a visitar monumentos históricos ou exposições de arte (Ramos et al., 2019). Com base nas respostas do utilizador é possível traçar uma eventual correspondência entre os seus gostos e as categorias turísticas existentes no sistema. Tendo em conta que cada ponto de interesse pode estar associado a uma ou mais categorias, após o cálculo do grau de compatibilidade do utilizador com uma categoria, pode-se aferir a sua intenção em visitar um determinado ponto de interesse.

3.1.3 Modelo de Domínio

O Modelo de Domínio é uma representação visual das classes conceptuais ou de objetos reais do domínio de interesse, assim como as associações resultantes das suas interações.

Como tem sido referido, o módulo de geração de rotas a desenvolver complementa uma plataforma já existente. Desta forma, o seu domínio e negócio já se encontrava previamente definido, razão pela qual grande parte das classes utilizadas neste projeto já existiam. Estas classes são importantes para a perceção dos conceitos de negócio e modelação dos objetos intervenientes relevantes para a criação de rotas turísticas. Por se tratar de um modelo de negócio um pouco extenso e nem todo ele relacionado à criação de rotas, são apenas apresentadas as classes necessárias à realização deste projeto, que se encontram representadas no modelo de domínio apresentado na Figura 17.

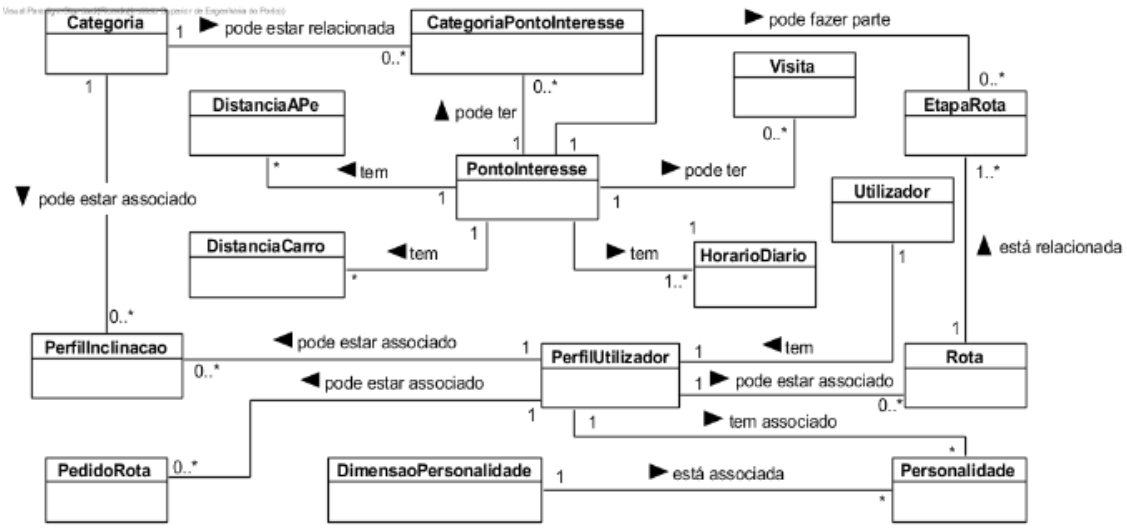


Figura 17 – Modelo de Domínio

As classes mais importantes no âmbito deste trabalho são os utilizadores, os pontos de interesse e as rotas geradas, uma vez que é em torno delas que se desenrola toda a ação relativa ao negócio deste projeto. O utilizador será geralmente um turista ou um viajante que deseja visitar pontos de interesse próximos a uma localização, que poderá ser a sua localização atual ou uma definida por si.

Quando o utilizador se regista no sistema é criado um modelo de utilizador contendo os dados de acesso ao sistema e um perfil que contém os seus dados pessoais, como o nome e data de nascimento. Após o seu registo é apresentado um inquérito, como o explicado no ponto 3.1.2, e calculadas as suas preferências tendo em conta as dimensões de personalidade. É também avaliada a sua predisposição relativamente a cada uma das categorias existentes no sistema. Por sua vez, a mesma categoria pode estar presente em diversos pontos de interesse. O ponto de interesse, além de poder estar relacionado com várias categorias, tem pelo menos um horário diário de funcionamento. Também relacionadas aos pontos de interesse estão as suas visitas que foram registadas no sistema, bem como as várias distâncias a percorrer, a pé ou de carro, entre os restantes pontos de interesse do sistema.

Cada vez que o utilizador faz um pedido de uma rota personalizada ao sistema, é registado um modelo representativo que fica associado ao perfil do utilizador. Após a concretização desse pedido é gerada uma rota com todos os dados relevantes, nomeadamente o seu tempo de início, distância, duração e ponto de partida. Relacionadas com esta rota estão também as várias etapas que a constituem. Por etapas entendem-se as deslocações entre dois pontos de interesse.

Os modelos representados não contemplam utilizadores com limitações físicas (mobilidade reduzida, entre outras), assim como a cobertura das suas necessidades específicas por parte dos pontos de interesse. Não foram contemplados, uma vez que ainda não foi determinada a forma de aferir essas limitações físicas do indivíduo. No entanto, é trabalho futuro incluir esta funcionalidade de forma a tornar a experiência mais adaptada a estes casos. Outras classes como as responsáveis pela criação do perfil do utilizador também não se encontram representadas no modelo de domínio, uma vez que apenas o seu resultado é relevante para o problema em questão.

3.2 Design da Arquitetura do Sistema

O sistema TheRoute, à semelhança de outras aplicações com um âmbito idêntico, possui uma arquitetura monolítica disposta por camadas, seguindo o princípio da responsabilidade única (Ramos et al., 2019), conhecido por *Single-Responsability Principle*. Segundo este princípio cada módulo ou classe deve ter responsabilidade sobre uma única parte da funcionalidade fornecida pelo *software*, encapsulando essa responsabilidade (Martin & Martin, 2007). Tendo esse princípio em consideração, de seguida são apresentadas duas secções com a arquitetura atual do sistema e a arquitetura do módulo a desenvolver.

3.2.1 Arquitetura Existente

A arquitetura por camadas do TheRoute é constituída na sua camada inferior por uma *Data Access Layer* (DAL), que trata das operações de CRUD com a base de dados e que ao mesmo tempo se comporta como uma camada de serviço, fornecendo dados às restantes camadas do sistema (Ramos et al., 2019). Ao obrigar as restantes camadas do sistema a consultar a DAL, reforça-se a consistência e a concordância dos dados (Ramos et al., 2019). A DAL é privada de qualquer lógica de negócio, tendo essa responsabilidade sido delegada inteiramente em cada um dos sistemas envolventes (Ramos et al., 2019). O módulo de geração de rotas contemplará todos os métodos e algoritmos necessários para fornecer as principais funcionalidades do sistema. Os algoritmos a utilizar recebem como entrada vários parâmetros provenientes dos dispositivos dos utilizadores, como são exemplo a disponibilidade de tempo do utilizador, o seu método de transporte preferido, entre outros (Ramos et al., 2019). O trajeto gerado terá de refletir as considerações necessárias tendo em conta estes parâmetros de entrada ou requisitos do utilizador e do meio. Este sistema também possui alguns algoritmos para a criação de rotas ajustadas em grupo, respeitando as necessidades e preferências pessoais de cada indivíduo (Ramos et al., 2019).

O *website* para os gestores de conteúdo tem como objetivo a adição de dados relevantes ao funcionamento da plataforma e é direcionado para os agentes turísticos (Ramos et al., 2019). Neste caso, os conteúdos são quaisquer dados relacionados com os pontos de interesse suportados pelo sistema, como é o caso das categorias em que se insere e dos seus detalhes internos (Ramos et al., 2019). Existem ainda duas camadas superiores, o *website* e a aplicação móvel, que contactam diretamente com os dispositivos utilizados pelo turista. É possível o utilizador registar-se no sistema através destes dois componentes, sendo necessário o preenchimento inicial de um formulário. Nesse formulário são expostas algumas perguntas ao turista que visam modelar alguns aspetos da sua personalidade e preferências culturais (Ramos et al., 2019).

A aplicação móvel permite ao utilizador ver os seus itinerários e rotas gerados através do sistema, com a apresentação de um mapa e de um plano de viagem. A arquitetura descrita pode ser verificada através do diagrama de componentes ilustrado na Figura 18.

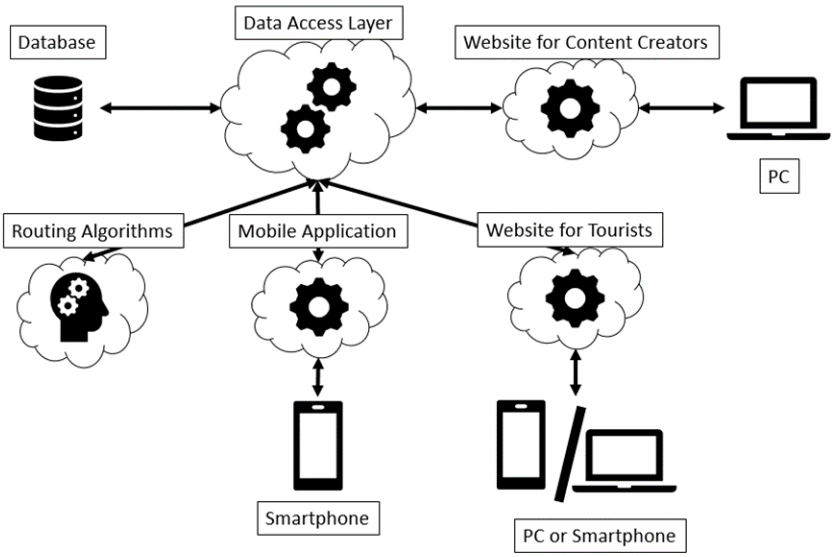


Figura 18 – Diagrama de componentes do sistema TheRoute
(Ramos et al., 2019)

3.2.2 Arquitetura do Módulo de Criação de Rotas

No seguimento da anterior descrição, a arquitetura do módulo a desenvolver nesta dissertação pode ser consultada na Figura 19 com recurso a um diagrama de componentes.

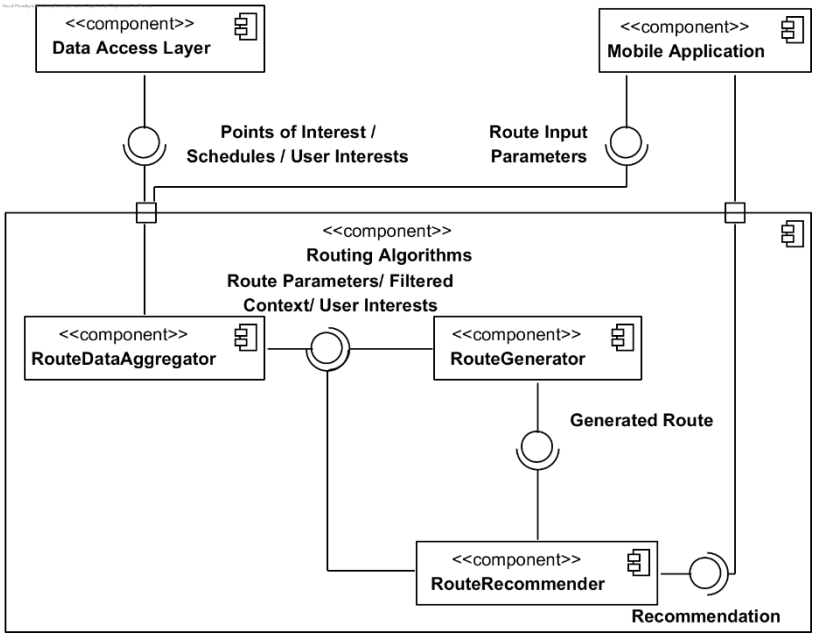


Figura 19 – Diagrama de componentes do módulo de geração de rotas

O módulo ou componente de criação de rotas não se encontra idealizado de uma forma convencional, uma vez que os seus principais constituintes se encontram distribuídos. Quando

for apresentado o diagrama de implantação, será explicado com maior detalhe o tipo de distribuição seguida. Com esta ideia em mente, o componente de criação de rotas recebe como entrada os diversos parâmetros necessários à criação da rota, como é o caso dos dados provenientes da aplicação móvel (identificador do utilizador, coordenadas de início da viagem, dia da semana, modo de deslocação, horas de início e fim da viagem) e dos dados provenientes da DAL que contêm os pontos de interesse e os horários válidos de acordo com os *inputs* da aplicação. Todos estes dados dão entrada num componente agregador, que modela os dados passados por parâmetro para objetos aceites pelo gerador de rotas. Ao ter acesso a esta informação, o componente responsável pela criação das rotas é capaz de gerar um caminho com o auxílio de heurísticas. Por fim, existe um componente que analisa o caminho gerado e cria uma recomendação de rota com o respetivo planeamento de viagem, que é posteriormente consultada por outros componentes do sistema e apresentada ao utilizador.

Para uma melhor perceção da distribuição idealizada, o diagrama de implantação representado na Figura 20 ilustra os nós que compõem o sistema e os seus principais componentes, assim como as comunicações que realizam entre si. A disposição dos componentes pode ser diferente da apresentada, não sendo por isso vinculativa. No entanto, o diagrama referido mostra uma possível configuração dos componentes.

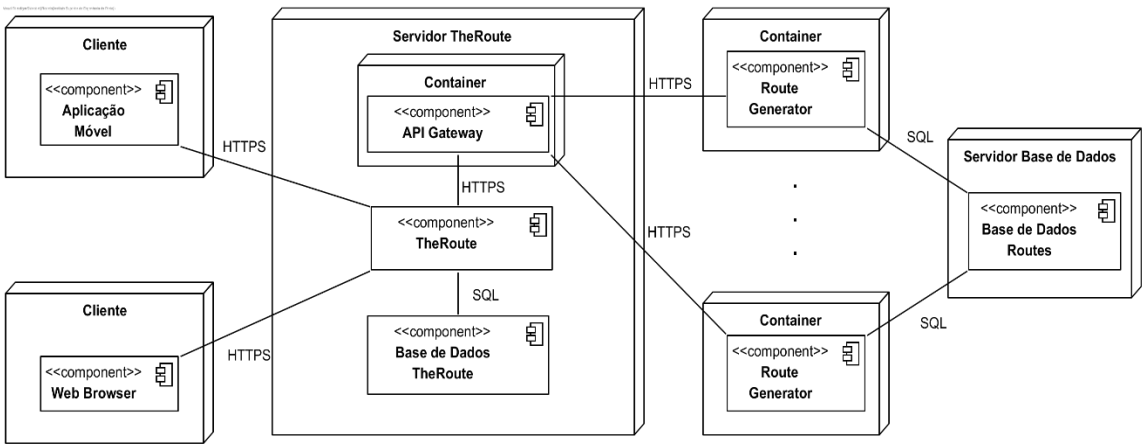


Figura 20 – Diagrama de implantação

Para a configuração apresentada, existem quatro nós distintos: o servidor TheRoute, as aplicações cliente, os microsserviços responsáveis pela criação das rotas e o servidor de base de dados de rotas.

No servidor do TheRoute encontram-se alojadas a aplicação monolítica, que continua a receber os pedidos enviados através das aplicações cliente; a base de dados, que fornece os dados de negócio à aplicação servidor; a API *gateway* que encaminha os pedidos de criação de rota para um microsserviço responsável por as gerar. O *gateway* permite também centralizar algumas das responsabilidades inerentes à monitorização dos *containers*, como é exemplo a gestão de carga (*load balancing*) de instâncias dos *containers*. Com a utilização do Ambassador, a API *gateway* nativo do Kubernetes, é também possível diminuir o esforço

relativo à implantação e manutenção do *gateway*, do que são exemplos a resiliência e escalabilidade de instâncias do microsserviço. Os microsserviços, por sua vez, encontram-se alojados em Docker *containers*, podendo estar distribuídos por diversas máquinas na *cloud*. Existe também um servidor que contém a base de dados a utilizar pelos microsserviços, onde são registadas as rotas geradas. Assim, cada instância do microsserviço de criação de rotas acede apenas a uma base de dados dedicada para esse fim, indo de encontro ao definido pelo padrão base de dados por microsserviço (*database per service*). Os nós cliente são os telemóveis ou *web* browsers usados pelos utilizadores da aplicação e comunicam via HTTPS com a aplicação servidor, utilizando os serviços *web* disponibilizados pela mesma.

A geração de rotas é um processo pesado para o sistema, uma vez que pode requerer muita utilização de CPU e memória. Se o módulo de geração de rotas fosse integrado na aplicação existente não seriam precisos muitos pedidos para que todo o sistema ficasse lento e deixasse de estar operacional. A arquitetura adotada para a criação das rotas permite escalar microsserviços consoante a carga do sistema, o que não seria possível se a criação de rotas fosse incluída na aplicação monolítica existente. Desta forma, as operações pesadas de pesquisa para a criação de uma nova rota são feitas por um sistema independente que pode correr em várias máquinas distribuídas, não provocando carga no servidor que detém o restante sistema.

Como arquitetura de implantação alternativa seria possível colocar a API *gateway* noutra máquina ou na *cloud* e permitir replicações das suas instâncias, por exemplo. É possível visualizar essa arquitetura na Figura 21, que para o caso específico deste projeto não traz grandes vantagens. Além dos custos de manutenção serem mais elevados para esta solução, não existiria uma diferença significativa no desempenho do sistema, uma vez que os pedidos provenientes da aplicação móvel continuam a convergir para a mesma aplicação monolítica.

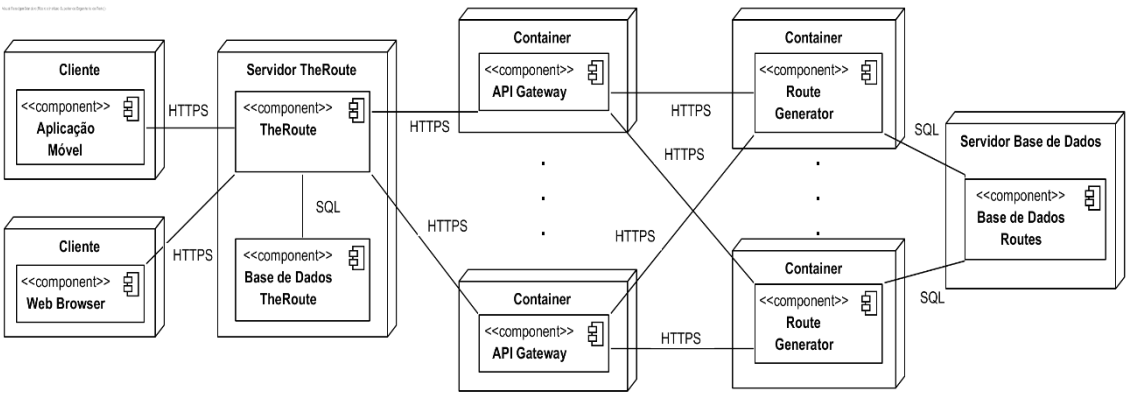


Figura 21 – Diagrama de implantação alternativa

3.3 Design do Caso de Uso

Nesta secção é analisada com maior detalhe a interação do utilizador com o sistema. Tendo em conta o contexto desta dissertação, e visto que o módulo de geração de rotas tem o único fim de criar o trajeto a realizar pelo utilizador da aplicação móvel, apenas existe um caso de uso relevante. Esse caso de uso consiste na criação de uma rota personalizada para o utilizador, tendo em conta os seus gostos e alguns aspetos do contexto da viagem.

Na Figura 22 é apresentada a vista geral do diagrama de sequência para o caso em questão. Devido à extensão do diagrama, parte das suas interações foram divididas por dois diagramas de sequência adicionais, apresentados na Figura 23 e Figura 24.

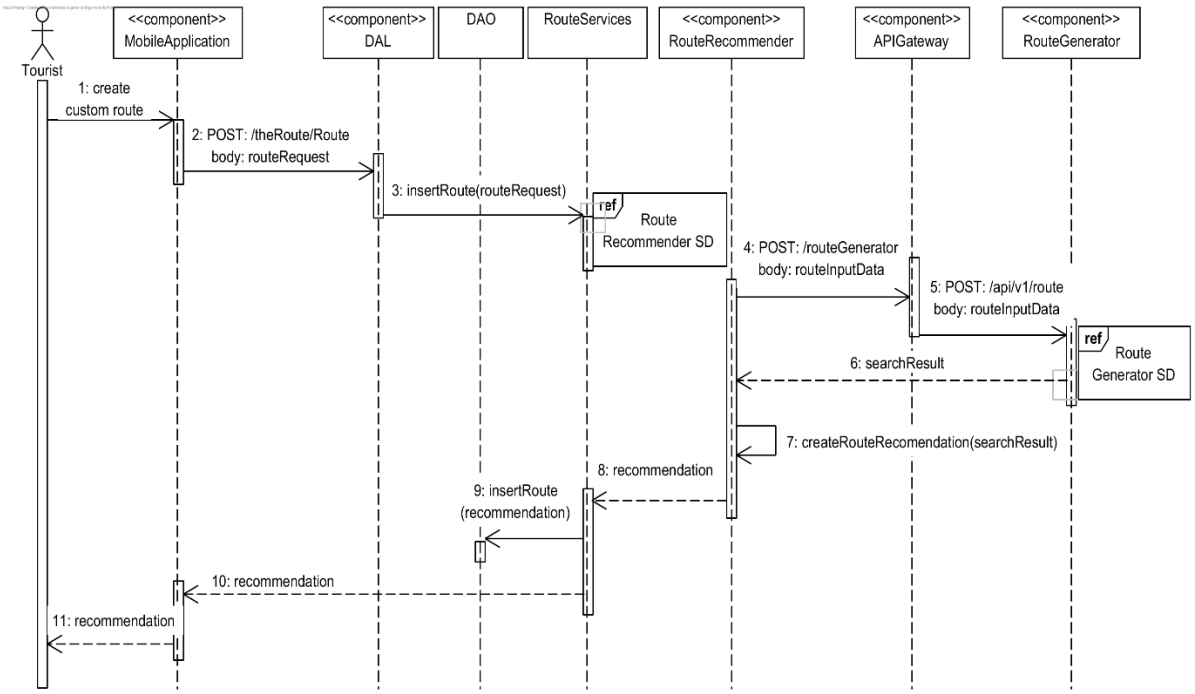


Figura 22 – Diagrama de sequência para a criação de uma rota personalizada

Depois de se autenticar na aplicação móvel, o utilizador depara-se com o ecrã principal da aplicação. Nesse ecrã existem várias opções, entre elas a de criação de uma rota turística personalizada. Ao selecionar essa opção é apresentado ao turista um formulário para inserção dos dados relevantes para a criação da rota. Os campos a preencher são o dia da semana em que pretende realizar o trajeto; a hora de início; a hora de fim; o meio de transporte a utilizar (carro ou a pé) e as coordenadas do ponto de partida (localização atual ou outra localização escolhida pelo turista). Depois de preencher esses dados o turista submete o formulário, sendo enviado um pedido HTTPS para a API do componente DAL pertencente ao sistema TheRoute. Os dados enviados no pedido são direcionados para a classe RouteServices, que é a responsável por tratar os pedidos relativos à gestão de rotas.

Até este ponto, as interações apresentadas não são muito diferentes das que já estavam implementadas no sistema TheRoute. O segmento do diagrama presente na Figura 23 revela já algumas das alterações ao fluxo tradicional da aplicação, com a introdução de novas classes e componentes.

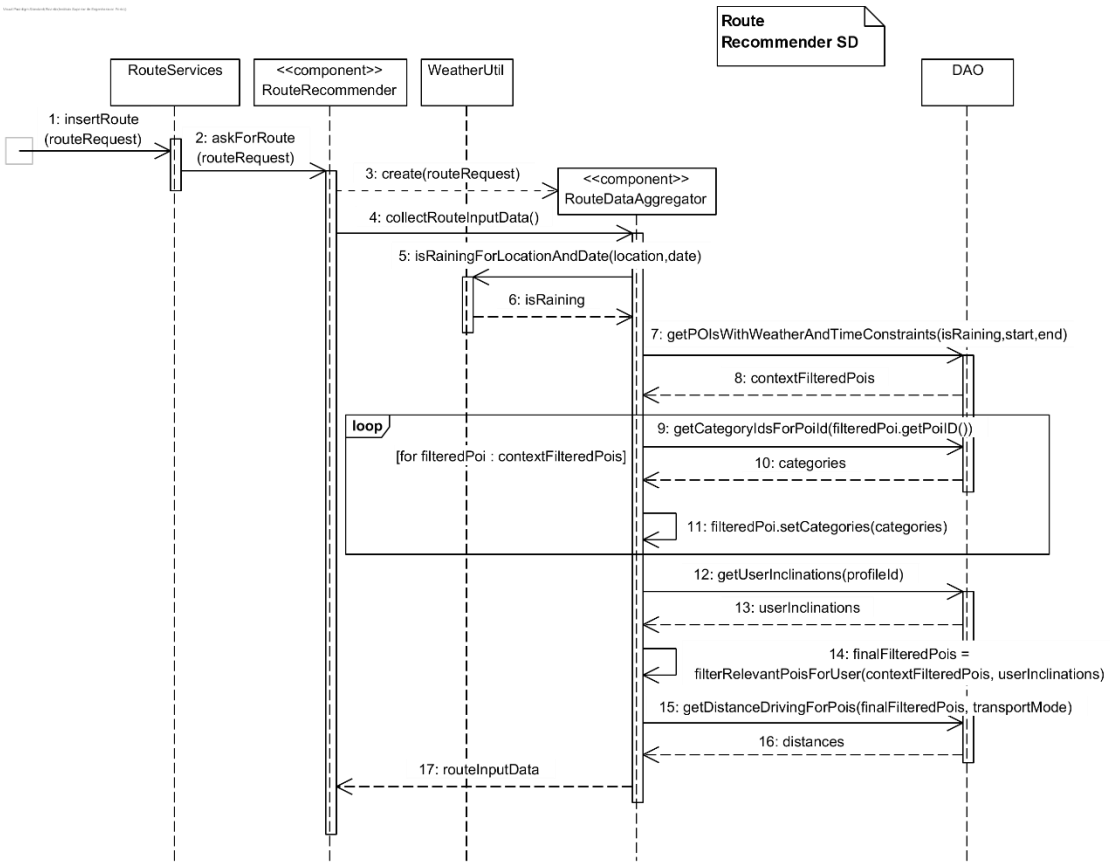


Figura 23 – Diagrama de sequência para a filtragem dos pontos de interesse

O componente RouteRecommender recebe os dados de entrada redirecionados pela classe RouteServices e cria uma instância do componente RouteDataAggregator. Este, por sua vez, tem a responsabilidade de recolher e filtrar todos os dados relevantes à criação da rota. Numa primeira fase procura informar-se sobre as condições meteorológicas para o dia da viagem. De seguida, acede ao DAO do sistema TheRoute para solicitar os pontos de interesse que estão abertos no período indicado e que obedecem às restrições meteorológicas obtidas anteriormente. Para cada um dos pontos de interesse obtidos, são solicitadas as suas respetivas categorias. Neste caso específico, as categorias do ponto de interesse são relevantes para determinar a pontuação de cada local, tendo em conta as tendências do utilizador para cada categoria. Após receber essa informação, o RouteDataAggregator faz a filtragem final dos pontos de interesse, descartando todos os que possuem pontuações abaixo de um determinado valor mínimo. Por fim, são pedidas todas as distâncias existentes entre os pontos de interesse selecionados e enviada a informação recolhida para o componente RouteRecommender. Cabe a este reencaminhar essa informação para o microsserviço responsável pela criação da rota. Para tal, e como o endereço do microsserviço não é

conhecido, o pedido deve passar primeiro pela API *gateway*, que fará o seu reencaminhamento para uma instância de microsserviço que esteja disponível.

O processo de criação de uma rota pode ser consultado no segmento do diagrama presente na Figura 24.

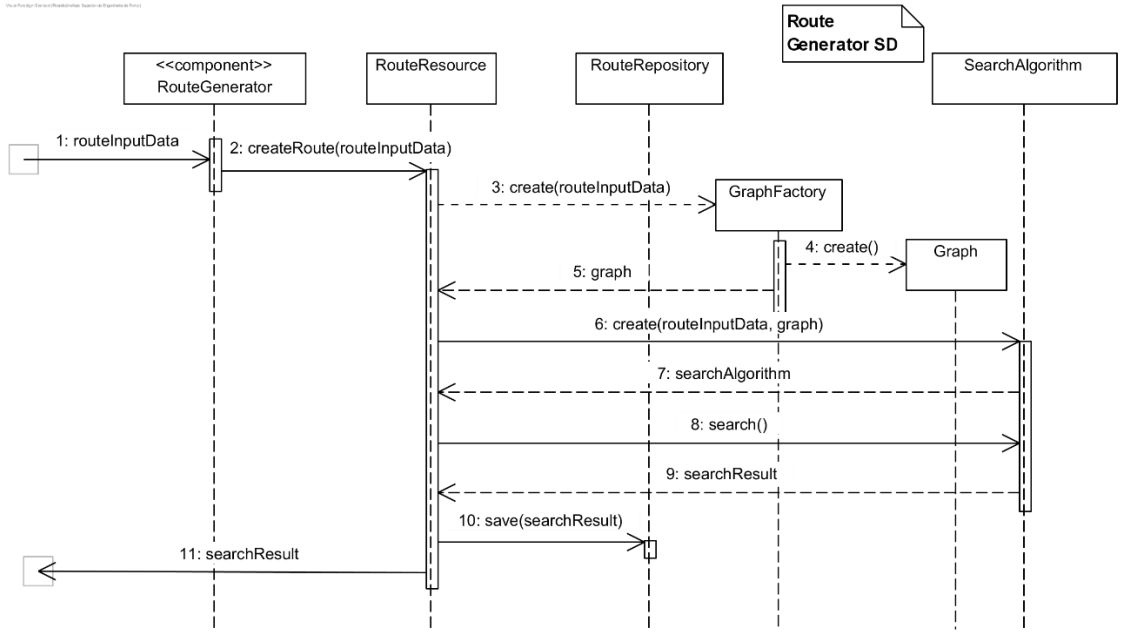


Figura 24 – Diagrama de sequência para a criação de uma rota

Com o conjunto de dados recebidos é possível construir um grafo entre todos os pontos de interesse filtrados. Esse grafo é construído pela classe GraphFactory. Os pontos de interesse são representados como nós do grafo e as distâncias entre eles constituem as arestas. É depois criada uma instância do algoritmo de pesquisa que recebe como parâmetro o grafo gerado e a informação enviada pelo RouteRecommender. O algoritmo gera uma rota que é guardada na persistência do microsserviço e enviada como resposta ao pedido do componente RouteRecommender. Com base nos dados que detém e na rota recebida, este componente cria uma recomendação, que é persistida na base de dados e enviada à aplicação móvel. Após a sua receção, a aplicação móvel apresenta graficamente ao utilizador a recomendação gerada.

Como se pode concluir através da análise ao caso de uso, o fluxo de geração de uma rota personalizada é um processo trabalhoso, que depende de diversos componentes. Por essa razão, torna-se fulcral que o algoritmo a utilizar pelo componente RouteGenerator retorne resultados em tempos razoáveis, de modo a não causar transtorno ao utilizador.

4 Implementação da Solução

Neste capítulo são descritas as principais fases do processo de implementação da solução desenvolvida no contexto desta dissertação. Na secção 4.1 são apresentadas a pesquisa e a abordagem inicial ao problema, assim como as respetivas conclusões que serviram de base à implementação proposta. Por sua vez, a secção 4.2 é a responsável pela apresentação dos algoritmos desenvolvidos. Ainda nesta secção são apresentadas as otimizações efetuadas para permitir uma maior velocidade de execução dos algoritmos e uma filtragem inicial mais seletiva dos pontos de interesse.

4.1 Abordagem e Considerações Iniciais

A fase inicial da dissertação consistiu numa pesquisa na literatura por soluções semelhantes à que se pretende desenvolver. Como foi possível verificar pelo estado da arte, foram encontrados bastantes trabalhos e protótipos na área da criação de rotas turísticas. Pelo que se conseguiu apurar, apesar de existirem sistemas semelhantes ao TheRoute, nenhuma das soluções cria rotas personalizadas tendo em conta o conjunto das características individuais do utilizador e o contexto envolvente. Por essa razão, este produto pode-se destacar dos restantes ao fazê-lo e ao conseguir gerar rotas turísticas num curto espaço de tempo.

Todo o *backend* já desenvolvido para o sistema TheRoute utiliza o Java como linguagem de programação. Uma vez que o módulo de geração de rotas é um projeto independente do restante *backend*, ao fazer toda a sua comunicação através de uma API REST, poder-se-ia utilizar outra linguagem para o seu desenvolvimento, como por exemplo Python ou C. No entanto, apesar de existirem outras linguagens que já provaram ser mais eficientes em problemas deste género, decidiu-se mesmo assim adotar o Java para desenvolver este módulo. Não tendo o melhor desempenho ao nível do consumo de recursos computacionais, o Java possui argumentos que muitas outras linguagens não detêm. Por ser uma linguagem orientada a objetos e possuir uma API rica em estruturas de dados bastante otimizadas, como são exemplos o Map, Set, List, Stack e Queue, esta linguagem ganha vantagem em relação a outras, onde muitas vezes é necessário implementar estas estruturas manualmente. Além

disso, possui uma sintaxe simples e intuitiva. O próprio *garbage collector* é otimizado e permite tirar uma grande responsabilidade do programador, que não precisa de se preocupar com a destruição de objetos que já não estão a ser utilizados em memória. Por esta razão, a probabilidade de acontecerem erros em tempo de execução é também menor. Por outro lado, ao nível dos custos de manutenção do projeto, é também vantajoso possuir um *backend* uniformizado a nível tecnológico, uma vez que a existência de várias linguagens de programação pode obrigar à contratação de profissionais especializados em cada uma das áreas. A quantidade de bibliotecas existentes e o seu respetivo suporte em Java é igualmente um fator a ter em conta.

Antes de se decidir qual a melhor abordagem a seguir para a implementação dos algoritmos de geração de rotas, houve um trabalho de investigação e experimentação de bibliotecas já existentes. Foram encontradas algumas *frameworks* com bastante aceitação pela comunidade, sendo que algumas são inclusivamente utilizadas em produtos disponíveis ao público. De seguida é apresentada uma lista com as *frameworks* exploradas:

- Jenetics: é uma biblioteca orientada a algoritmos genéticos e evolucionários. Possui um motor próprio responsável pelas evoluções iterativas das populações, capaz de avaliar os seus indivíduos com o auxílio de uma função de aptidão (*fitness*);

- jMetal: oferece uma grande variedade de algoritmos meta-heurísticos incluindo, entre outros, algoritmos genéticos, otimização por enxame de partículas e algoritmos multiobjectivo. A biblioteca também oferece suporte para realizar estudos experimentais completos e a geração automática de informações estatísticas dos resultados obtidos, aproveitando as capacidades dos núcleos disponibilizados pelo processador para acelerar o seu tempo de execução (Durillo & Nebro, 2011);

- JGraphT: é uma biblioteca bastante completa ao nível da quantidade de algoritmos oferecida. Disponibiliza algoritmos direcionados para problemas que envolvem pesquisas iterativas a grafos. Para o caso específico da pesquisa do caminho mais curto, possui uma enorme variedade de implementações algorítmicas, como são exemplos o algoritmo de Dijkstra, Bellman-Ford, Floyd-Warshall, entre muitos outros;

- Hipster4J: desenvolvida pelo Centro de Investigação em Tecnologias da Informação (CiTIUS) da Universidade de Santiago de Compostela, a Hipster4J permite resolver uma vasta gama de problemas de pesquisa em grafos. Fornece uma grande variedade de algoritmos de pesquisa clássicos implementados de forma iterativa, como são exemplos o Dijkstra, A*, IDA*, AD* e *Simulated Annealing* (Rodriguez-Mier, Gonzalez-Sieira, Mucientes, Lama, & Bugarin, 2014).

De todas as bibliotecas experimentadas, a Hipster4J pareceu ser a mais bem estruturada, não sendo complicada a sua perceção. No entanto, possui demasiada abstração de conceitos e os resultados obtidos para a grande maioria dos *inputs* introduzidos não correspondiam à realidade pretendida. As condições de paragem e as heurísticas definidas para a escolha do nó

seguinte parecem também não funcionar como o esperado. Num período anterior ao desta dissertação foi implementada uma solução com a biblioteca Jenetics. No entanto, o seu uso não se revelou muito viável, uma vez que demora muito tempo a encontrar uma solução válida. O jMetal seria bastante útil para geração de informações estatísticas a apresentar no próximo capítulo, mas as tentativas em utilizar a biblioteca para o contexto da dissertação não foram bem-sucedidas. A grande quantidade de parâmetros a configurar e a forma como a informação é organizada em ficheiros foram os principais fatores para o insucesso desta integração. O JGraphT é uma biblioteca bastante robusta que oferece diversas opções. Juntamente com o Hipster4J foi a que esteve mais próxima dos resultados esperados. Regra geral, as bibliotecas encontradas possuem um nível de complexidade demasiado elevado para o que se pretende neste projeto. Apesar de terem sido desenvolvidas de uma forma abstraída, é difícil adaptar o contexto do problema às condicionantes impostas. O elevado grau de complexidade na compreensão de código e o excesso de dependências externas, provocam também uma sobrecarga extra de recursos.

4.2 Solução Proposta

Após a pesquisa efetuada e as tentativas de utilização de *frameworks* já implementadas, decidiu-se que nenhuma das hipóteses encontradas se adequava à resolução do problema em tempo razoável. Por essa razão, optou-se por criar toda a solução de raiz, não recorrendo a bibliotecas de terceiros. As próximas secções apresentam a abordagem idealizada para a implementação dos mecanismos de geração de rotas.

4.2.1 Estrutura e Modelação

Para encontrar uma solução que resolva o problema proposto, é necessário descobrir um caminho entre diversos pontos de interesse, que corresponda às expectativas do utilizador e que tenha em consideração as condicionantes do meio envolvente. Considera-se que cada ponto de interesse possui uma ligação entre todos os restantes. Desta forma, os pontos de interesse podem ser interpretados como nós de um grafo ligados entre si através de caminhos ou arestas. Cada um destes trajetos tem associado um peso ou custo, que é medido tendo em conta um critério relevante para o problema. Em casos como o desta dissertação, onde os nós representam pontos geográficos num plano, o custo considerado costuma ser a distância espacial entre os locais. No contexto que tem sido apresentado, não seria de todo incorreto considerar este critério. No entanto, como é definido um espaço temporal no qual a viagem tem de ocorrer, optou-se por considerar o tempo de deslocação para medir o esforço de uma deslocação entre nós do grafo.

Com a execução de um algoritmo de pesquisa sobre um grafo, pretende-se encontrar uma rota, tendo em conta um determinado ponto de partida e um tempo limite para efetuar a totalidade do percurso. Para além do algoritmo de pesquisa poder variar, a heurística a utilizar para a escolha do ponto de interesse a visitar pode também diferir consoante a

implementação do algoritmo. Tendo em conta as semelhanças e diferenças enumeradas, recorreu-se ao polimorfismo para a definição conceptual de algoritmos de pesquisa e heurísticas, como a título de exemplo é demonstrado na Figura 25.

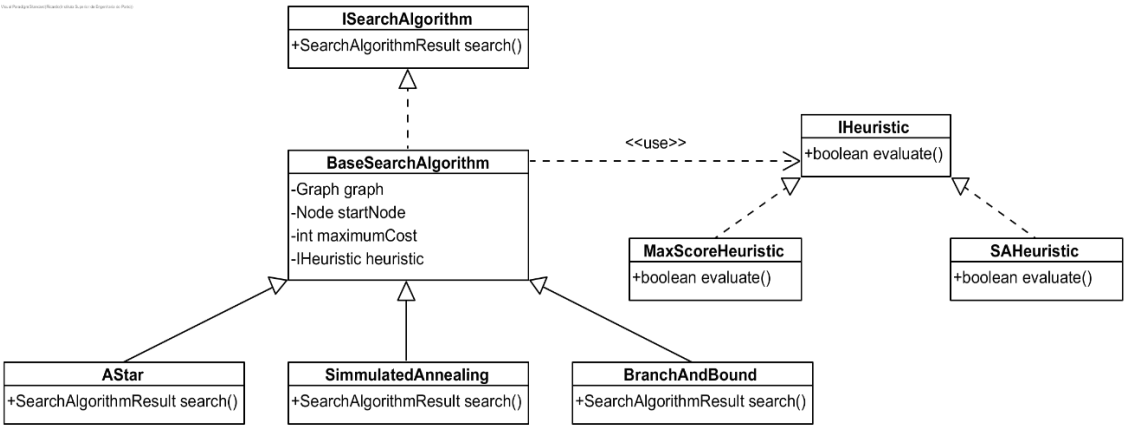


Figura 25 – Definição conceptual de algoritmos e heurísticas

O algoritmo de pesquisa, como o próprio nome indica, tem o objetivo de procurar algo num determinado espaço de pesquisa. A metodologia adotada para atingir esse fim é diferente de algoritmo para algoritmo. No entanto, se se pretender trocar uma implementação por outra, é esperado que essa alteração seja simples e não altere profundamente o fluxo de execução da aplicação. Para facilitar essa integração, foi criada a interface ISearchAlgorithm que todos os algoritmos de pesquisa devem implementar. Adicionalmente foi definida uma classe abstrata BaseSearchAlgorithm, que contém as propriedades comuns e necessárias à modelação do problema definido. Ao ser estendida pelas implementações concretas dos algoritmos, estes vão herdar as suas propriedades, podendo adicionalmente definir outras que sejam indispensáveis à sua execução. Desta forma, ao fazer uso do polimorfismo e da herança, consegue-se obter o mesmo tipo de resultado com implementações diferentes sem ter de alterar o restante fluxo de execução da aplicação.

No contexto desta dissertação, a função heurística a utilizar pelo algoritmo de pesquisa serve para avaliar se, para o objetivo definido, um movimento de um ponto de interesse para outro é favorável ou não. Essa função efetua um cálculo matemático definido e indica ao algoritmo se é benéfico ou não o movimento de um nó do grafo para um dos seus vizinhos. Desta forma, independentemente do tipo de cálculo a efetuar, a heurística terá obrigatoriamente de devolver uma avaliação sobre o movimento. Por essa razão, foi criada uma interface IHeuristic que declara o método *evaluate*, implementado por todas heurísticas. Assim, é possível abstrair o conceito de heurística da sua implementação, não ficando o algoritmo restringido a um tipo concreto de heurística.

4.2.2 Seleção e Otimização Inicial

Como foi referido na secção 3.1.1, para os N nós do grafo, o número de combinações de caminhos é $N!$. Por consequência, quanto maior for o número de nós do grafo, maior o número de ligações entre eles e maior o número de combinações possíveis. A base de dados criada para dar suporte aos testes da funcionalidade de criação de rotas possui 266 pontos de interesse. Apesar de parecer um número pequeno, se todos esses pontos fossem considerados na geração da rota, existem 266! combinações possíveis, o que equivale a aproximadamente $1{,}28 \times 10^{531}$ possibilidades. Está claro que com a evolução da plataforma os 266 nós serão rapidamente ultrapassados, podendo talvez chegar à casa dos milhares de ponto de interesse. Por essa razão, seja qual for o algoritmo a utilizar na geração da solução final, interessa que receba como entrada o menor número de nós e ligações possíveis, tendo em conta as restrições do problema. Uma boa seleção inicial dos constituintes do grafo torna-se, por isso, essencial ao bom desempenho da pesquisa.

Foram tidos em conta vários aspetos na filtragem inicial dos pontos de interesse. Além da escolha desses critérios, a sua ordem de realização é também relevante para qualidade da solução, uma vez que existem formas mais ou menos eficientes de restringir o número de *queries* à base de dados e a quantidade de objetos em memória. Um mau processo de seleção inicial dos dados pode atrasar a criação da recomendação turística e, consequentemente, prolongar o tempo de resposta ao pedido.

De seguida, são apresentadas as etapas adotadas para a filtragem inicial dos dados de entrada para o algoritmo de geração de rotas:

1. **Verificar os dados meteorológicos**

O modelo existente para representar o ponto de interesse possui três propriedades relativas a dados meteorológicos. Esses atributos são a temperatura mínima e a máxima aconselháveis para visitar o local e a sua exposição à chuva. Através da triangulação entre estes atributos e os dados meteorológicos consultados na API do OpenWeather é possível restringir os pontos que devem ser considerados para visita. É efetuada a seleção de um local se forem cumpridas três condições:

- A temperatura mínima para o dia da visita tem de ser igual ou superior à temperatura mínima aceitável para visitar o local;

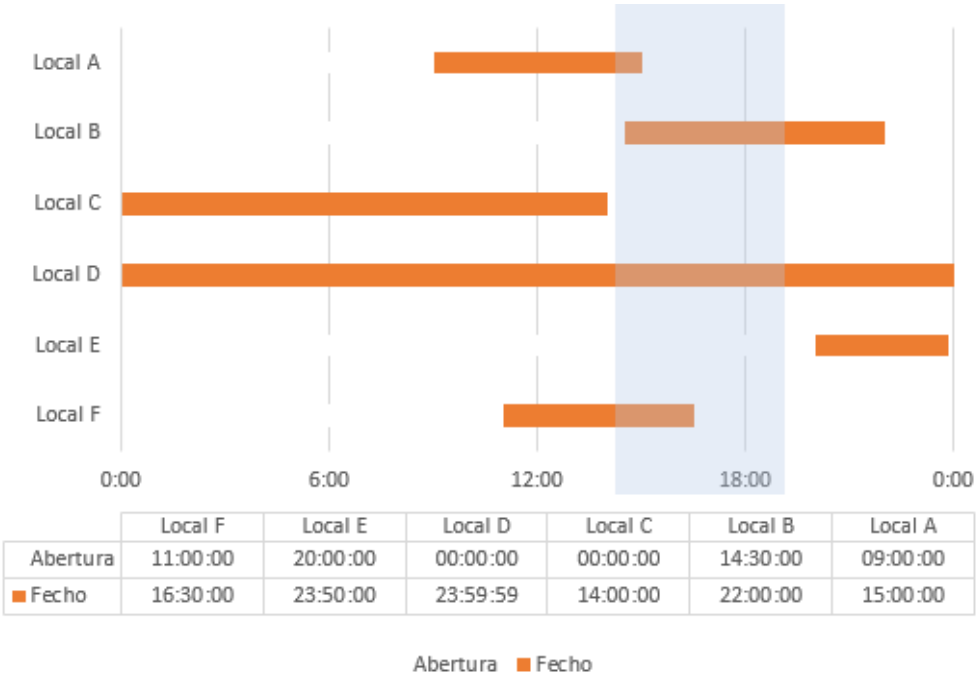
- A temperatura máxima para o dia da visita tem de ser igual ou inferior à temperatura máxima aceitável para visitar o local;

- A ocorrência ou ausência de chuva tem de ser compatível com os requisitos do ponto de interesse. Por outras palavras, pretende-se selecionar todos os pontos de interesse, caso não haja previsão de aguaceiros, ou então selecionar apenas os locais cuja ocorrência de chuva não afete a sua visita.

Tendo estas três condições reunidas, é possível dar início à filtragem dos pontos de interesse.

2. Filtragem inicial dos pontos de interesse

Para a filtragem dos pontos de interesse, além da satisfação das condições meteorológicas, é importante ter em conta se os horários de funcionamento são compatíveis com os da visita. O dia e horário do passeio são escolhidos pelo utilizador quando solicita uma recomendação de percurso. Assim, ao filtrar os pontos de interesse cujos horários de funcionamento são compatíveis com os tempos da viagem, é possível restringir a quantidade de locais a ter em conta para a pesquisa. Não faz sentido a primeira amostra de pontos de interesse conter locais que não se encontram abertos no dia e horário da visita ou, como já referido, sejam suscetíveis à influência das condições climatéricas. Desta forma, só são considerados para a amostra inicial os pontos de interesse que tenham um horário de funcionamento e condições atmosféricas compatíveis com a data e hora do passeio. A Figura 26 exemplifica o anteriormente descrito para um determinado conjunto de locais a visitar entre as 14h15 e as 19h00 de uma quarta-feira.



	Local F	Local E	Local D	Local C	Local B	Local A
Abertura	11:00:00	20:00:00	00:00:00	00:00:00	14:30:00	09:00:00
Fecho	16:30:00	23:50:00	23:59:59	14:00:00	22:00:00	15:00:00

Figura 26 – Exemplo de seleção de pontos de interesse para um intervalo de tempo

Esta filtragem inicial serve para excluir todos os pontos de interesse que se encontram fora da janela temporal definida pelo utilizador. No entanto, no decorrer da pesquisa efetuada pelo algoritmo, com a evolução do tempo alguns dos locais inicialmente filtrados podem já ter encerrado. Essa verificação é responsabilidade do algoritmo e terá de ser efetuada no decorrer da sua execução. Desta fase, surge a primeira lista filtrada contendo os pontos de interesse candidatos.

3. Procurar as preferências do utilizador para cada categoria

Cada ponto de interesse pode pertencer a várias categorias turísticas. Tal como referido na secção 3.1.2, cada turista possui o seu perfil de utilizador ao qual estão associadas as suas preferências para cada uma das categorias existentes no sistema. Desta forma, para escolher com qualidade os pontos de interesse candidatos à visita, é importante perceber o quanto eles se enquadram com os gostos do utilizador. As preferências do utilizador para cada categoria são atribuídas tendo em conta uma classificação entre zero e cinco. Para cada ponto de interesse presente na lista de candidatos faz-se uma média das pontuações das suas categorias, tendo em conta os respetivos valores ($valorCategoria$) correspondentes ao perfil do utilizador. Se o número de categorias (c) for superior a zero, o cálculo da pontuação para o ponto de interesse é feito recorrendo à seguinte fórmula:

$$PontuacaoIndividualPOI = \frac{\sum_{i=1}^{c} valorCategoria}{c} \quad (1)$$

No caso de não existirem correspondências entre as categorias do ponto de interesse e as que constam no perfil do utilizador, ou seja $c = 0$, é automaticamente atribuída a pontuação zero ao ponto de interesse.

Numa viagem em grupo, o cálculo da pontuação para um ponto de interesse é feito através da média das pontuações individuais do local, tendo em conta os gostos particulares de cada um dos n indivíduos.

$$PontuacaoGrupoPOI = \frac{\sum_{i=1}^{n} PontuacaoIndividualPOI\ (i)}{n} \quad (2)$$

Por fim, é obtida a lista final que contém todos os pontos de interesse candidatos a serem visitados, cujo critério de seleção é possuir uma pontuação igual ou superior a três. Foi definido este valor de aceitação por ser o valor intermédio na escala definida.

4. Colecionar as distâncias entre todos os pontos de interesse selecionados

Após obter todos os pontos candidatos à visita, resta aferir todas as distâncias existentes entre si. Como referido em 4.2.1, a deslocação entre os pontos de interesse é medida em unidades temporais, mais concretamente em minutos. Esses valores podem ser consultados através da Google Distance Matrix API, que devolve os dados relativos à distância espacial e temporal em formato JSON. É também possível obter esses valores para diversos tipos de meio de transporte, como são exemplo os suportados pela aplicação móvel do TheRoute (a pé ou de carro). Esta API é um serviço com custos associados, sendo o valor a pagar definido em função do número de pedidos e da frequência com que são efetuados. Uma vez que os pontos de interesse se encontram fixos geograficamente, as distâncias existentes entre si não se alteram. Por essa razão, os tempos estimados para as deslocações são também idênticos, podendo apenas variar um pouco quando se tem em conta os condicionamentos provocados pelo trânsito, no caso específico das deslocações de carro. Como a sugestão do plano de

viagem é normalmente pedido com antecedência, o fator trânsito não possui grande relevância para a criação do caminho. Por estas razões e tendo em conta a viabilidade económica do projeto, optou-se por guardar na base de dados todos os valores relativos a distâncias e tempos de deslocação entre pontos de interesse. Visto que existem dois meios de deslocação, é necessário pedir e guardar todos os valores tanto para deslocações a pé, como para deslocações de carro. Os valores das deslocações só precisam de ser consultados uma vez e apenas num sentido. Só serão necessárias novas inserções nas tabelas de deslocações sempre que se insira um novo ponto de interesse na aplicação, para se poder construir o grafo sem que faltem tempos de viagem. Esta solução além de ser mais económica é também mais rápida, uma vez que efetuar uma grande quantidade de pedidos a esta API é um processo lento e faria aumentar bastante o tempo de espera do utilizador.

4.2.3 Algoritmos Implementados

Após obter a lista com o conjunto dos pontos de interesse a incluir na pesquisa, assim como os respetivos tempos de deslocação, é construído um grafo com todos estes dados. O passo seguinte consiste em selecionar o ponto de interesse mais próximo à localização indicada pelo utilizador ou à localização física do seu dispositivo móvel. Essa seleção serve para decidir a origem da rota e é efetuada com recurso à seguinte fórmula:

$$Distância = 2 * R * \sin^{-1}\left(\left(\sin\left(\frac{\varphi_2 - \varphi_1}{2}\right)\right)^2 + \cos(\varphi_1) * \cos(\varphi_2) * \left(\sin\left(\frac{\lambda_2 - \lambda_1}{2}\right)\right)^2\right) \quad (3)$$

Onde:
R: Raio da esfera, neste caso é considerado o raio do planeta Terra ≈ 6 371 km
φ_1: Latitude da coordenada geográfica atual do utilizador
φ_2: Latitude da coordenada geográfica do ponto de interesse
λ_1: Longitude da coordenada geográfica atual do utilizador
λ_2: Longitude da coordenada geográfica do ponto de interesse

A fórmula em questão é conhecida como fórmula de Haversine (Marianne, 2014) e calcula a distância entre dois pontos de uma esfera através das suas latitudes e longitudes. Para o caso específico de encontrar o ponto mais próximo, a lista de pontos de interesse filtrados é iterada e são comparadas as coordenadas de cada ponto com a localização enviada pelo utilizador, procurando-se o ponto para o qual o valor resultante do cálculo é menor.

Uma vez construído o grafo e determinado o ponto de partida da visita, resta iterar o grafo com um algoritmo de pesquisa. Como tem vindo a ser mencionado, esta é a tarefa mais morosa de todo o processo de criação de uma rota. Para chegar a uma solução satisfatória foram efetuados vários testes a algoritmos semelhantes aos apresentados no estado da arte. No entanto, grande parte das hipóteses testadas falharam ao tentar encontrar um caminho satisfatório quando o número de nós do grafo era muito elevado ou quando o tempo de viagem era demorado. Apesar de se saber qual o ponto de partida, não se sabe qual o ponto alvo a atingir. A única condição de paragem definida é o tempo para o fim da viagem, que está

limitado pela aplicação móvel a um intervalo máximo de 24 horas. Este motivo também dificultou um pouco a adaptação dos algoritmos estudados, uma vez que não segue uma abordagem tradicional de procura entre um nó de origem e um nó de destino. Outro aspeto divergente da abordagem tradicional, no entanto menos condicionante, é que em vez de procurar o menor caminho o algoritmo deve encontrar o percurso cuja soma das pontuações dos seus pontos de interesse seja maior.

Existem enumeras abordagens no campo da algoritmia para a criação de algoritmos de pesquisa. As opiniões não são consensuais, enquanto alguns autores defendem a implementação de algoritmos recursivos, outros preferem algoritmos iterativos. Para este projeto decidiu-se utilizar algoritmos iterativos em detrimento dos recursivos, por se considerar que as implementações recursivas têm maior probabilidade de conduzir a ciclos infinitos do que as implementadas de forma iterativa. Se o algoritmo não terminar a sua execução no tempo certo e continuar a chamar-se recursivamente irá certamente provocar erros na execução do programa, pondo em causa os recursos do sistema. Além disso, os algoritmos recursivos envolvem uma maior sobrecarga de recursos, não devendo ser utilizados quando existe a possibilidade de existirem muitos níveis de recursão. Numa implementação iterativa estes problemas são mais fáceis de detetar, evitar e corrigir.

Das tentativas de adaptação efetuadas apenas se conseguiu obter resultados satisfatórios para o algoritmo A* e para o algoritmo *Ant Colony Optimization*. Para as restantes abordagens houve dificuldades em adaptar o algoritmo ao problema em questão. Em alguns casos foram implementadas soluções que demoravam demasiado tempo a encontrar uma rota. Para os casos de sucesso, cujos resultados iniciais foram mais animadores, surgiu também o problema de funcionarem bem para espaços de tempo reduzidos e não funcionarem para tempos de visita muito elevados. O tempo consumido na geração do caminho não era aceitável na maioria dos testes efetuados. Os algoritmos genéricos costumam obter boas soluções, mas não foram testados no âmbito desta dissertação, uma vez que existia outro trabalho de experimentação a ser desenvolvido em paralelo para este tipo de algoritmos. Ficou estabelecido que se os resultados desse trabalho fossem publicados antes da conclusão desta dissertação, seriam introduzidos como ponto de comparação no capítulo da Experimentação e Avaliação.

O *Ant Colony Optimization* é um algoritmo não determinístico devido à heurística utilizada para a aceitação do nó seguinte a visitar, o que torna bastante imprevisíveis os resultados obtidos. Além de imprevisíveis, os resultados nem sempre são próximos do valor máximo possível para a soma das pontuações dos pontos de interesse, podendo ficar muito aquém das expectativas do turista. O A* tem o inconveniente de ser um algoritmo bastante ganancioso em termos de consumo de memória e CPU. Por outro lado, dependendo das heurísticas utilizadas para a decisão do nó a escolher, pode ser determinístico, devolvendo a mesma solução para o mesmo conjunto de *inputs*.

As próximas duas secções apresentam os dois algoritmos desenvolvidos, recorrendo à exposição do seu pseudocódigo e a uma breve explicação sobre a sua implementação.

4.2.3.1 A*

O A*, apresentado na secção 2.3.7, é um algoritmo bastante utilizado para encontrar o caminho mais curto entre dois pontos num grafo. Para o problema a resolver, não interessa descobrir qual o caminho mais curto entre um nó de origem e um nó de destino, mas pode-se perfeitamente utilizar este algoritmo com algumas alterações. Em vez do caminho mais curto, é procurado o caminho com a maior soma das pontuações dos pontos de interesse. Portanto, em vez de se querer minimizar a soma dos custos da transição de um ponto de interesse para outro, pretende-se maximizar o valor resultante da soma das suas pontuações. Também não existe o conceito de nó de destino, uma vez que só se conhece o nó de origem, que é o nó mais próximo à localização do utilizador. Desta forma, através do nó de origem, é necessária uma pesquisa exploratória pelo grafo até atingir a condição de paragem. Essa condição consiste em a soma dos custos de deslocação de um ponto de interesse para o outro, incluindo os tempos de visita, ser igual ou inferior ao tempo total indicado para realizar a viagem. Uma vez atingida, termina a exploração desse caminho, havendo a possibilidade de explorar outros.

A implementação adotada, além dos pressupostos indicados anteriormente, utiliza ainda outras estratégias características do A*. É utilizada uma fila, conhecida como "*open set*", que serve para armazenar os nós sobre os quais existe interesse explorar durante a execução do algoritmo. Existe também uma lista com os nós já visitados, conhecida por "*closed set*". Esta lista serve para evitar a ocorrência de ciclos no grafo, não permitindo que se volte a passar por um nó já visitado. No início da execução do algoritmo, o nó de origem é adicionado à fila de nós a visitar, assim como à fila de nós visitados. De seguida, a fila de nós a visitar é percorrida até que não existam mais nós a explorar. O primeiro nó da fila é removido, sendo explorado cada um dos seus vizinhos. Se o vizinho não constar na lista de nós visitados, é verificado se a soma dos tempos de deslocação e visita até esse ponto é superior ao tempo limite da visita ou se o ponto de interesse se encontra fechado para o período desejado. Caso estas condições sejam satisfeitas, o caminho analisado é descartado, não voltando a ser verificado. No caso de ser possível efetuar a visita ao ponto de interesse em questão e a heurística escolhida para avaliação for favorável, adiciona-se o nó à fila de nós a explorar em futuras iterações. Caso contrário esse nó é ignorado e não volta a ser explorado.

O Algoritmo 1 demonstra a implementação em pseudocódigo do algoritmo A* para encontrar uma rota turística.

Algoritmo 1 – Algoritmo A* (pseudocódigo)

```
1: Input: startNode - Start Node; timeLimit - Time Limit; visitedNodes - Set of visited nodes
2: Output: bestPath - Best path found for the heuristic
3:
4: openSet ← ∅
5: closedSet ← visitedNodes
6: bestPath ← null
7: add startNode to openSet
8: while openSet ≠ ∅ do
9:   node ← openSet.poll()
10:  neighbours ← node.getNeighbours()
11:  foreach neighbour ∈ neighbours do
12:    if neighbour ∉ closedSet then
13:      add neighbour to closedSet
14:      path ← calculatePath(node, neighbour)
15:      pathStartingTime ← calculatePathStartingTime(path)
16:      pathEndingTime ← calculatePathEndingTime(path)
17:      if pathEndingTime > timeLimit ∨ ¬neighbour.isOpened(pathStartingTime) then
18:        continue                                              ▷ Path Discarded
19:      else
20:        if heuristic.evaluate(bestPath,path) then
21:          bestPath ← path
22:          add neighbour to openSet
23:        end if
24:      end if
25:    end if
26:  end for
27: end while
28: return bestPath
```

Contudo, o código acima demonstrado não é suficientemente rápido encontrar uma solução quando os espaços de tempo considerados são muito alargados ou quando existem muitos nós a analisar. Por essa razão, foi ainda necessário dividir a procura por espaços mais pequenos, aplicando cortes à pesquisa. Após se experimentar vários valores diferentes, concluiu-se que o algoritmo se tornava mais estável para pesquisas de 120 minutos. Dessa forma, o tempo total da visita é dividido em porções de 120 minutos e, de forma iterativa, é invocado o algoritmo acima apresentado. No caso específico de o tempo total da visita ser inferior ao valor definido, será feita apenas uma iteração tendo esse valor em consideração. Cada iteração devolve um caminho ou fragmento do trajeto final que, após concluídas todas as iterações, será integrado na rota a recomendar ao turista.

Para o algoritmo desenvolvido foram idealizadas duas variantes, tendo em conta duas heurísticas diferentes para a aceitação de locais numa rota.

Heurística para o Maior Resultado

A heurística para o maior resultado visa, tal como o próprio nome indica, encontrar o conjunto de pontos de interesse para o qual a soma das suas pontuações seja a mais elevada. Para a quantidade de dados a analisar, procurar a combinação de pontos com maior resultado é uma operação complicada, podendo consumir bastante tempo até encontrar uma solução. Como explicado anteriormente, a abordagem adotada consegue diminuir consideravelmente o espaço de pesquisa ao encontrar uma solução aceitável recorrendo a cortes. Consequentemente, esta heurística tem a simples responsabilidade de verificar se o caminho a analisar possui uma pontuação total igual ou superior à pontuação máxima registada pelo algoritmo até ao momento. Caso a heurística se aplique, esse caminho substitui o anterior com maior pontuação e passa a servir de comparação para as iterações seguintes. Devido a essa substituição ocorrer cada vez que se encontra um caminho com valor igual ou superior, pode ocorrer um dos seguintes cenários relativos ao caminho substituído:

- Rejeita-se um caminho que atingiu o tempo limite da visita e nunca iria possuir um valor superior ao novo caminho registado;

- Rejeita-se um caminho que, apesar de ainda não ter atingido o tempo limite da visita, se continuasse a ser explorado em futuras iterações poderia evoluir para valores superiores à solução final.

Por estas razões, esta heurística não garante a solução ideal para o problema, mas encontra uma solução otimizada para o turista. É otimizada porque apresenta um caminho que tem em conta os seus gostos e preferências, descartando à partida rotas que não seriam do seu agrado. Além disso, permite ao algoritmo responder em tempo aceitável, não fazendo o utilizador desistir de utilizar o sistema e optar por outros meios de planeamento para a sua rota turística.

Heurística baseada no *Simulated Annealing*

Após a análise do algoritmo SA, surgiu a ideia de utilizar a sua heurística para seleção do nó no algoritmo A*. Como já referido em 2.3.4, o processo de decisão no SA é efetuado por intermédio de uma heurística que testa a viabilidade do movimento de um nó para o outro através de uma função de aceitação de probabilidades. Essa função retorna um número (aceitação) que é comparado com um valor aleatório entre 0 e 1. Com a evolução do algoritmo a probabilidade de escolher maus movimentos diminui. No caso de a aceitação ser superior ao número gerado, o movimento é aceite. A função de aceitação idealizada é bastante parecida com a do SA, no entanto possui uma ligeira alteração para a tornar compatível com o algoritmo A* desenvolvido. Uma vez que não existe o conceito de temperatura no A*, esse valor foi substituído na equação pela diferença entre o tempo limite da visita e o tempo acumulado até à iteração a ser verificada. A função de aceitação de probabilidades utilizada pode ser visualizada na expressão seguinte.

$$Ap = e^{\frac{p_{max} - p_{atual}}{t_{final} - t_{atual}}} \quad (4)$$

Onde:
p_{max}: Pontuação máxima acumulada até à atual iteração
p_{atual}: Pontuação atual acumulada à atual iteração
t_{final}: Tempo limite da visita
t_{atual}: Tempo acumulado à atual iteração

Caso esta heurística se aplique, o caminho a testar na iteração em questão substitui o anterior com maior pontuação e passa a ser o trajeto a comparar nas iterações seguintes. Devido a essa substituição ocorrer com recurso à função de aceitação descrita, pode ocorrer um dos seguintes cenários relativamente ao caminho substituído:

- Rejeita-se um caminho próximo de atingir o tempo limite da visita e que dificilmente iria possuir um valor superior ao novo caminho registado;

- Rejeita-se um caminho que possuía uma pontuação superior ao novo caminho registado;

- Rejeita-se um caminho que, apesar de possuir menor pontuação, com a continuação do algoritmo poder-se-ia tornar mais vantajoso do que o escolhido.

Assim, com a utilização desta heurística encontra-se uma solução de seleção híbrida entre o algoritmo A* e o SA, que será um interessante ponto de comparação para a fase de experimentação e avaliação dos algoritmos.

4.2.3.2 Ant Colony Optimization

Tal como o descrito em 2.3.5.2, o *Ant Colony Optimization* (ACO) é um algoritmo de pesquisa que consiste na imitação do comportamento das formigas na deslocação entre a sua colónia e a fonte de alimento. Várias formigas saem da sua colónia à procura de alimento e, caso encontrem alguma coisa, deixam um rasto de feromonas pelo caminho. As restantes formigas que se encontram nas imediações seguem esse rasto, dirigindo-se para esse caminho. Quanto maior o número de formigas no caminho, mais atrativo ele é, uma vez que são libertadas mais feromonas. No entanto, com o passar do tempo, o efeito causado por estas hormonas tende a deteriorar-se devido ao fenómeno de evaporação. Quanto mais longo for um caminho, mais tempo demora a percorrer, sendo menor a densidade de feromonas. Por essa razão, tendem a evaporar-se ainda mais rapidamente. Desta forma, este algoritmo serve para resolver o caminho mais curto entre a colónia de formigas e a fonte de alimento. O caminho mais curto possui uma densidade mais elevada de feromonas sendo, por isso, mais atrativo para as formigas que convergem para um caminho aproximadamente ideal.

Para o problema que se pretende resolver pode-se adotar esta analogia entre as formigas e os turistas, uma vez que ambos desejam obter um caminho que satisfaça o seu objetivo. O que difere nesta comparação é a sua finalidade, uma vez que a formiga quer encontrar o caminho

mais curto entre a colónia e o alimento e o turista tenciona encontrar o percurso que melhor se identifique com as suas preferências. A solução implementada consiste numa adaptação do algoritmo ACO idealizado por (Jungblut, 2015). É uma implementação *multithread* na qual são criadas múltiplas "formigas", que começam de um ponto de origem fixo (ponto de interesse mais próximo do turista) e continuam a exploração de um caminho pelo grafo criado, até atingirem alguma das condições de paragem. Cada formiga decide o ponto de interesse seguinte a visitar com base nos cálculos de probabilidade expressos na seguinte fórmula.

$$p_{xy}^{k} = \frac{\left(\tau_{xy}^{\alpha}\right)\left(\eta_{xy}^{\beta}\right)}{\sum\left(\tau_{xz}^{\alpha}\right)\left(\eta_{xz}^{\beta}\right)} \tag{5}$$

Onde:
k: Número de ordem da formiga (1ª, 2ª, 3ª …)
x: Ponto de origem
y: Ponto seguinte
z: Outro possível ponto seguinte
τ_{xy}: Quantidade de feromonas depositadas no caminho entre x e y
η_{xy}: Conveniência para uma transição entre x e y
α: Valor constante para controlar a influência τ_{xy}, igual ou superior a 0
β: Valor constante para controlar a influência de η_{xy}, igual ou superior a 1

A probabilidade de uma formiga k transitar de um ponto x para um ponto y é influenciada pela quantidade de feromonas depositadas no caminho entre esses dois pontos τ_{xy} . Este valor é elevado a uma constante α, que é um parâmetro heurístico definido na criação do algoritmo e descreve o quão ganancioso o algoritmo é para encontrar um caminho no grafo. O valor obtido é multiplicado pelo valor da conveniência η_{xy} para o movimento entre x e y. Para este caso, a conveniência do movimento é calculada pelo inverso do valor do tempo $\left(\frac{1}{t_{xy}}\right)$ que é necessário para percorrer o caminho entre x e y. O valor obtido é elevado a β, que também é um parâmetro heurístico do algoritmo. Este parâmetro descreve a rapidez com que as formigas vão convergir para uma solução estável. Para obter uma probabilidade de transição de um vértice para o outro, o cálculo efetuado é dividido pela soma das restantes possíveis transições de estado, ou seja, pela soma dos valores calculados para restantes possíveis transições de um ponto de origem para outros pontos não visitados.

Todos os cálculos apresentados até ao momento são efetuados por cada uma das formigas que correm em *threads* independentes. Existem estruturas de dados partilhadas, como é o caso da matriz de distâncias entre pontos de interesse e da matriz de pontuações entre vértices do grafo. Ambas as matrizes são preenchidas antes da criação das *threads* exploradoras, sendo imutáveis do início ao fim da sua execução. Por só servirem para consulta, não houve a necessidade de utilização de estruturas de dados *thread-safe* para estes dois casos. No entanto, foi necessária a criação de uma terceira matriz com o intuito de conter os valores resultantes da atualização das feromonas por parte das tarefas que correm em

threads diferentes. Inicialmente essa matriz foi preenchida com os mesmos valores da matriz das pontuações divididos por cinco. Este cálculo é justificado pela escala de pontuações variar entre zero e cinco. Ao se dividir o valor da pontuação pela variação da escala, obtém-se uma probabilidade de escolha, que varia entre zero e um. Por existir a possibilidade de atualização dos valores da matriz por diversas *threads* concorrentes, utilizou-se uma matriz de AtomicDouble para manter a consistência dos dados. Sempre que as formigas escolhem um novo ponto a visitar, o valor da feromona é atualizado para esse ponto, recorrendo à fórmula apresentada em seguida.

$$\tau_{xy}^{k} = (1-\rho)\tau_{xy}^{k} + \Delta\tau_{xy}^{k} \tag{6}$$

Onde:
k: Número de ordem da formiga (1ª, 2ª, 3ª …)
x: Ponto de origem
y: Ponto seguinte
τ_{xy}^{k}: Quantidade de feromonas depositadas no caminho entre x e y
ρ: Coeficiente de evaporação da feromona, varia entre 0 e 1
$\Delta\tau_{xy}^{k}$: Quantidade de feromonas depositadas pela formiga k no caminho entre x e y

O cálculo de $\Delta\tau_{xy}^{k}$ é também obtido com o recurso a uma fórmula:

$$\Delta\tau_{xy}^{k} = \begin{cases} \frac{Q}{L_k} & se\ a\ formiga\ k\ usar\ o\ caminho\ xy\ na\ sua\ solução \\ 0 & se\ a\ formiga\ k\ não\ usar\ o\ caminho\ xy\ na\ sua\ solução \end{cases} \tag{7}$$

Onde:
Q: Valor constante
L_k: Tempo total levado para percorrer o caminho encontrado pela formiga k

Cada formiga começa a percorrer uma rota a partir do ponto inicial definido pelo utilizador e calcula um possível melhor caminho, ao utilizar as fórmulas acima expressas. Ao transitar pelos vários pontos no seu trajeto a formiga vai marcando esses locais, de forma a não voltar a visitá-los. Caso o tempo limite da viagem não tenha sido ultrapassado e o local esteja aberto para visita, a formiga adiciona esse ponto à lista de sítios visitados e atualiza a matriz de feromonas para ajudar as restantes a encontrar o melhor caminho.

De forma a permitir um desempenho equilibrado e eficaz do algoritmo, foram experimentados alguns valores para as constantes definidas como parâmetro. Após terem sido tentados alguns conjuntos diferentes de valores, chegou-se à conclusão de que 15000 formigas seriam suficientes para obter resultados de pesquisa satisfatórios e que as restantes constantes deveriam adotar os seguintes valores:

$$\alpha = 0;\ \beta = 9\ ;\ Q = 0.0001;\ \rho = 0.5$$

O Algoritmo 2 demonstra a implementação em pseudocódigo das etapas executadas por cada formiga para encontrar um caminho candidato.

Algoritmo 2 – Algoritmo a executar por cada formiga no ACO (pseudocódigo)

```
1: Input: startNode - Start Node; nodesCount - Number of Nodes;
   deltaTime - Time to complete the route; acoParameters - ACO constant parameters;
   timeMatrix - Matrix with times between nodes;
   distanceMatrix - Matrix with distances between nodes;
   scoresMatrix - Matrix with scores for each edge;
   scoresPhoromoneMatrix - Matrix with scores updated with ants phoromones
2: Output: antPath - Path found by the ant
3: antPath ← ∅
4: toVisit ← nodesCount
5: time ← timeMatrix[startNode][startNode]
6: distance ← distanceMatrix[startNode][startNode]
7: score ← scoresMatrix[startNode][startNode]
8: while (nextNode ← getNextProbableNode(lastNode)) ≠ −1 do
9:   timeAux ← timeMatrix[lastNode][nextNode]
10:  if (timeAux ≤ deltaTime) ∧ (nextNode.isOpened(startTime + time)) then
11:    add lastNode to antPath
12:    time ← timeAux
13:    distance ← distance + distanceMatrix[startNode][startNode]
14:    score ← score + scoresMatrix[startNode][startNode]
15:    decay ← acoParameters.getQ()/time
16:    scoresPheromoneMatrix.adjustScore(lastNode,nextNode,decay)
17:    visited[nextNode] ← true
18:    lastNode ← nextNode
19:  end if
20:  toVisit ← toVisit − 1
21: end while
22: return antPath
```

Caso a formiga encontre um caminho com uma pontuação igual ou superior à encontrada até esse momento, o valor e o respetivo caminho são registados para posterior comparação com os resultados obtidos pelas restantes formigas. O controlo de execução das diversas *threads* é efetuado através do ExecutorCompletionService do Java. Foi criada uma *thread pool* de tamanho fixo igual ao número de núcleos da CPU disponibilizados à Java Virtual Machine. Inicialmente são criadas e submetidas milhares de "formigas" ou tarefas para o *executor service,* que trata de criar uma fila ordenada de execução (Figura 27). À medida que as tarefas vão sendo concluídas, os resultados são armazenados numa lista e disponibilizados aos seus subscritores. Desta forma, consegue-se maximizar o número de tarefas a correr em simultâneo, equilibrando essa execução pelo número de núcleos disponibilizados ao processo.

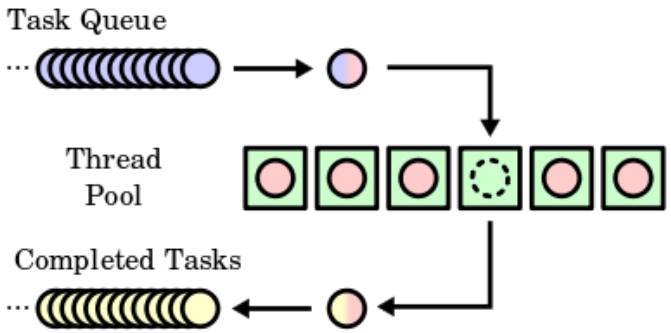


Figura 27 – Exemplo do funcionamento de uma *thread pool*
(Gupta, 2015)

Após concluir a execução de uma tarefa, cada *thread* vai à fila de tarefas restantes e executa a seguinte. Este processo repete-se até todas as tarefas serem executadas. Quando a totalidade das *threads* lançadas conclui o seu trabalho, o algoritmo devolve a rota descoberta com a maior pontuação encontrada para as preferências do turista.

4.2.4 Resultados Obtidos

Os resultados obtidos pela integração do módulo de geração de rotas no sistema TheRoute podem ser consultados através da aplicação móvel. Esta integração foi efetuada tendo em conta as considerações definidas na secção da arquitetura da solução (3.2). Após uma autenticação bem-sucedida no sistema, o utilizador tem a possibilidade de visualizar as suas rotas ou gerar novas. Podem ser criadas rotas com trajetos preestabelecidos ou, em alternativa, rotas personalizadas tendo em conta as suas preferências.

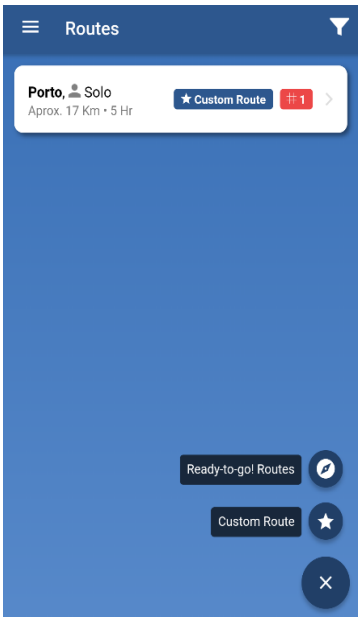


Figura 28 – Ecrã principal com menu de escolha do tipo de rota

As rotas preestabelecidas são introduzidas manualmente no sistema, não sendo influenciadas pelo módulo desenvolvido. No caso de o utilizador escolher uma rota personalizada, existe a possibilidade de optar por uma rota individual ou por uma rota em grupo.

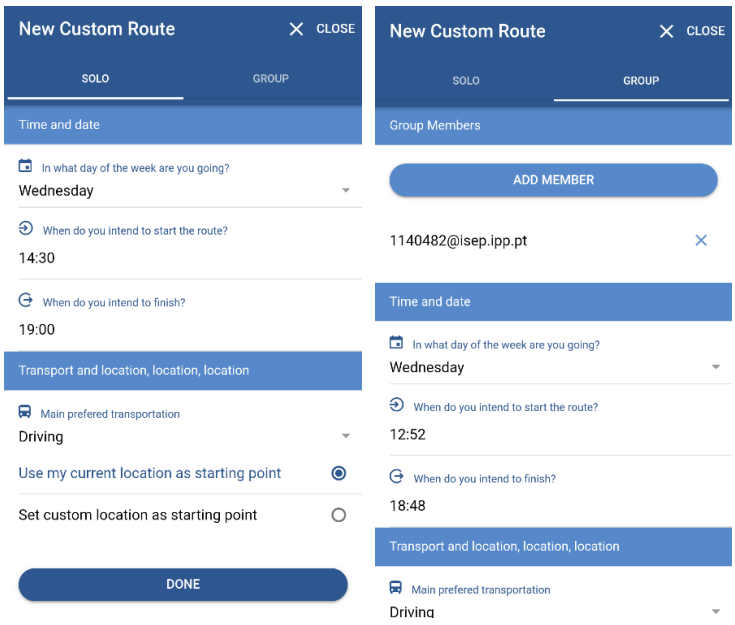


Figura 29 – Ecrãs de escolha para rota personalizada individual e em grupo

Depois do utilizador introduzir os dados solicitados, estes são enviados para a aplicação servidor que, por sua vez, os encaminha para os componentes responsáveis pela criação da recomendação turística. Após alguns segundos, a rota é calculada e enviada para a aplicação móvel, onde o turista pode consultar o plano de visita detalhado, assim como visualizar no mapa a disposição geográfica dos pontos de interesse a percorrer.

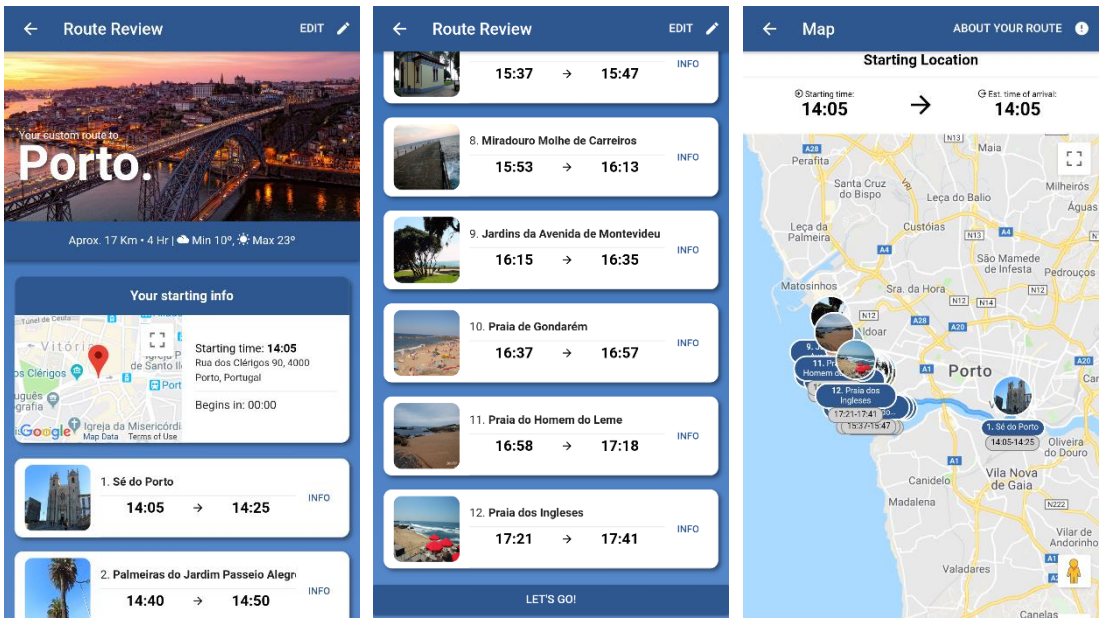


Figura 30 – Ecrãs de visualização da rota recomendada

5 Experimentação e Avaliação

Este capítulo apresenta a metodologia utilizada na experimentação e avaliação da solução proposta nesta dissertação. A secção 5.1 apresenta os testes realizados para a validação da implementação e a secção 5.2.1 descreve as métricas que foram utilizadas como medidas de avaliação da solução. As experiências realizadas para suportar as hipóteses definidas em 1.2 podem ser consultadas na secção 5.2.

5.1 Testes

Para garantir a qualidade e assertividade do módulo desenvolvido, foram elaborados alguns testes. Uma vez que um dos objetivos desta dissertação é a escolha do algoritmo mais adequado para a geração de rotas com base na análise do contexto envolvente, foram testados os algoritmos apresentados em 4.2 com um conjunto de cenários conhecidos. Os tipos de teste realizados são detalhados nas secções seguintes.

5.1.1 Testes Unitários

Cada um dos algoritmos estudados foi testado com o auxílio de testes unitários, tendo como objetivo atestar a validade dos métodos de pesquisa desenvolvidos. A ferramenta utilizada para desenvolver e executar os testes foi o JUnit. Com vista a simular o comportamento dos objetos de entrada nos algoritmos de geração de rotas, foram introduzidos objetos *mock* criados à imagem dos modelos de análise de contexto. Desta forma, assegura-se de uma forma controlada o funcionamento dos algoritmos face à introdução de objetos com comportamentos idênticos aos esperados no mundo real. Esta abordagem possibilita a experimentação das diversas soluções consideradas, independentemente da existência de dados reais. Com a realização deste tipo de teste é possível experimentar cada método isoladamente, não sendo necessária uma integração imediata no produto final.

Por não se conseguir prever antecipadamente as soluções geradas por cada uma das implementações, os testes realizados consistem na verificação dos resultados obtidos. A Figura 31 demonstra um exemplo de teste unitário realizado à função *search* do algoritmo A*, tendo em conta a heurística para o maior resultado.

```
@Test
public void searchSuccessTest() {
    IHeuristic heuristicToUse = new MaxScoreHeuristic();
    ISearchAlgorithm searchAlgorithm = new AStar(graph, nearestNode,dataset.getStartDate(), dataset.getEndDate()
            ,maximumCostSeconds,heuristicToUse,COST_OF_EACH_TIME_BLOCK);
    long startExecutionTime = System.currentTimeMillis();
    SearchAlgorithmResult result = searchAlgorithm.search();
    long endExecutionTime = System.currentTimeMillis();
    long executionTimeSecs = (endExecutionTime-startExecutionTime)/1000;
    //it should return some data
    assertNotNull(result);
    //it should return a path
    assertTrue( condition: result.getSegments().size()>0);
    //check if the algorithm has been executed in time
    assertTrue( condition: executionTimeSecs <= MAX_EXECUTION_TIME_SECONDS);
    //check the consistency of the results
    List<Segment> segments = new ArrayList<>(result.getSegments());
    Segment firstSegment = segments.get(0);
    Segment lastSegment = segments.get(segments.size()-1);
    assertEquals( expected: lastSegment.endTime-firstSegment.startTime,result.getTotalCost(), delta: 0);

    //collects the values from the algorithm parameters, makes all the the necessary calculations and compares
    //the values with the ones that were returned by the algorithm
    ensureCostValuesAreCorrect(segments,result);
}
private void ensureCostValuesAreCorrect(List<Segment>segments, SearchAlgorithmResult result){
    List<PoiDailyScheduleDTO> poisInfo = dataset.getPois() List<PoiDailyScheduleDTO>
            .stream() Stream<PoiDailyScheduleDTO>
            .filter(poi -> segments.stream()
                    .anyMatch(segment -> poi.getPoiID()==segment.poiId)) Stream<PoiDailyScheduleDTO>
            .collect(Collectors.toList());
    double expectedScore = poisInfo.stream().mapToDouble(x->x.getAverageCategoryValue()).sum();
    //test if the score is correct for the result
    assertEquals(expectedScore,result.getTotalScore(), delta: 0);

    int expectedDistance=0, expectedTime=0;
    for (int i = 0;i<segments.size()-1;i++) {
        int currentNodeId = segments.get(i).poiId;
        int nextNodeId = segments.get(i+1).poiId;
        DistanceDTO distanceDTO = dataset.getDistances() List<DistanceDTO>
                .stream() Stream<DistanceDTO>
                .filter(distance -> ((distance.getPoi1Id()==currentNodeId && distance.getPoi2Id()==nextNodeId) ||
                        (distance.getPoi1Id()==nextNodeId && distance.getPoi2Id()==currentNodeId))) Stream<DistanceDT
                .findFirst() Optional<DistanceDTO>
                .orElseGet( other: null);
        expectedDistance += distanceDTO.getDistance();
        expectedTime += distanceDTO.getDuration()+ (nextNodeId == distanceDTO.getPoi1Id()?
                distanceDTO.getPoi1VisitDuration():distanceDTO.getPoi2VisitDuration())*60;
        if(i==0){
            expectedTime+=(currentNodeId == distanceDTO.getPoi1Id()?
                    distanceDTO.getPoi1VisitDuration():distanceDTO.getPoi2VisitDuration())*60;
        }
    }
    //check if the expected time and distance costs match with the ones returned by the algorithm
    assertEquals(expectedDistance,result.getTotalDistance(), delta: 0);
    assertEquals(expectedTime,result.getTotalCost(), delta: 0);
}
```

Figura 31 – Exemplo de teste unitário

Este teste valida se os dados relativos à rota gerada são consistentes e se a solução é encontrada num tempo inferior ao máximo estipulado. Para os *inputs* definidos espera-se que

o algoritmo encontre um trajeto com valores de pontuação, tempo de visita e distância percorrida válidos. Para verificação da solução obtida pelo algoritmo foram executados individualmente todos os cálculos relativos a estas variáveis, tendo-se atingido os mesmos resultados.

5.1.2 Testes de Integração

Após a integração do módulo de geração de rotas com os restantes componentes do sistema TheRoute foram realizados testes para garantir a estabilidade da solução. Com o objetivo de assegurar que essa integração no sistema final foi concluída com sucesso, foram efetuados alguns testes para confirmar que o módulo desenvolvido se comporta como esperado nas suas interações com os restantes componentes do sistema. A realização dos testes de integração pode encontrar erros que dificilmente são detetáveis nos testes unitários, uma vez que englobam uma maior cobertura de código.

A Figura 32 exemplifica um teste de integração efetuado ao componente RouteDataAggregator.

```
@Test
public void collectRouteInputDataTest() {
    RouteRequestDTO request = inputs.get(0);
    RouteDataAggregator routeDataAggregator = new RouteDataAggregator(request);
    RouteInputDataDTO result = routeDataAggregator.collectRouteInputData();
    //it should return some data
    assertNotNull(result);

    //check if arguments in input are consistent with the built object
    assertEquals(request.getLatitude(),result.getCurrentLatitude(), delta: 0.0f);
    assertEquals(request.getLongitude(),result.getCurrentLongitude(), delta: 0.0f);
    assertEquals(request.getStarting_time(),result.getStartDate(), delta: 0.0f);
    assertEquals(request.getLimit_time(),result.getEndDate(), delta: 0.0f);

    //it is expected more than 1 poi for this input
    int numberOfPois = result.getPois().size();
    assertNotEquals( unexpected: 0,numberOfPois);

    //it is expected for N nodes: (N-1)+(N-2)+(N-3)+...+1 one way edges
    int edgeCount=0;
    for (int i=numberOfPois-1;i>0;i--){
        edgeCount+=i;
    }
    assertEquals(result.getDistances().size(),edgeCount, delta: 0);
}
```

Figura 32 – Exemplo de teste de integração

Este teste verifica o componente RouteDataAggregator, que é o responsável por recolher os dados de entrada para o algoritmo de geração de rotas. É expectável que alguns desses dados (latitude, longitude, tempo de início e tempo de fim da rota) sejam incluídos nos resultados da seleção efetuada, por serem relevantes para o algoritmo de geração de rotas. É também

verificada a relação entre o número de pontos de interesse recolhidos e as ligações estabelecidas entre si, de forma a garantir que não faltam dados essenciais à geração da rota.

5.1.3 Testes de Desempenho

Os testes de desempenho incidem sobre os algoritmos implementados e têm um papel importante na escolha do algoritmo que suportará o módulo de geração de rotas. São utilizadas as métricas definidas na secção 5.2.1 para testar o desempenho individual de cada uma das opções encontradas. O algoritmo a utilizar deve produzir uma solução o mais próxima possível dos gostos do utilizador, tendo também em conta outros aspetos provenientes das restrições do ambiente envolvente. No entanto, deve ser capaz de encontrar essa recomendação num período de tempo razoável. Após o estudo das soluções e a execução dos testes de desempenho, ficam reunidas as condições necessárias para a escolha do candidato ideal à execução da tarefa a que o módulo de geração de rotas se destina.

A secção 5.2 expõe em detalhe os testes de desempenho efetuados e a avaliação dos algoritmos desenvolvidos, comparativamente a diferentes dados de entrada.

5.2 Experiências

As hipóteses apresentadas são verificadas através da validação e do desempenho dos algoritmos estudados. Esta validação é efetuada através da realização de testes e simulações.

Numa primeira fase, as simulações são executadas fora do contexto aplicacional, ou seja, são efetuadas simulações ao módulo desenvolvido de uma forma independente do resto do sistema. Posteriormente, após a integração do módulo no contexto do sistema, são efetuadas avaliações ao seu comportamento quando em contacto com os restantes elementos constituintes da solução.

Com vista a avaliar a consistência e o desempenho do módulo desenvolvido são também recolhidos dados de teste, semelhantes aos esperados em condições reais. Esses dados servem para criar objetos e bases de dados fictícios (*mock*) que, por sua vez, atuam como entrada para os algoritmos utilizados. A introdução destes dados serve para ter uma ideia inicial do tempo necessário para a criação de uma solução válida para o utilizador, ajudando a avaliar o desempenho do algoritmo em questão.

Com o objetivo de experimentar e avaliar as soluções desenvolvidas, foram efetuadas várias simulações tendo em conta diferentes dados de entrada. Os testes realizados têm em consideração algumas condicionantes inerentes ao ambiente em que foram efetuados. Por condicionante entende-se todo o fator que possa influenciar os resultados obtidos nos testes. Em seguida são enumerados alguns desses fatores e a sua importância para a qualidade dos resultados:

- *Hardware* – As experiências foram efetuadas num computador portátil equipado com um processador Intel Core i7-3635QM 2.40GHz e 16GB de RAM. O processo para a geração de rotas é bastante exigente a nível de recursos, pelo que a capacidade de processamento e memória são aspetos importantes a considerar. À partida, o servidor ou servidores onde o módulo de geração de rotas será implantado terão maior capacidade de processamento que um computador portátil. Se os resultados obtidos nesta máquina forem razoáveis, infere-se que os resultados serão ainda mais satisfatórios num ambiente com mais recursos;

- Quantidade de dados a analisar – A base de dados utilizada para testes possui um total de 266 pontos de interesse sendo, por isso, o limite máximo de nós a considerar no grafo. Como todos os pontos de interesse têm ligação entre si, sendo esses caminhos bidirecionais, o total de caminhos existentes entre cada dois pontos (arestas do grafo) é, no máximo, 70490. É possível calcular o número de arestas do grafo (E) em função do número de nós a considerar (n) através da fórmula apresentada em (8);

$$E = 2 * \sum_{i=1}^{n-1} (n - i) \tag{8}$$

- Perfil do utilizador de testes – Foi criado um utilizador e o respetivo perfil de forma a permitir simular o caso de uso da criação de uma rota. O perfil e gostos do utilizador influenciam a filtragem dos pontos de interesse a considerar para a pesquisa. Por essa razão, os resultados obtidos também são condicionados por este fator;

- Dados de entrada do algoritmo – Dados introduzidos pelo utilizador ou induzidos pelo seu dispositivo móvel, assim como os dados meteorológicos e as constantes definidas nos algoritmos são também determinantes para a recomendação final.

Para efetuar uma boa análise dos resultados obtidos é importante perceber que alguns dos fatores mencionados anteriormente devem ser comuns a todos os testes, enquanto outros podem variar. O *hardware* e a população de dados a utilizar devem ser sempre os mesmos em todas as simulações, de maneira a garantir que não se está a beneficiar nenhuma das soluções algorítmicas em detrimento das restantes. No caso dos perfis de utilizador, os testes efetuados poderiam incidir sobre mais do que um. No entanto, para garantir as mesmas condições na seleção inicial dos pontos de interesse, decidiu-se manter o mesmo perfil de utilizador durante a execução dos testes. Como referido em 4.2.2, os dados relevantes do perfil do utilizador para efeitos de seleção de pontos de interesse são as suas pontuações relativas a cada categoria turística. A Tabela 2 contém os valores de cada categoria para o utilizador a considerar. Quanto mais elevado o valor, maior a probabilidade de o turista visitar pontos de interesse pertencentes a essa categoria.

Tabela 2 – Valores de cada categoria turística para o utilizador de teste

Categoria Turística	Valor (0-5)
Alojamento	1
Arquitetura	3
Científico	3
Comércio	1
Cultura	2
Desporto	4
Entretenimento	3
Estádio de Futebol	5
Fitness e Saúde	3
Histórico	3
Natureza	4
Religião	4
Restaurante / Café	3
Teatro	2
Trending	2

As variantes a considerar nos testes a realizar são os parâmetros de entrada do algoritmo enviados pela aplicação móvel. Cada vez que o turista solicita a criação de uma rota são enviados dados para o servidor que permitem restringir os pontos de interesse a analisar, tanto antes como no decorrer da execução do algoritmo. Os dados que constam no pedido são: local onde o turista se encontra (latitude e longitude); dia da semana no qual pretende efetuar o trajeto; hora de início e hora de término do passeio; meio de deslocação (a pé ou de carro); identificador do perfil de utilizador. Tendo em conta a diversidade de valores que podem ser enviados por um utilizador da aplicação, foram definidos alguns conjuntos de dados (*datasets*) para posterior análise em função das métricas de avaliação. Cada um destes *datasets* é constituído por uma lista de pontos de interesse e pelos respetivos tempos de deslocação entre si. Este conteúdo é conseguido através de triagens efetuadas aos dados persistidos, como mencionado em 4.2.2. A forma como os algoritmos calculam os caminhos é indiferente ao tipo de viagem que se pretende (viagem individual ou em grupo), razão pela qual se optou por experimentar apenas viagens individuais.

Foram escolhidas aleatoriamente quatro localizações diferentes para serem utilizadas como pontos de partida nos cenários de teste idealizados. A Tabela 3 contém as definições dos pontos considerados, tendo em conta as suas coordenadas geográficas (latitude e longitude) e a descrição da sua localização.

Tabela 3 – Pontos de partida a utilizar nos cenários de teste

Nome	Latitude	Longitude	Local
Ponto A	41.1483136	-8.6130035	Praça D. Filipa de Lencastre
Ponto B	41.1456697	-8.6127193	Rua Estreita dos Lóios
Ponto C	41.1434747	-8.6135366	Rua de Mouzinho da Silveira
Ponto D	41.1483486	-8.6276202	Massarelos

Todos os pontos geográficos considerados situam-se na cidade do Porto, local onde se encontram grande parte dos pontos de interesse guardados na base de dados. No entanto, estão dispersos pela cidade, o que permite selecionar pontos de interesse diferentes para o começo de cada rota calculada, como explicado em 4.2.3.

A Tabela 4 possui a definição dos cenários de teste idealizados para definir os dados de entrada a utilizar na criação dos *datasets*.

Tabela 4 – Cenários utilizados na criação dos *datasets*

Nome	Partida	Dia Semana	Hora Início	Hora Fim	Meio Transporte
Cenário 1	Ponto A	Quarta-feira	14h00	18h00	Carro
Cenário 2	Ponto B	Quarta-feira	12h30	20h30	Carro
Cenário 3	Ponto C	Quarta-feira	09h00	21h00	Carro
Cenário 4	Ponto D	Quarta-feira	00h00	23h59	Carro
Cenário 5	Ponto A	Quarta-feira	14h00	18h00	A pé

O dia da semana foi escolhido aleatoriamente, mas é o mesmo para todos os cenários de forma a garantir que os horários de abertura e fecho de cada ponto de interesse não se alteram durante a realização das experiências. Os tempos para o início e o fim da visita diferem para se conseguir obter conjuntos diferentes de pontos de interesse, uma vez que nem todos têm o mesmo horário de funcionamento.

O Cenário 4 possui uma janela temporal de visita um pouco excessiva, na medida em que dificilmente um turista opta por realizar um trajeto de 24 horas. No entanto, como a aplicação móvel o permite, tornou-se necessário verificar o comportamento dos algoritmos para esta situação extrema. O Cenário 5 é bastante semelhante ao Cenário 1, diferindo apenas no meio de transporte. A semelhança foi propositada para verificar as diferenças existentes na solução final, para diferentes meios de deslocação.

Cada um dos cenários referidos gera uma compilação de dados específica. Por se tratar de uma grande quantidade de dados, os *datasets* apresentados na Tabela 5 são exibidos de forma simplificada, mostrando apenas o conteúdo relevante à análise e experimentação a efetuar. Os tempos de visita encontram-se expressos em minutos.

Tabela 5 – *Datasets* utilizados nas experiências (vista simplificada)

Descrição	Nº POIs	Nº ligações entre POIs	Tempo mínimo visita	Tempo máximo visita	Tempo médio visita
***Dataset* 1**	92	8372	4	40	22.326
***Dataset* 2**	92	8372	4	40	22.326
***Dataset* 3**	94	8742	4	60	22.702
***Dataset* 4**	177	31152	4	180	23.237
***Dataset* 5**	92	8372	4	40	22.326

Como é possível verificar pela média dos tempos de visita em cada um dos *datasets*, os pontos de interesse possuem maioritariamente tempos de visita bastante curtos. Por essa razão, as rotas geradas irão possuir um maior número de pontos de interesse a visitar. Por outro lado, tempos de visita mais curtos dificultam a pesquisa do algoritmo, visto que é possível testar mais opções para o limite de tempo da viagem. É expectável que para dados reais os tempos de visita sejam mais extensos, mas utilizar tempos mais curtos torna o teste de verificação do comportamento dos algoritmos ainda mais exigente.

O trabalho desenvolvido em paralelo a esta dissertação, no âmbito da experimentação de algoritmos genéticos não foi concluído em tempo útil para fornecer resultados para comparação. Desta forma, os algoritmos a testar são três, uma vez que foram desenvolvidas duas soluções para o A* e uma para o *Ant Colony Optimization*. De maneira a facilitar o tratamento e análise dos resultados apresentados, são utilizadas abreviaturas para os nomes dos algoritmos a analisar. Assim sendo, os algoritmos passam a ser mencionados recorrendo à sua sigla, ou seja, o *Ant Colony Optimization* passa a ser denominado **ACO** e o A* intitulado **AS**. Como são testadas duas heurísticas diferentes para o AS, a heurística para a escolha da maior pontuação vai ser designada **AS-Max** e a heurística baseada no *Simmulated Annealing* denominada **AS-SA**. Para cada um dos 5 *datasets* a testar, foram efetuadas 30 simulações para os 3 algoritmos, o que totaliza 450 simulações. Os dados foram recolhidos e guardados num ficheiro CSV para serem posteriormente analisados com o auxílio da ferramenta Microsoft Office Excel.

5.2.1 Métricas de Avaliação

As métricas de avaliação a considerar estão relacionadas com a complexidade temporal do algoritmo e com o custo de tempo necessário para o turista concluir o trajeto. Cada um dos algoritmos necessita de efetuar um diverso conjunto de operações que se reflete no tempo que demora a retornar uma solução válida. Para medir a eficiência e desempenho de cada algoritmo, são realizados vários testes para conjuntos de dados definidos. O desempenho é testado pela relação qualidade-custo dos resultados, ou seja, pela qualidade da solução produzida no tempo de execução do algoritmo.

O algoritmo a utilizar deve maximizar o valor do somatório das pontuações de cada ponto de interesse. No entanto, o custo de cada caminho tem de ser avaliado antes desta

contabilização. Essa avaliação é importante, uma vez que o utilizador escolhe um intervalo de tempo no qual decorre a sua visita e esse tempo não pode ser ultrapassado. A função que determina o tempo necessário para o turista percorrer determinada rota encontra-se expressa na fórmula seguinte.

$$t_{total} = t_1 + \sum_{x=1}^{n-1} t_{x \to x+1} + t_{x+1} \quad (9)$$

Onde:
t_1: Tempo necessário para visitar o primeiro ponto de interesse da rota
n: Número total de pontos de interesse a visitar
$t_{x \to x+1}$: Tempo necessário de deslocação de um ponto de interesse para o seguinte
t_{x+1}: Tempo necessário para visitar o ponto de interesse seguinte da rota

O custo total da rota é definido pela soma do tempo necessário para visitar o primeiro nó com o somatório dos valores considerados para cada aresta do grafo. Por sua vez, o custo de cada aresta do grafo tem em consideração o valor temporal de deslocamento de um nó para o seguinte e o tempo que o nó de destino demora a ser visitado. Este cálculo é transversal aos algoritmos desenvolvidos e tem um grande impacto na escolha do caminho final a apresentar, uma vez que é uma das condições de paragem da pesquisa. Em teoria, quantos mais percursos entre nós existirem no caminho final, maior será a sua pontuação. No entanto, a quantidade de pontos de interesse pode não ser suficiente para encontrar o melhor trajeto para o turista. O algoritmo de pesquisa tem uma grande responsabilidade neste aspeto, porque é ele que seleciona o trajeto a seguir, de iteração em iteração. Se a seleção ocorrer de forma errada, o resultado final é certamente prejudicado. A escolha de pontos de interesse com pontuação fraca relativamente aos restantes considerados para a pesquisa é um dos fatores que inviabiliza a hipótese de o caminho com mais pontos de interesse ser o que possui a melhor pontuação. Assim sendo, a pontuação dos locais a visitar influencia a satisfação do turista face ao resultado da recomendação. Essa satisfação é calculada através da fórmula expressa em (10), tendo em conta os n pontos de interesse visitados. No caso de se tratar de um trajeto em grupo, a fórmula de cálculo é a apresentada em (11).

$$S_{turista} = \sum_{i=1}^{n} PontuacaoIndividualPOI \quad (10)$$

$$S_{turistas} = \sum_{i=1}^{n} PontuacaoGrupoPOI \quad (11)$$

O cálculo das pontuações para cada ponto de interesse encontra-se definido no ponto 3 da secção 4.2.2.

A qualidade do algoritmo pode também ser verificada pelo aproveitamento do tempo introduzido pelo utilizador para a realização da rota, ou seja, o trajeto gerado deve

proporcionar a maior taxa de ocupação possível ao turista. Essa taxa pode ser apresentada em percentagem através da fórmula (12).

$$aproveitamento\ (\%) = \frac{t_{rf} - t_{ri}}{t_{df} - t_{di}} * 100 \qquad (12)$$

Onde:

t_{ri}: Tempo real obtido para o começo da rota
t_{rf}: Tempo real obtido para o fim da rota
t_{di}: Tempo definido para o começo da rota
t_{df}: Tempo definido para o fim da rota

Nas próximas secções são apresentadas as análises às experimentações efetuadas para cada uma das métricas a considerar.

5.2.2 Cenário Tempo de Execução

Como tem vindo a ser referido ao longo do documento, o tempo de execução do algoritmo é um dos fatores determinantes para a viabilidade da solução a incorporar no TheRoute. Para uma melhor análise comparativa dos dados recolhidos foram gerados alguns gráficos, um para cada *dataset* criado. Em cada caso, são apresentados os tempos de execução em segundos para os algoritmos testados nas 30 execuções efetuadas.

A Figura 33 demonstra os tempos de execução para cada um dos algoritmos, tendo como dados de entrada os recolhidos no *Dataset* 1.

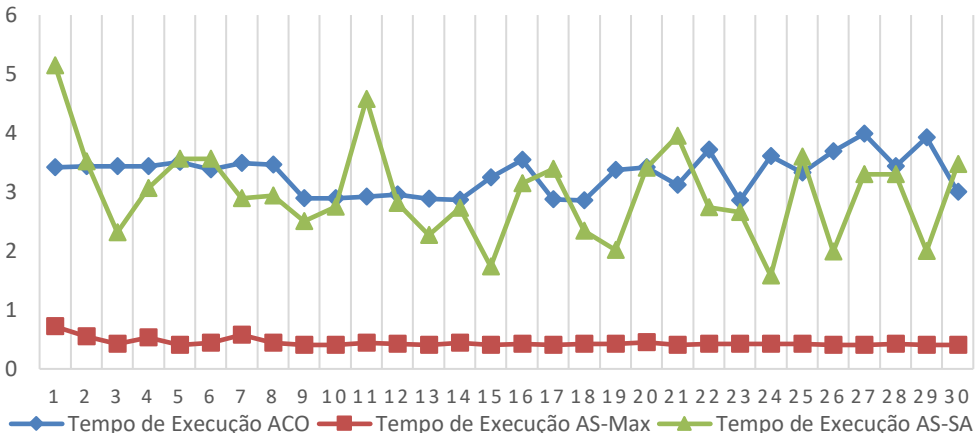


Figura 33 – Tempos de execução dos algoritmos para o *Dataset* 1

Tendo em conta um tempo de viagem máximo de 4 horas e um total de 92 pontos de interesse, o algoritmo que possui um tempo de execução mais favorável e uniforme é o AS-Max. Com um tempo inferior a 1 segundo em todas as execuções, este algoritmo revela ser bem mais eficiente que os restantes. O AS-SA é o que possui as maiores oscilações de tempo, superando os 5 segundos na sua execução mais lenta. O ACO apresenta resultados sempre

próximos dos 3 segundos. Uma vez que é a eficiência em encontrar um caminho válido que está a ser avaliada para esta métrica, o algoritmo AS-Max é, sem qualquer dúvida, o que revela melhor desempenho para o conjunto de dados testado no *Dataset* 1. Os caminhos gerados pelos algoritmos para este *dataset* são os mais curtos, visto que foram estipuladas apenas 4 horas para o concluir.

Apesar de se terem comprovado valores bastante atrativos para tempos de viagem reduzidos, convém verificar se para os restantes casos estipulados esses valores se mantêm próximos dos obtidos ou se sobem abruptamente. Ao considerar um tempo mais extenso de viagem, 8 horas e os mesmos 92 pontos de interesse, obteve-se o gráfico (Figura 34) para os tempos de execução com dados do *Dataset* 2.

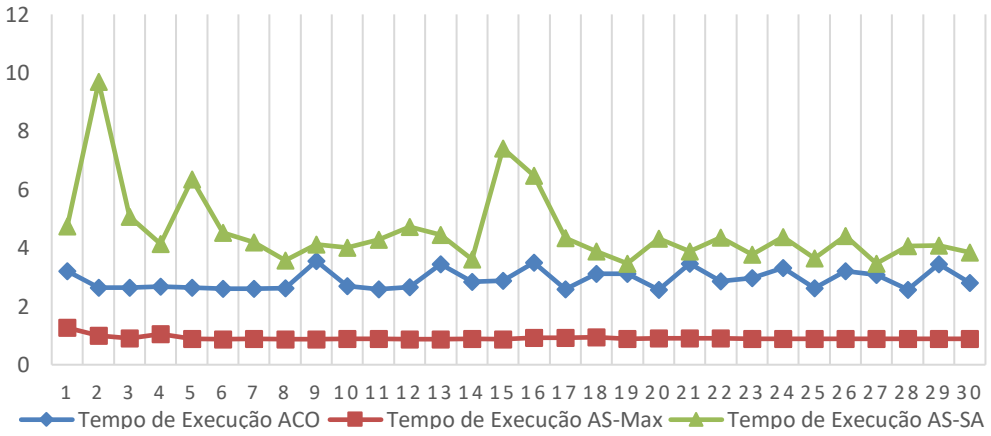


Figura 34 – Tempos de execução dos algoritmos para o *Dataset* 2

Pela análise dos valores apresentados, consegue-se concluir que o algoritmo com melhor desempenho para o *Dataset* 2 é novamente o AS-Max. Mais uma vez, todas as suas execuções ocorrem em tempos que rondam 1 segundo, o que é inferior a qualquer um dos tempos conseguidos pelos restantes algoritmos. Por sua vez, o ACO surge em segundo lugar com tempos de execução próximos aos conseguidos para o *Dataset* 1, sempre a rondar os 3 segundos. O AS-SA é o que revela inequivocamente os piores resultados, atingindo novamente diferenças elevadas entre o valor da sua melhor e pior execução. Na sua pior execução, o tempo necessário para calcular o caminho resultante chega a atingir valores próximos dos 10 segundos.

Como seria de esperar, os valores obtidos para o dobro do tempo de viagem já revelaram algumas diferenças relativamente ao cálculo de rotas para janelas temporais mais reduzidas. É agora altura de verificar se essas disparidades aumentam com um tempo de viagem ainda mais elevado. Para o *Dataset* 3 esse valor é fixado nas 12 horas, existindo um incremento de dois pontos de interesse relativamente ao *dataset* anterior, ou seja, a pesquisa vai incidir sobre 94 pontos de interesse.

A Figura 35 contém os resultados dos tempos de execução dos algoritmos para o *Dataset* 3.

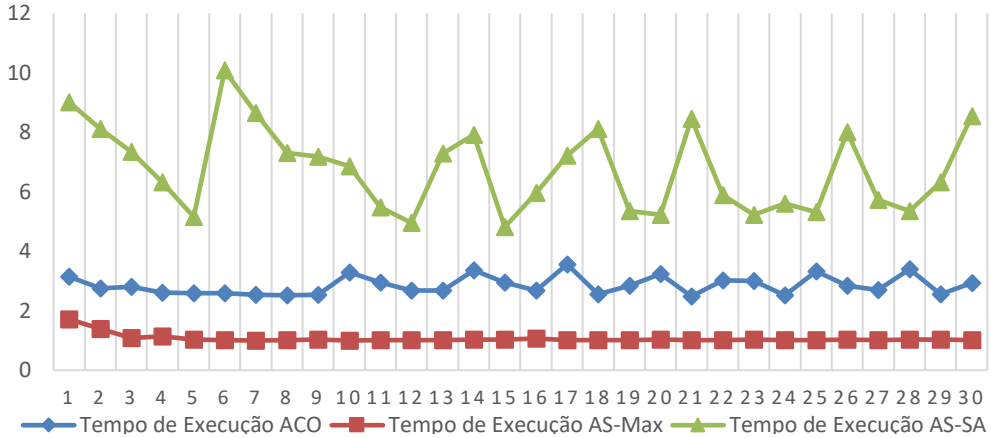


Figura 35 – Tempos de execução dos algoritmos para o *Dataset* 3

Comparativamente aos tempos de execução verificados para o *Dataset* 2, não ocorreram alterações muito significativas para este *dataset*. Apesar de o tempo para percorrer o percurso e o número de nós e arestas a considerar no grafo ter aumentado, a execução dos algoritmos foi semelhante ao caso anterior. O AS-Max volta a ser o algoritmo mais rápido a devolver uma solução, subindo ligeiramente os valores temporais da sua execução, mas nunca atingindo os 2 segundos. O ACO continua a rondar valores na casa dos 3 segundos e o AS-SA sofreu uma subida ainda considerável relativamente aos seus picos mínimos e máximos. No sexto ensaio chegou mesmo a ultrapassar a barreira dos 10 segundos de execução.

O grande teste à rapidez de execução é efetuado para o *Dataset* 4, onde são considerados 177 pontos de interesse passíveis de serem visitados num espaço de 24 horas. A Figura 36 apresenta o gráfico conseguido para os tempos de execução dos algoritmos, tendo em conta os dados do *dataset* referido.

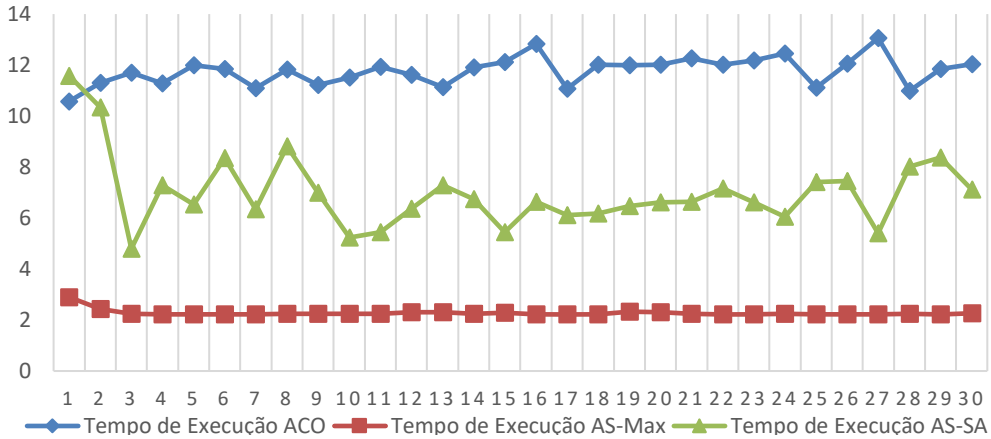


Figura 36 – Tempos de execução dos algoritmos para o *Dataset* 4

Como habitual, o algoritmo que respondeu mais rapidamente ao pedido para geração de uma rota foi o AS-Max, não ultrapassando os 3 segundos em nenhuma das execuções. Entre o segundo e o terceiro lugar houve mudanças comparativamente ao *dataset* anterior, tendo o ACO sido o mais lento desta vez, com valores a oscilar entre os 10 e os 13 segundos. O AS-SA

voltou a ser o algoritmo com maiores variações no tempo de execução, com valores compreendidos entre os 5 e os 12 segundos. Podem-se considerar bastante satisfatórios os resultados obtidos para o *Dataset* 4, uma vez que os algoritmos responderam num curto espaço de tempo para o cenário que exigia maior processamento de dados.

Por último, o gráfico presente na Figura 37 compara os tempos de execução para o *Dataset* 5.

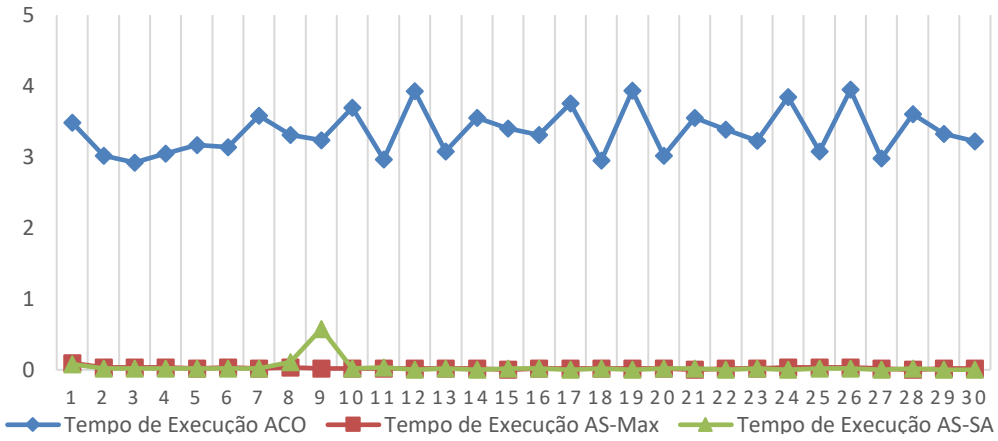


Figura 37 – Tempos de execução dos algoritmos para o *Dataset* 5

Como já foi explicado anteriormente, os *inputs* utilizados para o *Dataset* 5 são muito semelhantes aos do *Dataset* 1, diferindo apenas no meio de transporte utilizado. O meio utilizado influencia os tempos de deslocação entre pontos de interesse, que são superiores aos tempos de deslocação de automóvel. Ao consumir mais tempo nos trajetos entre pontos de interesse, é expectável que o turista visite menos locais do que se se movesse com o auxílio de um automóvel. Por essa razão é também de esperar que os algoritmos atinjam as suas condições de paragem mais cedo para este tipo de deslocações. Pela análise dos valores do gráfico consegue-se constatar esse facto, pelo menos para as execuções do AS-Max e do AS-SA. No caso do AS-Max, todas as 30 execuções foram terminadas em poucas centésimas de segundo. As oscilações que caracterizam o AS-SA não foram tão visíveis para este conjunto de dados, existindo apenas um caso em que ultrapassou o meio segundo de tempo de execução. O ACO foi novamente o mais lento, ao encontrar as suas rotas em tempos próximos dos 3 segundos.

Conclusões

Da análise efetuada para o tempo de execução de cada uma das soluções encontradas conclui-se que o algoritmo com maior velocidade de resposta é o AS-Max, em todos os casos de estudo efetuados. Os restantes algoritmos também obtiveram resultados bastante animadores. Os picos no tempo de execução obtidos pelo algoritmo AS-SA devem-se à natureza aleatória da escolha do nó a visitar. O tempo extra demorado pelo ACO, mesmo para viagens mais curtas, pode ser justificado pela sobrecarga causada pela criação das diversas *threads* e pela quantidade de tarefas executadas nas mesmas. Foi também verificado que os tempos de execução aumentaram conforme se foi acrescentando gradualmente mais pontos de interesse e mais tempo para efetuar a rota, o que já era esperado. No entanto, todos os

testes superaram as expectativas ao serem encontradas rotas válidas num espaço de tempo muito reduzido. Desta forma é suportada a hipótese número 1, ou seja, o algoritmo escolhido é eficiente e permite obter soluções válidas para os diversos parâmetros de entrada num curto espaço de tempo.

Apesar de todos os tempos de execução se encontrarem bastante abaixo do limite máximo estipulado para obter uma rota turística, é de salientar que para estes testes estão a ser considerados apenas o tempo de começo e fim da execução dos algoritmos. Outros fatores como a velocidade da ligação Internet, o tempo comunicação entre os componentes do TheRoute e o tempo para recolha dos dados também influenciam o tempo necessário para apresentar uma recomendação ao turista. Por outro lado, uma grande velocidade de execução por parte do algoritmo garante uma maior margem para suportar algum tipo de demora que possa surgir.

O tempo de execução é, sem dúvida, um fator importante a ter em conta. No entanto, o facto do algoritmo ser mais rápido não garante que a recomendação apresentada seja considerada a melhor para o problema em questão. Uma solução é tão mais adequada aos gostos do turista, quanto mais elevado for o somatório das pontuações dos pontos de interesse a visitar. A secção 5.2.3 apresenta uma análise idêntica à efetuada na presente secção, mas direcionada para os valores da pontuação obtidos nos caminhos gerados.

5.2.3 Cenário Pontuação Obtida

Cada uma das simulações avaliadas na secção anterior obteve uma pontuação relativa ao caminho gerado. Para uma melhor análise desta métrica foram utilizados gráficos de barras, que permitem a comparação dos resultados obtidos pelos algoritmos nas suas primeiras 15 execuções. Desta forma consegue-se ter uma melhor perceção visual dos valores obtidos sem perder muita informação, uma vez que para este parâmetro as restantes execuções revelam valores bastante próximos dos obtidos nas primeiras. Também são apresentadas tabelas com alguns dos dados recolhidos durante as simulações efetuadas, relativos ao número mínimo e máximo de pontos de interesse a visitar, que podem estar relacionados com os valores das pontuações obtidas.

Os resultados obtidos nas simulações para o *Dataset* 1 podem ser consultados na Tabela 6, sendo apresentadas as pontuações obtidas em cada rota gerada no gráfico da Figura 38.

Tabela 6 – Variação do número de pontos de interesse visitados para o *Dataset* 1

Algoritmo	Nº mínimo POI a visitar	Nº máximo POI a visitar
ACO	12	12
AS-Max	11	11
AS-SA	12	13

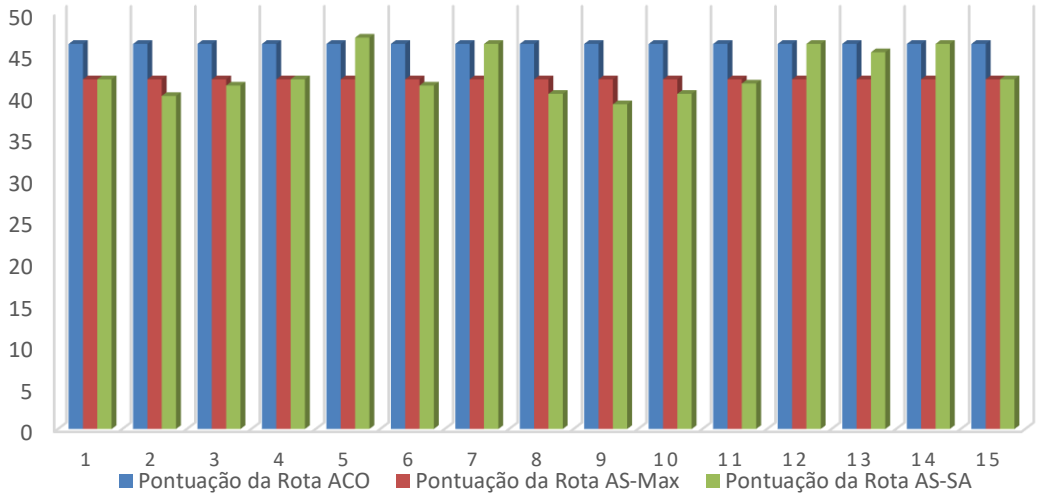


Figura 38 – Pontuações das rotas obtidas pelos algoritmos para o *Dataset* 1

Como é possível verificar pela análise do gráfico de barras, todos os algoritmos apresentam pontuações muito próximas para caminhos relativamente curtos (4 horas). No entanto, consegue-se identificar o ACO como o algoritmo que obtém resultados mais elevados na grande maioria dos casos, atingindo sempre os 46.25 pontos. O AS-Max também apresenta valores constantes na ordem dos 40 pontos. É de salientar que tanto o AS-Max como o ACO obtêm caminhos sempre com o mesmo número de pontos de interesse, 11 e 12, respetivamente. Neste caso, o número de pontos de interesse a visitar numa rota influencia os resultados obtidos, visto que o ACO consegue gerar uma rota com mais um elemento que o AS-Max. No entanto, a diferença da pontuação entre estes dois algoritmos já é significativa (6.5 pontos), o que pode indicar que a seleção efetuada no ACO está mais bem conseguida que no AS-MAX. Relativamente ao AS-SA, verifica-se que os seus resultados variam entre os 39 pontos e os 47 pontos. Desta forma, este algoritmo obtém o valor máximo e o valor mínimo registado nesta série de simulações. Comparativamente aos restantes dois algoritmos testados, o AS-SA ultrapassa os valores obtidos pelo ACO por uns escassos 0.75 pontos e consegue ser inferior ao AS-Max em apenas 1 ponto. É de realçar que o AS-SA varia também no número de pontos de interesse a visitar, entre 12 e 13.

Ao aumentar o tempo da rota para o dobro, obteve-se as variações no número de pontos de interesse a visitar para as rotas geradas demonstrados na Tabela 7. O gráfico de barras ilustrado na Figura 39 revela as pontuações obtidas em cada simulação efetuada para o *Dataset* 2.

Tabela 7 – Variação do número de pontos de interesse visitados para o *Dataset* 2

Algoritmo	Nº mínimo POI a visitar	Nº máximo POI a visitar
ACO	23	23
AS-Max	20	20
AS-SA	23	23

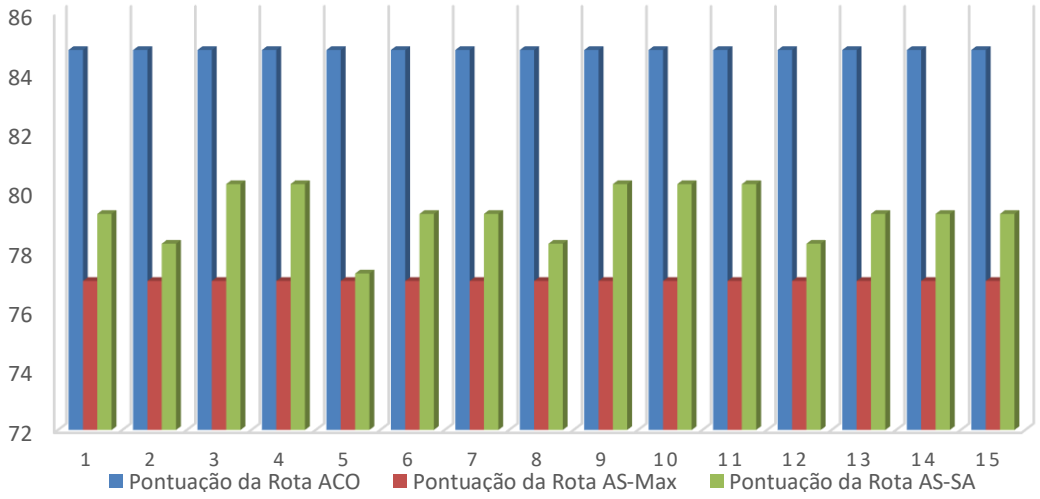


Figura 39 – Pontuações das rotas obtidas pelos algoritmos para o *Dataset* 2

Comparativamente ao gráfico anteriormente analisado, o da Figura 39 demonstra já diferenças significativas entre as pontuações obtidas para cada algoritmo. Neste conjunto de simulações o ACO possui uma enorme vantagem de pontos relativamente aos restantes dois. Obtém o resultado 84.75 em todas as suas execuções, apresentando sempre rotas constituídas por 23 pontos de interesse. Em segundo lugar, mas bastante afastado do ACO, surge o AS-SA com pontuações que variam entre os 77.25 e os 80.25 para rotas constituídas por 23 pontos de interesse. Em último aparece o AS-Max que obtém uns escassos 77 pontos para 20 locais a visitar. O aumento do tempo da rota resultou também numa ampliação das disparidades entre as pontuações obtidas entre os algoritmos de pesquisa, o que serviu para destacar o ACO como o que obtém os melhores resultados.

Para trajetos de 12 horas, tendo em conta os dados recolhidos para o *Dataset* 3, alcançou-se a quantidade de pontos de interesse a visitar presente na Tabela 8 e as pontuações apresentadas no gráfico da Figura 40.

Tabela 8 – Variação do número de pontos de interesse visitados para o *Dataset* 3

Algoritmo	Nº mínimo POI a visitar	Nº máximo POI a visitar
ACO	32	32
AS-Max	30	30
AS-SA	31	31

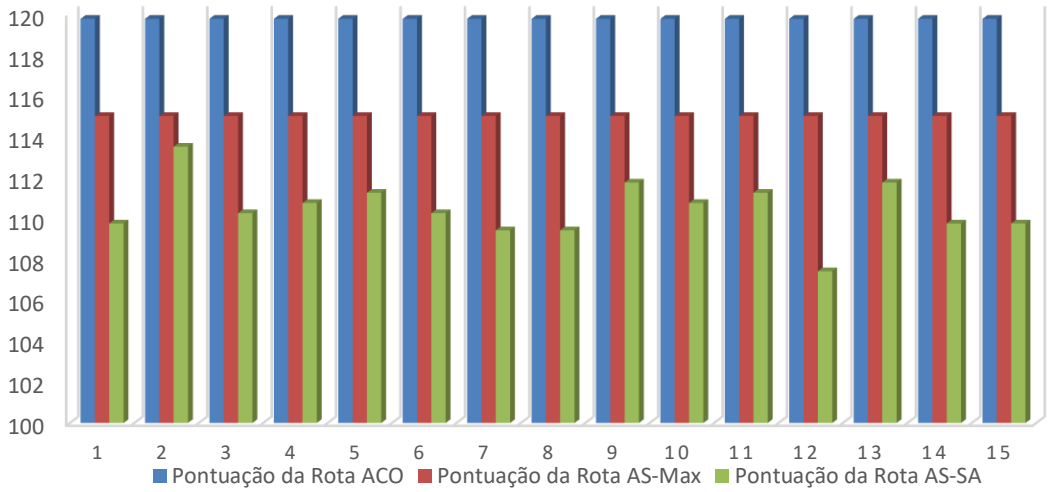


Figura 40 – Pontuações das rotas obtidas pelos algoritmos para o *Dataset* 3

Nesta série de simulações, o algoritmo ACO voltou a revelar-se o que melhor pontua, atingindo um expressivo resultado de 119.75 em todas as suas 15 execuções, com rotas constituídas por 32 pontos de interesse. O segundo lugar desta vez foi conseguido pelo AS-Max com pontuações de 115 para 30 locais a visitar. Por último surge o AS-SA com as suas habituais variações de pontos, atingindo um mínimo de 107.42 e um máximo de 113.5 pontos para 31 pontos a visitar. Para o tempo estipulado, o algoritmo AS-Max voltou a aproximar-se do ACO, diminuindo a diferença de pontos em menos de 5 unidades. Em sentido contrário, o AS-SA afastou-se do líder das pontuações e tornou significativa a sua diferença de resultados, tendo sido considerado o pior da série.

Para o período de tempo mais alargado (24 horas), utilizando os dados de entrada do *Dataset* 4, foram recolhidos os valores referentes ao número de pontos de interesse a visitar apresentados na Tabela 9 e as pontuações dos trajetos, apresentadas na Figura 41.

Tabela 9 – Variação do número de pontos de interesse visitados para o *Dataset* 4

Algoritmo	Nº mínimo POI a visitar	Nº máximo POI a visitar
ACO	54	54
AS-Max	51	51
AS-SA	51	54

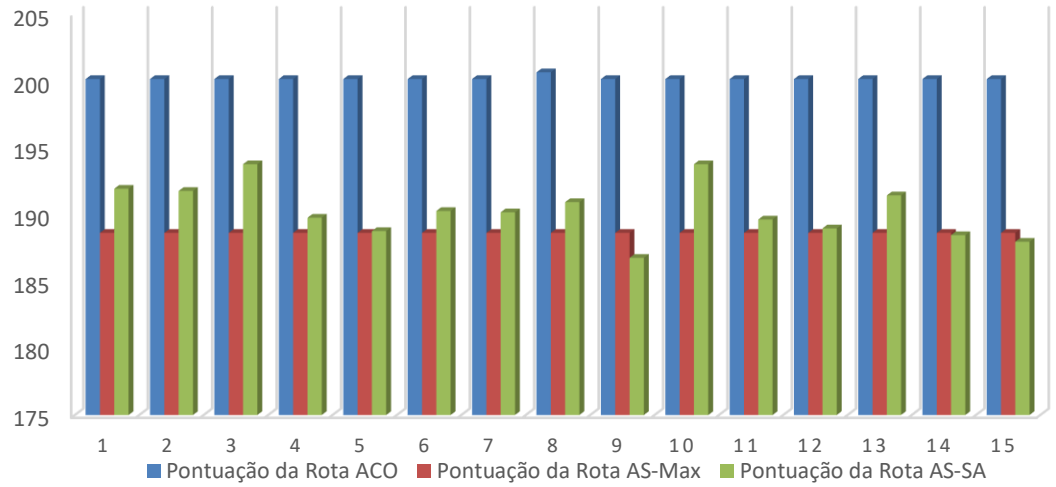


Figura 41 – Pontuações das rotas obtidas pelos algoritmos para o *Dataset* 4

Uma vez mais, o algoritmo ACO revelou ser o mais competente a encontrar caminhos com a maior pontuação ao atingir valores a rondar os 200 pontos, para 54 pontos de interesse a visitar. O algoritmo a ocupar o segundo lugar desta classificação mudou novamente, sendo desta vez o AS-SA que apresenta um valor máximo de 193.75 e um valor mínimo de 186.75 pontos, variando entre os 51 e os 54 pontos de interesse a visitar. Por último surge o AS-Max, bastante afastado dos restantes dois algoritmos, com uns constantes 188.62 pontos para 51 locais. A quantidade de pontos que constituem as rotas é muito elevado, uma vez que são considerados na solução final locais com tempos de visita muito curtos e próximos geograficamente.

O último caso testado para esta métrica é a comparação da pontuação obtida pelos algoritmos que têm como dados de entrada os provenientes do *Dataset* 5. Tal como para a métrica anteriormente avaliada, este teste tem especial interesse na constatação das diferenças entre as deslocações de automóvel e as deslocações a pé. Como tal, na Tabela 10 são apresentados os números de pontos de interesse que constituem as rotas geradas e na Figura 42 as respetivas pontuações obtidas.

Tabela 10 – Variação do número de pontos de interesse visitados para o *Dataset* 5

Algoritmo	Nº mínimo POI a visitar	Nº máximo POI a visitar
ACO	9	9
AS-Max	8	8
AS-SA	9	10

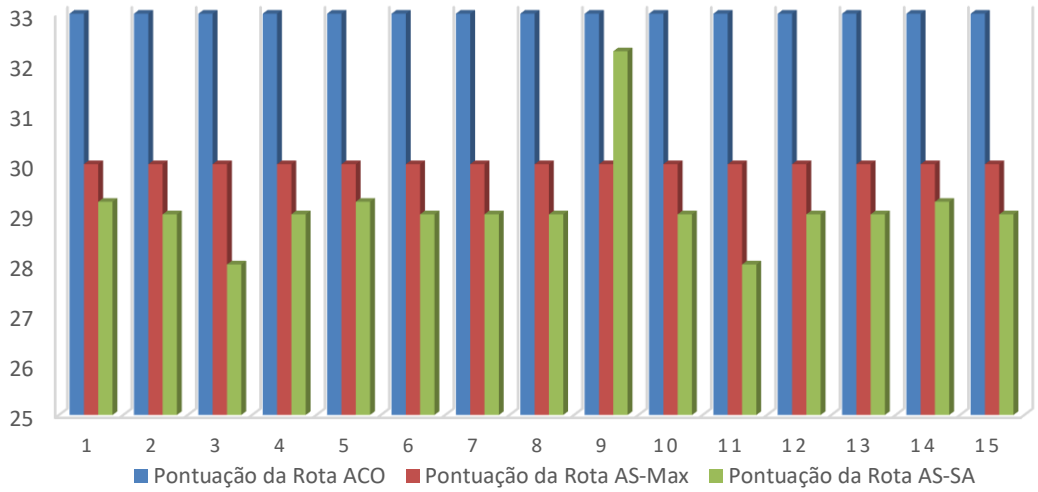


Figura 42 – Pontuações das rotas obtidas pelos algoritmos para o *Dataset* 5

Relativamente aos resultados obtidos com os dados do *Dataset* 1, que possui períodos de viagem homólogos aos do *Dataset* 5, é verificado que não existe grande diferença relativamente ao posicionamento dos algoritmos na classificação das pontuações obtidas. O ACO continua a ser o mais fiável para esta métrica, obtendo 33 pontos em todas as suas execuções, para um total de 9 pontos de interesse. O AS-Max atinge valores sempre na casa dos 30 pontos com rotas de 8 locais. O AS-SA alcança maioritariamente os piores resultados, atingindo um mínimo de 28 e um máximo de 32.5 pontos. O número de pontos de interesse que constituem as suas rotas variam entre 9 e 10 elementos. Tendo em conta os valores verificados, consegue-se comprovar que para um mesmo conjunto de dados o meio de transporte tem uma influência significativa na quantidade de pontos de interesse visitados e na pontuação obtida.

Conclusões

A análise efetuada para as pontuações obtidas nas rotas encontradas pelos diferentes algoritmos permite concluir que o ACO é o mais adequado para a métrica em estudo, sendo o que suporta mais convictamente a hipótese número 2. Desta forma, é possível a recomendação de um trajeto que agrade ao turista, tendo em conta vários aspetos pessoais do seu perfil e condicionantes do contexto envolvente. O AS-SA tornou a revelar-se bastante inconstante nos resultados obtidos, ao obter diferenças de valores consideráveis em condições de teste idênticas. A heurística baseada no *Simulated Annealing* tem, mais uma vez, uma grande influência para a aleatoriedade dos resultados. Para o caso do AS-Max, verifica-se que os cortes efetuados na pesquisa da rota prejudicam a pontuação da solução obtida, sendo este algoritmo o que obteve a pior pontuação na maioria dos testes efetuados.

Tendo sido concluída a análise para os tempos de execução e para as pontuações obtidas nas rotas geradas, resta analisar a última métrica definida. A secção 5.2.4 apresenta uma análise idêntica às efetuadas até ao momento, mas direcionada para as percentagens obtidas relativamente ao aproveitamento do tempo definido para a realização da rota.

5.2.4 Cenário Aproveitamento do Tempo Definido para a Rota

Antes de concretizar o pedido para a criação de uma rota, o turista define tempos para o início e o fim do trajeto a realizar. Os algoritmos desenvolvidos têm em consideração esses valores na pesquisa e construção da solução que é apresentada ao utilizador. No entanto, como as rotas estão a ser geradas com o objetivo de obter uma solução que se aproxime dos gostos do turista, nem sempre o tempo pretendido para a concretização do trajeto é preenchido na totalidade. Apesar de ser importante que o caminho seja gerado num curto espaço de tempo e que a sua pontuação seja elevada, o algoritmo não deve apresentar uma solução que não aproveite bem os requisitos temporais definidos. Por essa razão, torna-se também importante avaliar a taxa de aproveitamento para as soluções geradas.

Em cada um dos casos de estudo, são apresentados os horários mínimos e máximos obtidos para o tempo de término da rota nas 30 execuções efetuadas, assim como a média da taxa percentual de aproveitamento do tempo definido para o trajeto. A Tabela 11 apresenta os horários da rota obtidos e a taxa de aproveitamento do tempo definido para a concretização de uma rota de 4 horas, calculada tendo em conta os resultados obtidos para o *Dataset* 1.

Tabela 11 – Horários da rota e média da taxa de aproveitamento para o *Dataset* 1

Algoritmo	Horário Início	Horário Mínimo Fim	Horário Máximo Fim	Taxa Média Aproveitamento
ACO	14:00:00	17:38:42	17:38:44	91.13%
AS-Max	14:00:00	17:58:41	17:58:41	99.45%
AS-SA	14:00:00	17:49:33	17:59:57	98.93%

Como é possível verificar pela análise das taxas de aproveitamento, todos os algoritmos atingem valores acima dos 91%. O AS-Max obtém a melhor média ao gerar rotas que ocupam 99.45% do tempo definido pelo utilizador. Por sua vez, o AS-SA é o que consegue atingir a segunda melhor média com valores a rondar os 98.93%. Em último surge o ACO com 91.13% de aproveitamento, bastante aquém dos restantes algoritmos. Enquanto para os algoritmos AS-Max e AS-SA se contabilizam desperdícios de poucos minutos ou segundos, em alguns casos, o ACO não consegue aproveitar aproximadamente 20 minutos numa rota de 4 horas. Resta apurar a evolução desta tendência para rotas com horários mais alargados de visita.

A Tabela 12 contém os horários e aproveitamentos referentes às rotas geradas para um período de 8 horas, tendo em conta os dados recolhidos no *Dataset* 2.

Tabela 12 – Horários da rota e média da taxa de aproveitamento para o *Dataset* 2

Algoritmo	Horário Início	Horário Mínimo Fim	Horário Máximo Fim	Taxa Média Aproveitamento
ACO	12:30:00	20:04:45	20:04:45	94.74%
AS-Max	12:30:00	20:29:10	20:29:10	99.83%
AS-SA	12:30:00	20:22:55	20:29:59	99.56%

Com o aumento do período definido para concluir a rota é verificado um maior aproveitamento do tempo. O ACO continua a ser o algoritmo com o pior rendimento nesta métrica, atingindo o valor médio de 94.74%, o que corresponde a um desperdício de aproximadamente 25 minutos numa rota de 8 horas. No entanto, comparativamente à análise anterior, o aumento do tempo não aproveitado foi de apenas 5 minutos, o que não é muito significativo tendo em conta que o tempo da rota duplicou. Os restantes algoritmos também obtiveram uma ligeira melhoria na sua percentagem de aproveitamento. O AS-Max voltou a obter a melhor classificação com 99.83%, pouco à frente do registado pelo AS-SA com 99.56% do tempo aproveitado.

Os dados apresentados na Tabela 13 são referentes aos horários e ao seu correspondente aproveitamento para rotas programadas para 12 horas, de acordo com os dados recolhidos no *Dataset* 3.

Tabela 13 – Horários da rota e média da taxa de aproveitamento para o *Dataset* 3

Algoritmo	Horário Início	Horário Mínimo Fim	Horário Máximo Fim	Taxa Média Aproveitamento
ACO	09:00:00	20:32:10	20:32:10	96.13%
AS-Max	09:00:00	20:59:37	20:59:37	99.95%
AS-SA	09:00:00	20:39:16	20:59:57	99.59%

Uma vez mais, com o aumento do período dedicado para a realização da rota, verificam-se valores mais elevados para as médias de aproveitamento do tempo. Os algoritmos AS-Max e AS-SA são os que possuem uma maior taxa de aproveitamento para esta métrica, fixando-se em valores perto dos 100%. O ACO continua a ampliar a quantidade de minutos desperdiçados, chegando a atingir valores próximos dos 28 minutos.

Para o período de viagem de 24 horas, com os dados recolhidos no *Dataset* 4, obtém-se os horários e os valores relativos à taxa de aproveitamento apresentados na Tabela 14.

Tabela 14 – Horários da rota e média da taxa de aproveitamento para o *Dataset* 4

Algoritmo	Horário Início	Horário Mínimo Fim	Horário Máximo Fim	Taxa Média Aproveitamento
ACO	00:00:00	22:53:58	23:16:18	95.52%
AS-Max	00:00:00	23:57:49	23:57:49	99.85%
AS-SA	00:00:00	23:30:03	23:58:49	99.49%

Apesar de o valor das taxas médias ser inferior para este *dataset* relativamente ao anterior, verifica-se que o tempo definido para a viagem é praticamente preenchido na totalidade pelo AS-Max e pelo AS-SA, atingindo valores muito próximos dos 100%. O ACO voltou a ser o pior cotado nesta métrica ao ter uma média de aproveitamento 95.52%. Ao obter o seu melhor valor em apenas uma das simulações, em que desperdiçou aproximadamente 43 minutos do tempo previsto, verificou-se que esse desperdício foi superior a 1 hora em todas as restantes.

Por último, a Tabela 15 compara os horários e as taxas de aproveitamento dos algoritmos para o *Dataset* 5.

Tabela 15 – Horários da rota e média da taxa de aproveitamento para o *Dataset* 5

Algoritmo	Horário Início	Horário Mínimo Fim	Horário Máximo Fim	Taxa Média Aproveitamento
ACO	14:00:00	17:36:41	17:36:41	90.28%
AS-Max	14:00:00	17:59:34	17:59:34	99.82%
AS-SA	14:00:00	17:53:24	17:59:55	99.16%

Os resultados obtidos para o *Dataset* 5 continuam a suportar os que foram obtidos para os restantes analisados até ao momento. Apesar de neste caso as rotas a efetuar serem calculadas para um meio de transporte diferente, existe coerência nos valores obtidos nas suas taxas de aproveitamento. Os valores obtidos assemelham-se bastante aos do *Dataset* 1, que é o mais idêntico ao nível das condições de teste. O algoritmo AS-Max é, pela quinta vez, o que aproveita melhor o tempo definido pelo turista para completar a rota, ao conseguir uma média de valores igual a 99.82%. Pouco atrás surge o AS-SA com 99.16% e em último o ACO com 90.28%. Para o ACO verifica-se um desperdício do tempo estipulado na ordem dos 23 minutos, mais 3 do que no *Dataset* 1.

Conclusões

A partir da análise efetuada para o aproveitamento do tempo da rota definida pelo turista, conclui-se que o algoritmo AS-Max é o que melhor satisfaz esta métrica em todos os casos de estudo efetuados. O AS-SA também obteve resultados bastante animadores, apesar da inconstância das suas variações. O único algoritmo que ficou aquém das expectativas foi o ACO, tendo sido comprovados desperdícios de tempo bem superiores aos obtidos pelos restantes. Foi também verificada uma tendência para a subida da taxa de aproveitamento com o aumento gradual do tempo de viagem estipulado. Ao apresentar taxas sempre acima dos 99% o AS-Max suporta categoricamente a hipótese número 3, preenchendo quase na totalidade o tempo indicado pelo turista.

6 Conclusões

Este capítulo visa apresentar as conclusões sobre o trabalho desenvolvido no decorrer desta dissertação. A secção 6.1 incide principalmente sobre a realização dos objetivos propostos e sobre algumas particularidades da solução desenvolvida, sendo o impacto científico desta dissertação apresentado na secção 6.2. De seguida, a secção 6.3 expõe algumas das limitações da solução atual, assim como algumas ideias a desenvolver no futuro. Por fim, a secção 6.4 apresenta as considerações finais do autor desta dissertação relativamente a tudo o que foi desenvolvido e aos benefícios conseguidos pela sua conclusão.

6.1 Síntese e Conclusões do Trabalho

O trabalho desenvolvido nesta dissertação surgiu a partir da necessidade de integração de um módulo de geração de rotas no projeto TheRoute, um sistema de recomendação turística existente. Um dos objetivos principais deste sistema é apresentar aos seus utilizadores rotas personalizadas, tendo em conta as suas preferências e a análise de aspetos do contexto envolvente. Para conseguir desempenhar esse papel da melhor forma, são traçados perfis de utilizador através de questionários efetuados sobre a sua personalidade. Através dos resultados obtidos nessa avaliação, é possível inferir classificações para cada uma das categorias turísticas existentes no sistema TheRoute. O trabalho efetuado consiste em gerar os trajetos e os respetivos horários de visita, tendo em conta o perfil dos utilizadores que solicitaram a recomendação.

Numa primeira fase foram analisadas algumas das soluções existentes no âmbito dos sistemas de recomendação turística. Apesar de se ter verificado a existência de alguns projetos semelhantes na área do turismo, nenhum dos sistemas se propõe a gerar recomendações dinâmicas tendo em conta o conjunto das características individuais do utilizador e o contexto envolvente. A grande maioria, mesmo os que têm em consideração as preferências do utilizador, apresenta uma série de trajetos pré-calculados para um determinado perfil. Por estas razões, tornou-se ainda mais desafiante contribuir para um sistema que pode oferecer algo único.

A análise continuou através da investigação de algoritmos capazes de gerar trajetos ao examinar um determinado espaço de pesquisa. Para resolver estes problemas de otimização são normalmente utilizados algoritmos baseados em heurísticas. A utilização de heurísticas é fundamental para determinar a qualidade das soluções obtidas para o problema em questão. Estes algoritmos utilizam recursos computacionais consideráveis, mas são capazes de produzir soluções aceitáveis. Encontrou-se uma enorme variedade de algoritmos baseados em heurísticas capazes de gerar caminhos entre diversos pontos. Devido ao processo de criação de rotas ser demorado, principalmente para grafos com muitos nós e ligações, a sua implementação e experimentação deve ser bem planeada e ter em consideração várias preocupações sobre o seu desempenho.

Na fase da implementação, conseguiu-se desenvolver três soluções viáveis para a geração de rotas. Duas utilizam o algoritmo A* para encontrar o caminho a recomendar ao turista, outra utiliza o *Ant Colony Optimization*. Todas as soluções foram testadas tendo em conta as métricas definidas e a experimentação de possíveis cenários reais. A melhor classificação para o tempo de execução foi obtida pelo algoritmo denominado AS-Max, uma vez que encontrou mais rapidamente uma solução. Outra métrica importante considerada foi a pontuação obtida para a rota gerada, que se encontra diretamente relacionada com as preferências do turista ou do grupo de turistas. Neste caso específico o ACO foi o algoritmo que se destacou, ao conseguir atingir as melhores soluções em todos os testes efetuados. A última métrica analisada consistiu na medição da taxa de aproveitamento do tempo indicado pelo turista para a concretização do trajeto. Verificou-se que o AS-Max é o que consegue preencher mais esse tempo, ao ganhar com uma pequena margem para o AS-SA. Os resultados obtidos foram de encontro às hipóteses definidas, uma vez que se conseguiu obter recomendações de rotas turísticas em curtos espaços de tempo, tendo em consideração aspetos da análise do contexto e o perfil do utilizador. Por se ter destacado em duas das três métricas avaliadas, o algoritmo eleito como o mais adequado para o módulo de geração de rotas foi o AS-Max. Apesar de não ser o que alcança as melhores pontuações para os caminhos gerados, é o que possui uma melhor relação qualidade-custo para as soluções geradas. Além disso, consegue aproveitar praticamente a totalidade do tempo definido pelo turista para a realização da rota.

Por fim, o módulo de geração de rotas foi integrado como um microsserviço no sistema TheRoute.

6.2 Impacto Científico

O trabalho desenvolvido nesta dissertação deu origem a um artigo científico que será submetido para apresentação na 18ª Conferência Internacional sobre Aplicações Práticas de Agentes e Sistemas Multiagentes (PAAMS). A conferência está agendada entre os dias 17 e 19 de junho de 2020 na cidade italiana de Áquila.

6.3 Limitações e Trabalho Futuro

De todas as soluções desenvolvidas o ACO é, sem dúvida, o mais sofisticado e o que possui um maior potencial de evolução. Como se verificou nos testes efetuados, este algoritmo é o que consegue obter trajetos com pontuações mais elevadas. Apesar do algoritmo escolhido ser o melhor em duas das três dimensões identificadas para a satisfação do utilizador, não conseguiu superar o ACO em nenhuma das experiências realizadas para esta métrica. Por essa razão, considera-se trabalho futuro tentar melhorar o desempenho do ACO nas vertentes do tempo de execução e do aproveitamento do tempo definido para a conclusão da rota. Ao obter melhorias nestas duas métricas, principalmente na do aproveitamento da rota, conseguia-se alcançar uma solução mais completa e robusta do que a implementada para o AS-Max.

Outra melhoria que pode ser efetuada a nível algorítmico é a substituição da heurística utilizada pelo A*. Foram utilizadas duas heurísticas, uma seleciona o nó pelo valor máximo da pontuação e a outra efetua essa seleção tendo em conta a aleatoriedade da heurística do *Simulated Annealing*. Da forma como este algoritmo foi implementado, facilmente se substitui a heurística utilizada por uma nova. Resta fazer uma investigação para encontrar uma função que consiga maximizar ainda mais a pontuação do trajeto gerado. Os cortes efetuados nesta pesquisa são indispensáveis, uma vez que facilmente ocorrem problemas relacionados com o uso excessivo de memória se o espaço temporal não for limitado. Tentar melhorar a forma como esses cortes são feitos também é uma possível melhoria, visto que estão a ser descartadas opções que podem ser mais vantajosas do que o resultado obtido.

Ao nível das funcionalidades suportadas pelo TheRoute, era interessante a aplicação permitir ao utilizador definir horários de pausa para descanso ou refeição. Até ao momento, este aspeto não é considerado e o trajeto criado pelo módulo de geração de rotas não possui qualquer interrupção. Por uma questão prática e de usabilidade, o turista deveria poder definir estes períodos quando solicita uma nova rota, de forma a permitir a recomendação de um horário de visita mais próximo às suas necessidades.

Outro aspeto importante a considerar é a inclusão na aplicação móvel de perguntas acerca da mobilidade e necessidades especiais que possam condicionar o utilizador. Apesar do atual modelo de dados do TheRoute já incluir este conceito, não é ainda possível saber se o turista possui algum tipo de deficiência ou limitação que o impeça de visitar certos pontos de interesse. Os pontos de interesse também devem incluir informação sobre a sua acessibilidade, de modo a serem selecionados apenas os que o utilizador com necessidades especiais pode visitar.

6.4 Apreciação Final

Após completar todas as etapas descritas no presente documento, considera-se que os objetivos definidos para esta dissertação foram alcançados na sua plenitude.

A elaboração do módulo de geração de rotas contribuiu para o enriquecimento da plataforma TheRoute, uma vez que lhe permitiu adquirir a sua principal funcionalidade para a criação de recomendações turísticas. As arquiteturas de *software* exploradas, bem como a pesquisa e as experiências efetuadas ampliaram o conhecimento em áreas pouco exploradas pelo autor desta dissertação, pelo que os contributos do trabalho realizado são considerados enriquecedores e serão certamente valiosos para o desempenho de trabalhos futuros.

Referências

Abdmouleh, Z., Gastli, A., Ben-Brahim, L., Haouari, M., & Al-Emadi, N. (2017). Review of optimization techniques applied for the integration of distributed generation from renewable energy sources. *Renewable Energy*, *113*. https://doi.org/10.1016/j.renene.2017.05.087

Adomavicius, G., & Tuzhilin, A. (2009). Context-Aware Recommender Systems. *In Technology-Enhanced Learning Depend on Context", the 1st Workshop on Context-Aware Rss for Learning*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.423.4220&rep=rep1&type=pdf

Alhanjouri, M., & Alfarra, B. (2011). Ant Colony versus Genetic Algorithm based on Travelling Salesman Problem. *International Journal of Computer Technology and Applications*, *2*, 570–578.

Borràs, J., Moreno, A., & Valls, A. (2014). Intelligent tourism recommender systems: A survey. *Expert Systems with Applications*, *41*(16), 7370–7389. https://doi.org/10.1016/j.eswa.2014.06.007

Champiri, Z. D., Shahamiri, S. R., & Salim, S. S. B. (2015). A systematic review of scholar context-aware recommender systems. *Expert Systems with Applications*, *42*(3), 1743–1758. https://doi.org/10.1016/j.eswa.2014.09.017

Costa, A. L., Cunha, M. C., Coelho, P. A. L. F., & Einstein, H. H. (2016). Application of the Simulated Annealing Algorithm for Transport Infrastructure Planning. In X.-S. Yang, G. Bekdaş, & S. M. Nigdeli (Eds.), *Metaheuristics and Optimization in Civil Engineering* (pp. 235–253). https://doi.org/10.1007/978-3-319-26245-1_11

Dorigo, M., Birattari, M., & Stützle, T. (2006). Ant Colony Optimization. *IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE*. Retrieved from https://courses.cs.ut.ee/all/MTAT.03.238/2011K/uploads/Main/04129846.pdf

Durillo, J. J., & Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, *42*(10), 760–771. https://doi.org/10.1016/j.advengsoft.2011.05.014

Eiselt, H. A., & Sandblom, C.-L. (2000). Heuristic Algorithms. In H. A. Eiselt & C.-L. Sandblom (Eds.), *Integer Programming and Network Models* (pp. 229–258). https://doi.org/10.1007/978-3-662-04197-0_11

Esmin, A. A. A., Coelho, R. A., & Matwin, S. (2015). A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data. *Artificial Intelligence Review*, *44*(1), 23–45. https://doi.org/10.1007/s10462-013-9400-4

Garcia, A., Arbelaitz, O., Linaza, M. T., Vansteenwegen, P., & Souffriau, W. (2010). Personalized Tourist Route Generation. In F. Daniel & F. M. Facca (Eds.), *Current Trends in Web Engineering* (pp. 486–497). Springer Berlin Heidelberg.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, *13*(5), 533–549. https://doi.org/10.1016/0305-0548(86)90048-1

Glover, F., Martí, R., & Laguna, M. (2007). Principles of Tabu Search. In T. Gonzalez (Ed.), *Handbook of Approximation Algorithms and Metaheuristics* (Vol. 20073547, pp. 23-1-23–12). https://doi.org/10.1201/9781420010749.ch23

Gupta, L. (2015). Thread pool. Retrieved September 25, 2019, from https://howtodoinjava.com/wp-content/uploads/2015/03/Thread_pool.png

Hong Kong, U. (n.d.). Traveling salesman problem (TSP). Retrieved September 25, 2019, from https://i2.cs.hku.hk/~hkual/files/graphs/Traveling%20salesman%20problem%20(TSP)%201.png

Jungblut, T. (2015). *Ant Colony Optimization for TSP problems* [Java]. Retrieved from https://github.com/thomasjungblut/antcolonyopt

Kashyap, H. (2017). *Heuristic Search-A star algorithm*. Retrieved from https://www.slideshare.net/hemak15/lecture-14-heuristic-searcha-star-algorithm

Kennedy, J. (2006). *Handbook of nature-inspired and innovative computing*. Retrieved from https://link.springer.com/content/pdf/10.1007/0-387-27705-6_6.pdf

Koen, P. A., Ajamian, G. M., Boyce, S., Clamen, A., Fisher, E., Fountoulakis, S., … Seibert, R. (2002). Fuzzy Front End: Effective Methods, Tools, and Techniques. *The PDMA ToolBook for New Product Development*, 32.

Koen, P. A., Bertels, H. M. J., & Kleinschmidt, E. (2014). *Managing the Front End of Innovation - Part 1*.

Kokash, N. (2005). *An introduction to heuristic algorithms*. 9.

Kurata, Y., & Hara, T. (2013). CT-Planner4: Toward a More User-Friendly Interactive Day-Tour Planner. *ResearchGate*. https://doi.org/10.1007/978-3-319-03973-2_6

Marianne. (2014, July 4). The haversine. Retrieved October 5, 2019, from plus.maths.org website: https://plus.maths.org/content/lost-lovely-haversine

Martin, R. C., & Martin, M. (2007). *Agile principles, patterns, and practices in C#*. Upper Saddle River, NJ: Prentice Hall.

Mehlhorn, K., & Sanders, P. (2008). ShortestPaths.pdf. Retrieved March 5, 2019, from http://people.mpi-inf.mpg.de/~mehlhorn/ftp/Toolbox/ShortestPaths.pdf

Mitchell, M. (1996). *An Introduction To Genetic Algorithms*. The MIT Press.

Monczka, R. M., Handfield, R. B., & Giunipero, L. (2008). *Purchasing and Supply Chain Management* (4th ed.). Cengage Learning.

Morrison, D. R., Jacobson, S. H., Sauppe, J. J., & Sewell, E. C. (2016). Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, *19*, 79–102. https://doi.org/10.1016/j.disopt.2016.01.005

Mukhairez, H., & Maghari, A. (2015). Performance Comparison of Simulated Annealing, GA and ACO Applied to TSP. *International Journal of Intelligent Computing Research (IJICR)*, *Volume 6*, 647–654. https://doi.org/10.20533/ijicr.2042.4655.2015.0080

Nicola, S., Ferreira, E. P., & Ferreira, J. J. P. (2012). A novel framework for modeling value for the customer, an essay on negotiation. *International Journal of Information Technology & Decision Making*, *11*(03), 661–703. https://doi.org/10.1142/S0219622012500162

Portal do INE. (2018, December 17). Retrieved January 27, 2019, from Conta Satélite do Turismo para Portugal website: https://www.ine.pt/xportal/xmain?xpid=INE&xpgid=ine_destaques&DESTAQUESdest_boui=340406560&DESTAQUESmodo=2

Portal do INE. (2019, August 2). Retrieved October 5, 2019, from Alojamento turístico com crescimento mas em desaceleração - 2018 website: https://www.ine.pt/xportal/xmain?xpid=INE&xpgid=ine_destaques&DESTAQUESdest_boui=354232444&DESTAQUEStema=55581&DESTAQUESmodo=2&xlang=pt

Porumbel, D. C. (2012). Heuristic algorithms and learning techniques: applications to the graph coloring problem. *4OR*, *10*(4), 393–394. https://doi.org/10.1007/s10288-011-0193-5

PÚBLICO. (2018, April 4). Lisboa e Porto têm mais turistas por residente do que Londres e Barcelona. Retrieved January 27, 2019, from PÚBLICO website: https://www.publico.pt/2018/04/04/sociedade/noticia/lisboa-e-porto-tem-mais-turistas-por-residente-que-londres-e-barcelona-1809036

Raamesh, L. (2013). Integrated multi objective test case prioritization system. *University*. Retrieved from http://shodhganga.inflibnet.ac.in:8080/jspui/handle/10603/24118

Ramos, C., Marreiros, G., Martins, C., Faria, L., Conceição, L., Santos, J., … Lima, L. S. (2019). A Context-Awareness Approach to Tourism and Heritage Routes Generation. In P. Novais, J. J. Jung, G. Villarrubia González, A. Fernández-Caballero, E. Navarro, P. González, … D. Durães (Eds.), *Ambient Intelligence – Software and Applications –, 9th International Symposium on Ambient Intelligence* (pp. 10–23). Springer International Publishing.

Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender Systems: Introduction and Challenges. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender Systems Handbook* (pp. 1–34). https://doi.org/10.1007/978-1-4899-7637-6_1

Rodriguez-Mier, P., Gonzalez-Sieira, A., Mucientes, M., Lama, M., & Bugarin, A. (2014). Hipster: An open source Java library for heuristic search. *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*, 1–6. https://doi.org/10.1109/CISTI.2014.6876914

Russell, S. J., & Norvig, P. (2009). *Artificial Intelligence: A Modern Approach* (3rd ed.). Upper Saddle River, New Jersey: Prentice Hall.

Sun, Y., & Lee, L. (2004). *Agent-Based Personalized Tourist Route Advice System*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.643.8693&rep=rep1&type=pdf

van Laarhoven, P. J. M., & Aarts, E. H. L. (1987). *Simulated Annealing: Theory and Applications* (1st ed.). Springer, Dordrecht.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., & Oudheusden, D. V. (2011). The City Trip Planner: An expert system for tourists. *Expert Systems with Applications*, *38*(6), 6540–6546. https://doi.org/10.1016/j.eswa.2010.11.085

Visitacity. (n.d.). Visit A City: Create Your Personal Travel Guide. Retrieved September 25, 2019, from https://www.visitacity.com/

Woodall, T. (2003). *Conceptualising "Value for the Customer": An Attributional, Structural and Dispositional Analysis*. 45.

Yang, W.-S., & Hwang, S.-Y. (2013). iTravel: A recommender system in mobile peer-to-peer environment. *Journal of Systems and Software*, *86*(1), 12–20. https://doi.org/10.1016/j.jss.2012.06.041

Yao, J., Lin, C., Xie, X., Wang, A., & Hung, C.-C. (2010, January 1). *Path Planning for Virtual Human Motion Using Improved A* Star Algorithm*. 1154–1158. https://doi.org/10.1109/ITNG.2010.53

Yousefikhoshbakht, M., Didehvar, F., & Rahmati, F. (2013). *Modification of the Ant Colony Optimization for Solving the Multiple Traveling Salesman Problem*. 17.

Zeithaml, V. A. (1988). Consumer Perceptions of Price, Quality, and Value: A Means-End Model and Synthesis of Evidence. *Journal of Marketing*, *52*(3), 2. https://doi.org/10.2307/1251446

Zeng, W., & L. Church, R. (2009). Finding shortest paths on real road networks: The case for A. *International Journal of Geographical Information Science*, *23*, 531–543. https://doi.org/10.1080/13658810801949850

Zopiatis, A., & Constanti, P. (2012). Extraversion, openness and conscientiousness: The route to transformational leadership in the hotel industry. *Leadership & Organization Development Journal*, *33*, 86–104. https://doi.org/10.1108/01437731211193133