Aplicação do Problema do Caixeiro Viajante para determinação de rotas turísticas

Ana Paula Picolo[1], Roger Sá da Silva[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS), Campus Veranópolis – RS – Brasil

anapaulapicolo@hotmail.com, roger.silva@veranopolis.ifrs.edu.br

Resumo. *O Problema do Caixeiro Viajante (PCV) é um problema clássico de otimização combinatória que é intensamente investigado em matemática computacional devido sua vasta área de aplicação e a complexidade de obtenção de uma solução ideal. Uma área de estudo que possui uma relação com pontos de interesse que podem ser associados utilizando o PCV são os pontos turísticos percorridos pelos turistas em suas viagens. Este trabalho se propôs ao desenvolvimento de uma solução computacional para o cálculo de rotas turísticas usando o PCV buscando a determinação de um método de solução do PCV adequado para a utilização em uma aplicação móvel e a geração de um roteiro com pontos turísticos de interesse do usuário para as rotas turísticas através da aplicação. Com o uso da heurística do Vizinho Mais Próximo foi possível validar a proposta em um protótipo de aplicativo móvel, constatando-se a viabilidade de utilização desta aplicação.*

Abstract. *The traveling salesman problem (TSP) is a classic combinatorial optimization problem that is intensively investigated in computational mathematics due to its wide area of application and the complexity of obtaining an ideal solution. An area of study that has a relationship with points of interest that can be associated using TSP are the tourist spots visited by tourists on their trips. This paper proposed the development of a computational solution for the calculation of tourist routes using PCV, seeking to determine a suitable PCV solution method to use in a mobile application and the generation of a route with the user tourist points of interest to the tourist routes of the application. Using the Nearest Neighbor heuristic, it was possible to validate the proposal in a mobile application prototype, verifying the feasibility of using this application.*

1. Introdução

O problema do caixeiro viajante (PCV), do inglês *Travelling Salesman Problem* (TSP), é um problema clássico de otimização combinatória. A sua origem está relacionada a um caixeiro viajante que pretende visitar um conjunto de cidades uma única vez, retornando à cidade inicial ao final do percurso, e realizando esse trajeto com a menor distância possível [Oliveira 2015].

O problema, que parece modesto, é um dos mais intensamente investigados em matemática computacional e pode ser aplicado em diversas áreas como logística, genética, manufatura, telecomunicações e neurociência [Applegate et. al 2007]. Considerando a utilização tradicional deste problema, é possível vislumbrar uma área de estudo que possui relação com pontos de interesse que podem ser associados: pontos turísticos percorridos por turistas em suas viagens.

O turismo tem uma importância muito grande no desenvolvimento socioeconômico, podendo atingir diretamente 52 atividades econômicas diferentes.

Além disso, em muitas regiões é a única atividade econômica que permite associar a geração de emprego e renda com a conservação do meio natural. Ainda, o contato dos visitantes com a população de uma localidade favorece o desenvolvimento cultural de ambos [Ignarra 2013]. Portanto, entende-se por ser relevante a implementação de uma ferramenta que possa relacionar os pontos nos quais os turistas pretendem realizar seu percurso de uma forma favorável. Neste trabalho são utilizados os pontos turísticos da cidade de Veranópolis, no Rio Grande do Sul.

Diante desse contexto, o trabalho tem como objetivo o desenvolvimento de uma solução computacional que permita o cálculo de rotas turísticas usando o problema do caixeiro viajante, buscando a sua validação por meio de um protótipo de aplicativo móvel. De maneira mais específica, busca-se a determinação de um método de solução do PCV adequado para a utilização na aplicação móvel e a geração de um roteiro com pontos turísticos de interesse do usuário através da aplicação.

Neste trabalho, inicialmente, na Seção 2 encontra-se a revisão da literatura que apresenta a pesquisa sobre os métodos de solução do PCV e a sistematização de trabalhos relacionados; na Seção 3 está a metodologia utilizada no desenvolvimento do trabalho; na Seção 4 realiza-se a apresentação e discussão dos resultados; na Seção 5 estão as considerações finais; e, por fim, na Seção 6 elenca-se as referências bibliográficas.

2. Revisão da Literatura

Existem diversos problemas computacionais relacionados à otimização combinatória que envolvem a determinação de propriedades dos grafos [Karp 1972]. O PCV é um dos problemas que fazem parte deste meio.

Um grafo (G) é um objeto matemático formado por dois conjuntos, um deles é chamado de vértice (V) e o outro é um conjunto de relações entre os vértices, chamado de conjunto de arestas (E). Se dois vértices v e w de V estiverem relacionados, diz-se que entre eles existe uma aresta pertencente a E, chamada (v,w) ou simplesmente vw. Portanto, para se conhecer um grafo é necessário saber o que são os vértices e como estão ligados entre si. Com isso pode-se denominar um dado grafo como sendo $G=(V,E)$ [Boaventura Netto e Jurkiewicz 2009].

Os grafos podem ser classificados como simétricos e assimétricos. Será simétrico se o custo Cv,w e Cw,v forem iguais e, se for diferente, o grafo é dito assimétrico [Bispo 2018]. Um grafo $G = (V,E)$ é chamado de completo se possuir ao menos uma ligação entre cada par de vértices [Boaventura Netto e Jurkiewicz 2009]. A Figura 1 mostra um exemplo de grafo completo.

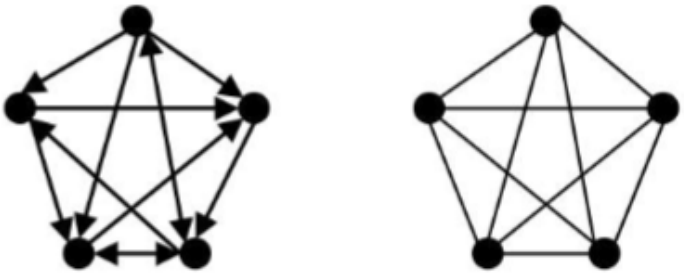

Figura 1. Representação gráfica de um grafo completo composto por seus vértices e arestas [Boaventura Netto e Jurkiewicz 2009]

Cada viagem realizada pelo caixeiro viajante pode ser representada como um grafo onde cada destino é um vértice e, se houver uma rota direta que conecte dois destinos distintos, então haverá uma aresta entre esses dois vértices. O PCV é resolvido se houver uma rota mais curta que visite cada destino uma vez e permita ao "vendedor" (caixeiro viajante) voltar ao início [Brucato 2013].

Encontrar a solução ideal do PCV é um processo computacionalmente difícil. Algoritmos exatos foram propostos, mas nem sempre eles podem ser usados devido à alta complexidade computacional e consequente tempo para produzir uma solução ótima. Algoritmos de aproximação heurística foram projetados e podem produzir rapidamente soluções boas, mas não ótimas. Esses métodos buscam fornecer uma compensação entre a complexidade da computação e a qualidade das soluções [Hu et al. 2021].

Uma vantagem importante dos algoritmos exatos que usam Programação Inteira reside na possibilidade de provar que soluções ótimas podem ser obtidas se o algoritmo tiver sucesso na execução. Além disso, informações valiosas sobre os limites inferior e superior em relação a solução ótima são obtidas, mesmo se o algoritmo for finalizado antes de completar sua execução. Outro benefício desses métodos é que eles permitem que partes do espaço de busca em que a solução ótima não pode ser encontrada sejam desconsideradas. Contudo, uma das desvantagens dos métodos exatos é que, para muitos problemas, o tamanho das instâncias que podem ser solucionadas é limitado pelo custo computacional muito elevado onde o tempo de processamento, em função do número de possibilidades, cresce com o tamanho da instância. Além disso, o consumo de memória pode ser muito alto, levando ao término antecipado da aplicação. [Dumitrescu e Stützle 2003].

Devido à complexidade dos métodos exatos, são utilizados métodos heurísticos para a resolução do PCV. As heurísticas consistem em algoritmos que possibilitam a obtenção de soluções boas, que não são ótimas, mas em um tempo bastante inferior quando comparado aos métodos exatos. As heurísticas mais aplicadas no PCV são: heurísticas construtivas, heurísticas de melhoria iterativa e metaheurísticas. Uma maneira para avaliar o comportamento de uma heurística é verificar a relação proporcional entre a pior solução que a heurística pode obter e a solução ótima. [Oliveira 2015]. São fatores decisivos para escolha de uma determinada heurística o tempo de execução e a maleabilidade a respeito da variação em relação à solução ótima [Bispo 2018].

2.1. Métodos de solução do PCV

Existem diversos métodos de solução do PCV. Nas próximas subseções alguns deles são explanados sinteticamente.

2.1.1. Métodos Exatos

Os algoritmos exatos são projetados para encontrar a solução ideal para o PCV, que seria o trajeto com o menor comprimento. Eles são computacionalmente caros porque devem considerar implicitamente todas as soluções para identificar o caminho ótimo. A execução de um algoritmo exato em um computador poderoso por um longo período de tempo pode não ser muito econômico considerando o fato que pode-se encontrar uma solução, dentro de uma pequena porcentagem do ideal, mais rápido em um pequeno microcomputador. Por isso, os algoritmos heurísticos ou aproximados são frequentemente preferidos aos algoritmos exatos para resolver os grandes PCVs que ocorrem na prática [Potvin 1996].

Dentre os métodos exatos para solução do PCV encontram-se o Método de Força Bruta (FB), Método *Branch & Bound* e *Branch & Cut*. Na Tabela 1 é possível verificar que no caso de poucos pontos (vértices) o problema do caixeiro viajante não é tão complicado, porém, quando o problema tem mais pontos, a contagem dos caminhos possíveis já é um número enorme. No caso de 20 pontos é uma quantidade tão grande de cálculos que é impossível resolvê-lo em tempo real. Houve um grande número de estudiosos na história que tentaram encontrar um algoritmo de otimização universal, porém sem sucesso [Štencek 2013].

Tabela 1. Contagem de permutações para resolver PCV com algoritmo exato. Adaptado de Štencek (2013)

Contagem de pontos	Contagem de maneiras possíveis
4	24
8	40 320
12	479 001 600
16	20 922 789 888 000
20	2 432 902 008 176 640 000
25	15 511 210 043 330 985 984 000 000

Problemas que dependam da análise das permutações dos elementos de um conjunto podem exigir algoritmos exponenciais sendo que o número de permutações de um conjunto com n elementos é n! (fatorial), valor que pode ser aproximado pela expressão exponencial $n^n e^{-n}(2\pi n)^{1/2}$. Por outro lado, algoritmos que examinam os sucessores de cada vértice tomado como nova raiz da iteração poderiam ter de examinar $n - 1$ vértices, na primeira iteração até chegar a um único ao final. A soma destes termos é igual a $n(n - 1)/2$, que é um termo quadrático. Portanto, a complexidade do algoritmo do problema do caminho mínimo a partir de um vértice é, no pior caso, da ordem de n^2, ou abreviadamente $O(n^2)$ [Boaventura Netto e Jurkiewicz 2009].

A Tabela 2 mostra as implicações da complexidade exponencial apresentando estimativas do tempo de computação para algoritmos de diversas complexidades, considerando-se uma máquina que consegue solucionar exatamente um algoritmo O(n) de um grafo com 10 vértices em 10^{-6} segundos [Boaventura Netto e Jurkiewicz 2009].

Tabela 2. Tempo de computação para algoritmos de diversas complexidades. [Boaventura Netto e Jurkiewicz 2009]

n	O(n)	$O(n^2)$	$O(n^3)$	$O(2^n)$
10	10^{-6} seg	10^{-4} seg	10^{-3} seg	10^{-3} seg
50	5.10^{-6} seg	$2{,}5.10^{-3}$ seg	0,125 seg	35,7 anos
100	10^{-5} seg	10^{-2} seg	1 seg	4×10^{14} séculos
500	5.10^{-5} seg	0,25 seg	2 min 25 seg	$1{,}03.10^{135}$ séculos

2.1.2. Métodos heurísticos e metaheurísticos

Existem diversos métodos heurísticos e metaheurísticos para a solução do PCV. Na sequência são analisados alguns dos métodos encontrados em trabalhos comparativos da área.

2.1.2.1. Algoritmos genéticos

Os algoritmos genéticos (AG) são técnicas de busca aleatória que simulam alguns dos processos observados na evolução natural desenvolvidos por Holland e seus alunos na Universidade de Michigan nos anos 60 e 70 [Potvin 1996].

Este método é iniciado com um conjunto de soluções, chamado população, representadas por cromossomos. As soluções obtidas de uma população são registradas e utilizadas para formar uma nova população na expectativa de que a nova geração seja melhor do que a anterior em suas características. As soluções empregadas para formar novas soluções são selecionadas de acordo com seus atributos de aptidão onde, quanto mais adequados eles são, maior a probabilidade de se replicarem. Esta ação é repetida até que certas condições sejam satisfeitas [Mukhairez e Maghari 2015]. O mecanismo de busca abrange as seguintes fases: avaliação da adequação de cada cromossomo, seleção dos cromossomos pais e aplicação dos operadores de mutação e recombinação aos cromossomos pais. Os novos cromossomos resultantes dessas operações formam a próxima geração e esse processo é repetido até que o sistema não consiga mais melhorar [Potvin 1996].

2.1.2.2. Colônia de formigas

No método de Otimização Colônia de Formigas (OCF) as buscas são distribuídas por agentes com capacidades básicas simples que imitam o comportamento de formigas reais. A forma utilizada por elas para comunicação entre os indivíduos sobre os caminhos e para decidir para onde ir, consiste em trilhas de feromônios depositados no solo. Enquanto uma formiga isolada se move ao acaso, uma formiga que encontra uma trilha previamente colocada pode detectá-la e tem grande probabilidade de segui-la, reforçando assim a trilha com seu próprio feromônio. Dessa forma surge um

comportamento coletivo onde, quanto mais as formigas seguem uma trilha, mais atraente essa trilha se torna para ser seguida. Portanto, o processo pode ser caracterizado por um ciclo de feedback positivo, onde a probabilidade de uma formiga escolher um caminho aumenta com o número de formigas que escolheram o mesmo caminho anteriormente [Dorigo et al. 1996].

Como o interesse desse método é no uso de colônias artificiais como ferramenta de otimização, o sistema tem algumas diferenças para um real, no sentido de que as formigas artificiais tem alguma memória, não são completamente cegas e vivem em um ambiente onde o tempo é discreto [Dorigo et al. 1996].

2.1.2.3. Busca tabu

A Busca Tabu (BT) é uma técnica de otimização usada para obter soluções quase ótimas de problemas de otimização combinatória [Karamcheti e Malek 1991]. Nesta técnica, inicia-se com uma permutação arbitrária e faz-se uma sucessão de movimentos para transformar essa permutação em uma ótima ou o mais próximo disso. Isto é equivalente a começar com um passeio gerado aleatoriamente e fazer uma sucessão de trocas de borda tentando reduzir o custo do passeio até chegar no custo mínimo. Pode-se usar a troca 2-opt como o movimento básico [Malek et al. 1989].

Este método apresenta um elemento para restringir a pesquisa, classificando alguns de seus movimentos como proibidos (ou seja, tabu), e um elemento para liberar a pesquisa por uma função de memória de curto prazo que fornece um "esquecimento estratégico" [Glover 1989].

2.1.2.4. Recozimento simulado

O Recozimento Simulado (RS) ou *Simulated Annealing* se origina da analogia entre o processo de recozimento físico e o problema de encontrar soluções mínimas para problemas de minimização discretos. O processo de recozimento é conhecido na física como um processo térmico para obter estados de baixa energia de um sólido em um banho de calor [Dekkers e Aarts 1991]. Neste método, a temperatura do sistema é lentamente reduzida a zero e, caso ela seja diminuída lentamente o suficiente, o sistema deve terminar entre os estados de energia mínima ou suficientemente baixa. Portanto, o algoritmo de recozimento pode ser entendido como um processo para minimizar uma função de custo (energia) sobre um conjunto finito (os estados do sistema) [Gelfand e Mitter 1985].

2.1.2.5. Rede Neural de Hopfield

O projeto de redes neurais artificiais geralmente se baseia em sua semelhança com o cérebro humano. As Redes neurais do tipo Hopfield (RNH) podem ser usadas com sucesso na resolução de problemas de otimização sendo compostas de elementos analógicos altamente interconectados, os neurônios [Mandziuk e Macuk 1992]. As informações de estado de saída do neurônio são decididas com base nas informações de entrada de estado de uma comunidade de neurônios. Cada neurônio troca informações com outros neurônios da rede, os quais aplicam essas informações para conduzir a rede a alcançar a convergência. A operação da rede neural de Hopfield é um processo de relaxamento que permite que uma função de energia alcance uma solução ótima com

menos computação [Chen e Huang 2001].

2.1.2.6. Vizinho mais próximo

O método do vizinho mais próximo (VMP) é uma heurística simples e rápida para a construção de um tour do PCV. O objetivo é encontrar um passeio de duração mínima [Hurkens e Woeginger 2004]. Neste método parte-se de uma cidade inicial escolhida arbitrariamente, sendo escolhida repetidamente para a próxima cidade, a cidade mais próxima da atual que ainda não foi visitada. Uma vez que todas as cidades tenham sido escolhidas, o passeio é encerrado retornando-se à cidade inicial [Johnson 1990].

2.1.2.7. 2-OPT

O algoritmo 2-Opt é provavelmente a heurística de pesquisa local mais básica para o PCV. Ele consegue obter bons resultados tanto com relação ao tempo de execução quanto à razão de aproximação. Nesse método, inicialmente é gerado um passeio arbitrário que é melhorado através de melhorias sucessivas que trocam duas das arestas do percurso por duas outras arestas. Em cada etapa, o algoritmo seleciona duas arestas que sejam distintas e aparecem em certa ordem no passeio e, em seguida, substitui essas arestas por outras, desde que essa alteração diminua a duração do passeio. O algoritmo termina quando nenhuma outra etapa de melhoria adicional é possível [Englert et al. 2014].

2.1.2.8. Otimização da Fisiologia da Árvore

A Otimização da Fisiologia da Árvore (OFA) é um algoritmo metaheurístico inspirado em um sistema de crescimento de plantas através do conceito de brotos e raízes. Nas plantas, os brotos se estendem para encontrar a luz solar enquanto que as raízes, em direção oposta, buscam água e nutrientes para a sobrevivência dos brotos. Assim, a sustentabilidade da planta é assegurada pela associação desses dois sistemas. Esta é a ideia que foi utilizada no algoritmo onde os brotos com ramos definidos encontram uma solução potencial com a ajuda da variável raízes [Halim e Ismail 2019].

2.2. Trabalhos relacionados

Na busca por um método de solução do PCV que pudesse ser utilizado neste projeto, foi realizada uma pesquisa de trabalhos que apresentassem comparações entre as técnicas para possibilitar o embasamento desta escolha.

No trabalho de Mukhairez e Maghari (2015) foi realizada uma comparação de desempenho em termos de tempo de execução e menor distância dos métodos de recozimento simulado, algoritmos genéticos e otimização de colônia de formigas. Estes autores apontam que o recozimento simulado tem o menor tempo de execução mas, para a menor distância, ele fica em segundo lugar. Já em termos de distância mais curta entre as cidades, colônia de formigas tem melhor desempenho, porém fica na última posição em termos de tempo de execução.

No artigo de Pasquier et al. (2007) é apresentado um estudo comparativo de três metaheurísticas implementadas para resolver o PCV: Algoritmos Genéticos (AG), Otimização de Colônia de Formigas (OCF) e Recozimento Simulado (RS). Em relação

ao RS, foi observado que ele apresenta simplicidade de implementação e facilidade de otimização, mas a qualidade de seus resultados e tempo de processamento são maiores do que as melhores soluções médias obtidas. Em relação ao AG, os resultados são considerados aceitáveis, porém carece de estabilidade. Sua implementação foi excepcionalmente complicada em comparação com os outros, mas oferece o melhor tempo de processamento. O estudo conclui sobre a vantagem de aplicar a OCF que oferece a melhor qualidade dos resultados, uma implementação relativamente simples e uma relativa facilidade de otimização, mas não o melhor tempo de processamento.

O artigo de Hui (2012) apresenta a comparação entre Algoritmo Genético básico (AG), Rede Neural de Hopfield (RNH) e algoritmo básico de Colônia de Formigas (OCF) para resolver o PCV. Foi encontrado que a complexidade de tempo do algoritmo genético é a mais alta, e a menor é do algoritmo da colônia de formigas. A complexidade espacial do algoritmo genético é a maior, sendo que as redes de Hopfield e o algoritmo de colônia de formigas têm a mesma complexidade espacial. Por causa dos operadores de *crossover* e mutação, o algoritmo genético pode escapar do ótimo local e tornar possível encontrar o ótimo global. O resultado da rede de Hopfield e algoritmos de colônia de formigas pode ser bom ou ruim dependendo do parâmetro, pois o valor do parâmetro afeta diretamente a convergência do algoritmo e os resultados dos prós e contras. E esses dois algoritmos são fáceis de colocar em um ótimo local. A quantidade de linhas de código dos três algoritmos implementados no MATLAB e usadas para refletir a quantidade de implementação do algoritmo, são 186, 93 e 96 demonstrando que o mais difícil é o algoritmo genético. O resultado do algoritmo de colônia de formigas é mais próximo da solução ótima e os resultados do algoritmo genético e da rede de Hopfield são bastante diferentes do valor do resultado ótimo. Por fim, as pontuações totais consolidadas dos algoritmos genéticos, redes de Hopfield e algoritmos de colônia de formigas foram 0,29, 0,18, 0,53, respectivamente, que o representa que o melhor algoritmo é o colônia de formigas, o segundo é o algoritmo genético, o terceiro é a rede de Hopfield.

No trabalho de Bispo (2015) o problema da roteirização do turismo de Recife fundamentado no PCV em um aplicativo móvel foi testado com 3 métodos para sua resolução sendo eles, algoritmo Força Bruta (FB), Vizinho Mais Próximo (VMP) e Vizinho Mais Próximo combinado à 2-OPT. O algoritmo Força bruta teve uma tendência de crescer o tempo de execução de forma explosiva conforme a quantidade de pontos aumenta, sendo viável para até 8 pontos, considerando 5 segundos um tempo de espera razoável, sendo que o teste feito com 11 pontos mostrou-se impraticável com um consumo muito elevado de memória e CPU, o que resultou na interrupção da aplicação. O vizinho mais próximo desempenhou rapidamente o seu papel, porém a cada ponto adicionado o roteiro ficou mais distante do ótimo. Contudo, o custo com memória e CPU foi inferior comparado ao anterior. Ainda, o algoritmo 2-OPT revelou um ganho de até 50,4 minutos e conseguiu melhorar em até 20% a rota gerada pelo vizinho mais próximo. Os testes de CPU e memória foram efetuados com os 20 pontos turísticos cadastrados no aplicativo. Houve um aumento de memória em 1.6 MB após o toque de gerar roteiro. Mesmo assim, o processo perdurou pouco mais de 1 segundo, sendo considerado o mais rápido dentre os executados.

No trabalho de Halim e Ismail (2017) são utilizados 15 problemas de PCV e são

comparados o tempo de computação, o valor estatístico e a convergência utilizando os algoritmos Vizinho Mais Próximo (VMP), Algoritmo Genético (AG), Recozimento Simulado (RS), Busca Tabu (BT), Otimização de Colônia de Formigas (ACO) e Otimização de Fisiologia de Árvores (TPO). Para o tempo de computação, foi encontrado que os métodos Busca Tabu e Otimização de Colônia de Formigas requerem 6 e 3 vezes, respectivamente, mais do que outros algoritmos. Nesta avaliação, entre os algoritmos computados mais rápidos estão o Vizinho Mais Próximo, seguido pelo Otimização de Fisiologia de Árvores e Algoritmo Genético. Já na comparação estatística, os algoritmos que chegaram mais perto da rota ótima são Busca Tabu, Otimização de Fisiologia de Árvores e Algoritmo Genético. Porém, constataram que a precisão do Busca Tabu deprecia em problemas de tamanhos maiores. Para convergência dinâmica, os autores verificaram que os algoritmos que conseguem convergir mais rápido em 300 iterações são Vizinho Mais Próximo, Busca Tabu e Otimização de Colônia de Formigas, sendo que a Otimização de Fisiologia de Árvores pode convergir para melhores resultados em tamanhos menores. Além disso, relataram que o Recozimento Simulado tem a dinâmica de convergência mais lenta em comparação com outros algoritmos.

Além de comparações entre alguns métodos de solução do PCV, foi realizada uma pesquisa de trabalhos que relacionassem o PCV com aplicações no ramo do turismo. No artigo de Faizah et al. (2019) foi utilizado o algoritmo de colônia de formigas para encontrar a rota de viagem mais curta do turismo em Palembang utilizando aplicativo móvel com sistema operacional Android e API do Google Maps. No trabalho de Zacarias et al. (2015) foi criado um aplicativo voltado para os turistas que visitam a Cidade de Puebla e a Cidade do México sendo que para a geração do mapa, usou-se o algoritmo do Google Maps e também o algoritmo do caixeiro viajante para encontrar a rota ideal para visitar os locais escolhidos pelos turistas. Ainda, no artigo de Montoya (2020) buscou-se determinar uma rota boa o suficiente para a seleção personalizada de destinos de viagem para o turismo em Honduras usando um modelo básico de Problema do Caixeiro Viajante resolvido com a implementação da heurística do vizinho mais próximo utilizando o recurso macros do *software* Excel.

3. Metodologia

As características dos métodos exatos e heurísticos devem ser levadas em consideração para escolha dos métodos para utilização na implementação da solução do PCV. Em se tratando da aplicação dos métodos associados ao desenvolvimento de uma aplicação móvel, as particularidades dos dispositivos móveis também interferem na escolha do método mais adequado.

As características dos dispositivos móveis estabelecem as restrições que influenciam a maioria das decisões técnicas e não técnicas no desenvolvimento de aplicativos de software para dispositivos móveis. Dentre as características pode-se destacar: tamanho de tela pequeno, tamanho de memória pequeno, capacidade de bateria pequena, limitações de largura de banda de rede, possuir vários acessórios, sensores, GPS, acelerômetro, sensor de luz e câmeras digitais, ter vários protocolos de rede, possuir interação com o mundo real, estar sempre ligado e disponível, uso individual e pessoal [Fernandes e Ferreira 2016].

Considerando esses fatores, é preciso realizar uma escolha contextualizada visto que existem diversas soluções possíveis a depender do objetivo de aplicação, dos recursos computacionais, etc. Portanto, faz-se necessário uma análise criteriosa para determinação da solução mais viável para o PCV no escopo deste trabalho. Nesse sentido, a partir do levantamento bibliográfico de métodos exatos, heurísticos e metaheurísticos que buscam solucionar o problema do caixeiro viajante, foi realizada uma análise sobre qual dos métodos poderia ser utilizado na aplicação.

Para a escolha do método a ser utilizado na aplicação, alguns critérios foram considerados, são eles: a capacidade de geração de uma solução próxima da ideal; o baixo tempo de execução, com o intuito de melhorar a experiência do usuário; e o custo computacional e a complexidade de implementação baixos, possibilitando a utilização adequada em dispositivos móveis.

Analisando os dados obtidos nos trabalhos relacionados foi possível elencar grupos de métodos que possuem características semelhantes e que poderiam vir a ser escolhas pertinentes para utilização no projeto, segundo os critérios já citados.

Os algoritmos utilizados nas análises foram: Força Bruta (FB), Algoritmos genéticos (AG), Otimização Colônia de formigas (OCF), Recozimento simulado (RS), Rede Neural de Hopfield (RNH), Vizinho mais próximo (VMP), 2-opt, Busca tabu (BT), Otimização de Fisiologia de Árvores (OFA).

O algoritmo exato Força Bruta (FB) foi o único enquadrado entre os métodos que apresentam solução ótima, porém apresenta um consumo muito elevado de memória e CPU e o tempo de execução cresce se a quantidade de pontos aumenta.

Os demais métodos que não apresentam solução ótima estão no grupo dos métodos heurísticos, onde dividiu-se os algoritmos que apresentam baixo tempo de execução: Recozimento simulado e Vizinho mais próximo.

Ainda, nesse mesmo grupo, em relação ao custo computacional/complexidade de implementação percebe-se que os métodos apresentam baixo custo computacional, porém destacam-se algumas outras características. Quanto ao tempo de processamento, AG tem rápida capacidade de busca global, OCF tem longo tempo de execução, RS e VMP tem baixo tempo de execução, BT e OFA tem tempo de computação de 6 a 3 vezes maiores que outros.

Dentro desse grupo destacou-se o método do Vizinho Mais Próximo que, embora não garanta uma solução ótima, consegue chegar bastante próximo a isso, além de possuir um tempo de convergência dinâmica e um tempo de computação rápidos e de apresentar a complexidade do algoritmo quadrática que, no pior caso, é da ordem de n^2 ou $O(n^2)$. Além disso, uma característica importante do algoritmo deste método, determinante para a sua escolha, é o fato deste ser construído partindo-se de um ponto inicial, pois a definição deste ponto é importante para a aplicação para que o usuário possa optar pelo ponto turístico de onde deseja iniciar a rota.

Segundo Huang e Yu (2017) as heurísticas baseadas em nós, como no caso do vizinho mais próximo, constroem um circuito, expandindo os nós em um conjunto de pontos Q, um por um, até que todos sejam visitados. A estrutura dessas heurísticas está representada no pseudo-algoritmo da Figura 2.

```
    Entrada: Q: conjunto de pontos do PCV
    Saída: T: PCV para Q
1   início
2       T ← init(Q);
3       enquanto T não contém todos os nós em Q faça
4           v ← selecionar(Q, T);
5           inserir(v, T);
6       retorne T;
```

Figura 2. Estrutura de heurísticas baseadas em nós. Adaptado de Huang e Yu (2017)

A heurística do vizinho mais próximo não consome tempo para encontrar um conjunto de pontos PCV inicial para init(Q), escolhendo-se aleatoriamente um nó de Q. Desse modo, T para init(Q) contém apenas um nó selecionado aleatoriamente. Em cada iteração seguinte, é escolhido um ponto entre os nós que ainda não foram selecionados, o qual é inserido na extremidade do caminho atual computado na iteração anterior [Huang e Yu 2017].

Baseado nesse conceito, adaptou-se o algoritmo concedendo a opção de escolha do nó inicial ao usuário do aplicativo ao invés de ele ser gerado aleatoriamente. Em suma, a lógica de programação desenvolvida pelo algoritmo em questão segue as seguintes etapas:

1. Seleção do ponto turístico que será definido como ponto inicial;
2. Seleção do ponto não visitado mais próximo;
3. Classificação do ponto visitado atualmente como visitado;
4. Se houverem pontos não visitados, retorno para a etapa 2;
5. Se todos os pontos foram visitados encerra-se a execução.

Após a definição do método, buscou-se os pontos que serviriam para formação do roteiro. Foram utilizados 20 pontos turísticos constantes no mapa turístico da cidade de Veranópolis/RS, que são os seguintes: Ponte Ernesto Dornelles, Belvedere do Espigão, Capela São João Batista, Grutinha N. Sra. de Lourdes, Capela N. Sra de Monte Bérico, Cascata da Usina Velha, Capitel São Marco, Caverna Indígena, Torre Mirante da Serra, Igreja Medianeira, Arco Sul, Gruta Nossa Sra. De Lourdes, Aeroclube Veranópolis, Casa da Cultura, Casa Saretta, Igreja Matriz São Luiz Gonzaga, Portal Monumento, Vila Bernardi, Arco Norte, Balneário Retiro [Município de Veranópolis 2021].

Com a definição dos pontos turísticos, foi realizado um levantamento de imagens e informações principais sobre eles para inclusão na aplicação. Além disso, cada um dos pontos foi localizado no mapa com a ferramenta Google Maps para obtenção de sua latitude e longitude. Essas coordenadas geográficas foram utilizadas para o cálculo das distâncias entre os pontos no algoritmo, a partir das quais ocorre a definição do próximo ponto a ser visitado.

Para elaboração do protótipo do aplicativo móvel foi utilizado o Flutter™. O

Flutter é o kit de ferramentas de IU do Google de código aberto para criação de interfaces nativas no iOS e Android. A base do Flutter é o Dart que fornece a linguagem e os tempos de execução que impulsionam os aplicativos. Como ambiente de desenvolvimento do aplicativo foi usado o software Android Studio versão 3.6, sendo projetado para o sistema operacional Android.

Definido o método e as ferramentas necessárias para a elaboração do projeto, foi executada a fase de modelagem do sistema desenhando as funcionalidades básicas necessárias para seu adequado funcionamento. A Figura 3 mostra o diagrama de classes e na Figura 4 é apresentado um diagrama que mostra a sequência de eventos que ocorrem durante o processo de execução do aplicativo para a criação do roteiro turístico.

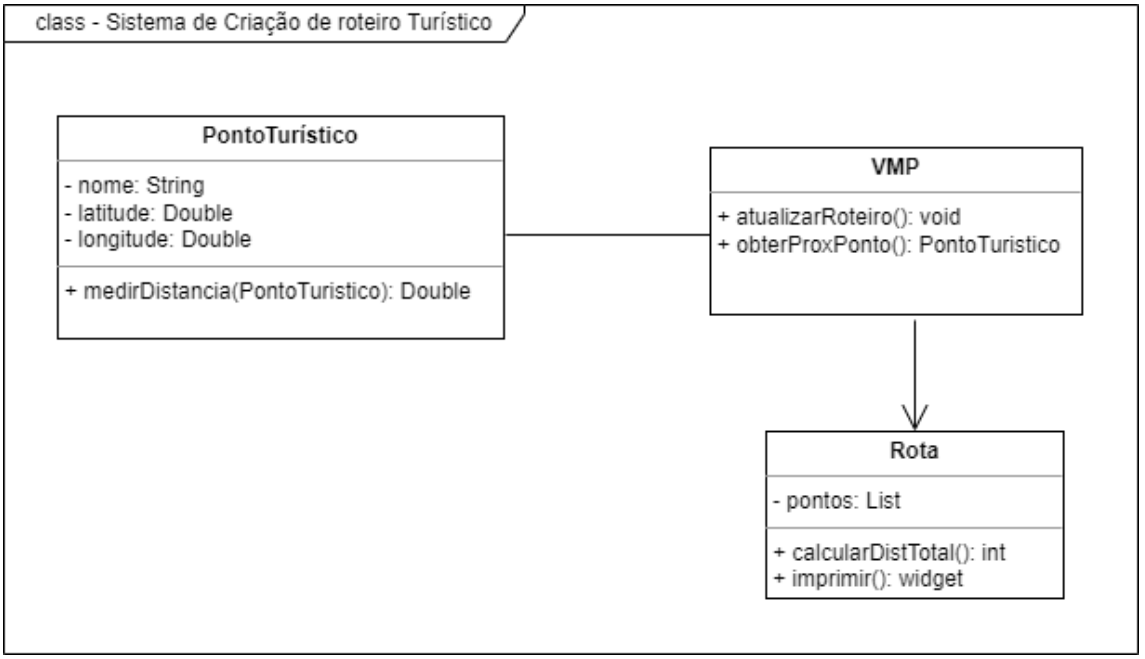

Figura 3. Diagrama de classes [autor]

No início do processo de criação de um roteiro turístico, apresentado na Figura 4, o usuário seleciona um ponto turístico inicial criando um objeto da classe *PontoTurístico*. O mesmo ocorre com os demais pontos selecionados. Em seguida, quando o usuário seleciona a opção de cálculo do trajeto, os objetos são incluídos no objeto da classe *Vizinho Mais Próximo (VMP)* e, através de um laço de repetição (loop), para cada ponto, é realizado o cálculo do roteiro e os pontos são inseridos no objeto da classe *Roteiro*. Por fim, o retorno é apresentado na interface do sistema.

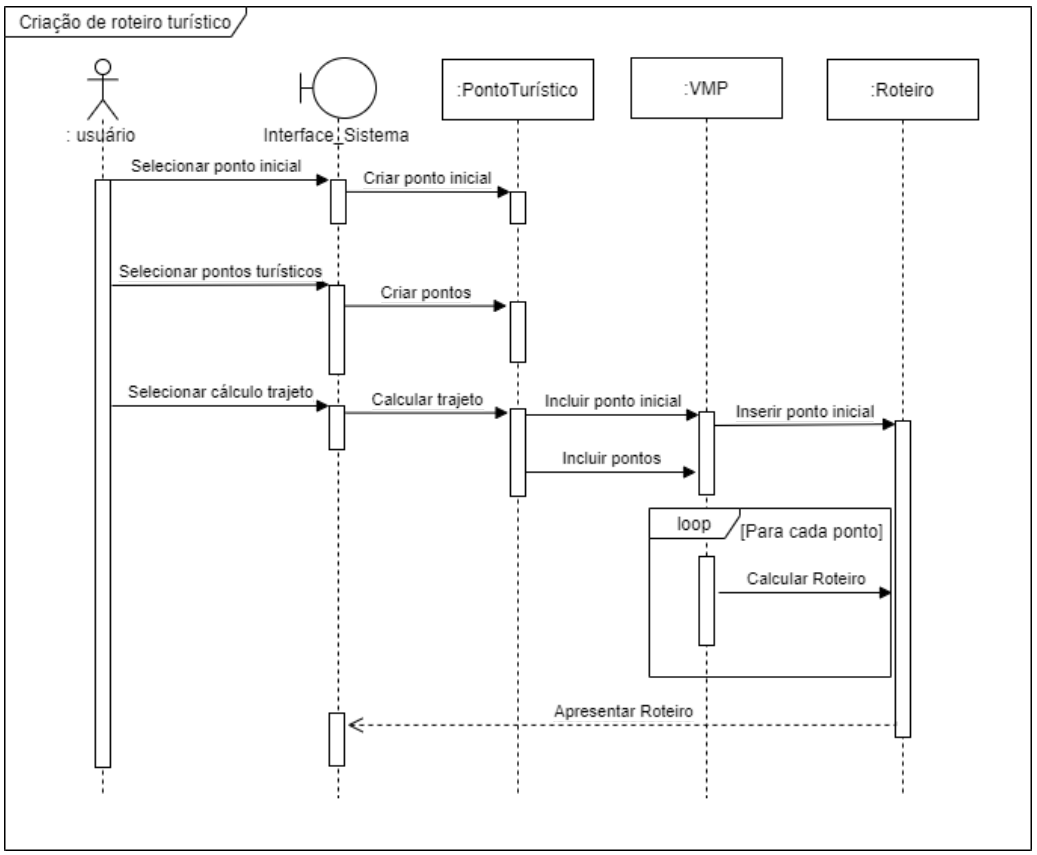

Figura 4. Diagrama de sequência [autor]

4. Apresentação e discussão dos resultados

O protótipo de aplicativo foi constituído com uma página que possui uma lista dos pontos turísticos, contendo foto, nome e breve descrição, o que permite a interação do usuário para a escolha dos pontos que deseja visitar, assim como a possibilidade de escolha do ponto que será o início do trajeto. Após essas escolhas, o botão "Calcular trajeto" leva à outra página que mostra uma lista de pontos turísticos ordenados conforme o resultado da rota calculada. Ainda, um botão "Ver mapa" leva a uma tela onde é apresentado um mapa, gerado através do Google Maps, onde estão marcados os pontos turísticos do trajeto. A Figura 5 mostra as principais telas do aplicativo.

O protótipo foi utilizado e validado através do emulador do Android Studio e também em 3 dispositivos físicos: Asus Zenfone 3 com memória RAM de 3GB e CPU Octa-core 2.0 GHz, Samsung Galaxy J2 Core com memória RAM de 1.0 GB e CPU Quad-core de 1.4GHz e Xiaomi Poco X3 NFC com memória RAM de 6GB e CPU Octa-core Max 2.30 GHz. Com isso, foi possível constatar que o aplicativo foi capaz de executar todas as ações propostas gerando roteiros de acordo com os pontos turísticos selecionados pelo usuário.

Dado o aspecto dos métodos heurísticos de solução do PCV de apresentar soluções boas porém não ótimas [Hu et al. 2021], entende-se que os resultados da execução do algoritmo do vizinho mais próximo no aplicativo móvel geraram roteiros com essa característica.

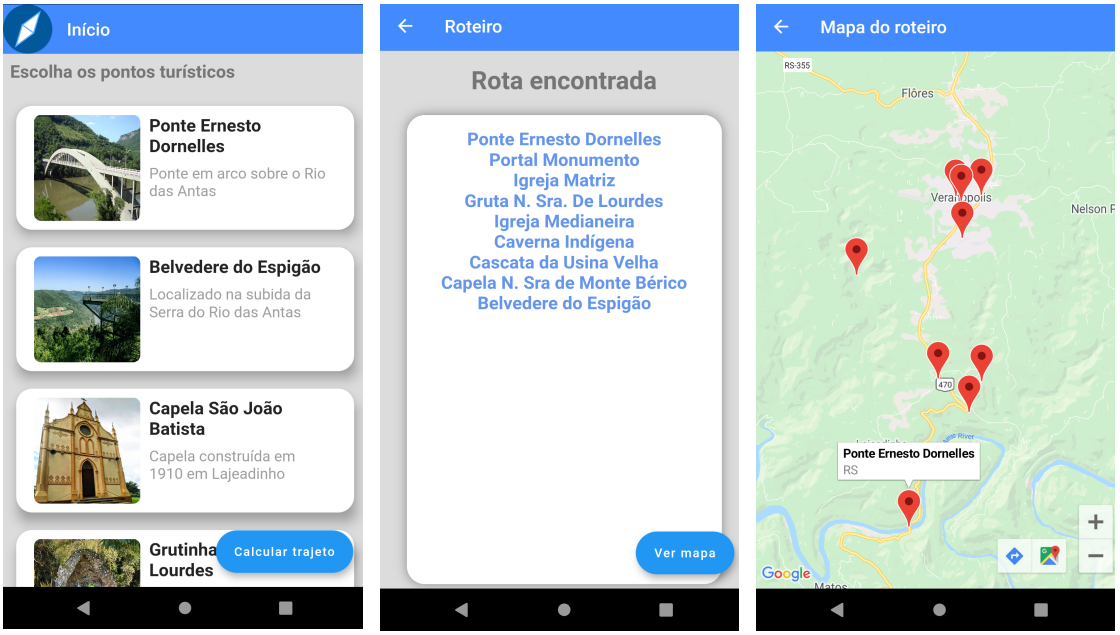

Figura 5. Principais telas do protótipo do aplicativo desenvolvido [autor]

Em relação ao desempenho do protótipo quanto ao tempo de computação, não foram realizados testes quantitativos, apenas dimensionado através da experiência do usuário. Contudo, foi observado que o algoritmo conseguiu apresentar em tela os roteiros gerados, independente do número de pontos selecionados, de maneira imediata após a interação do usuário no botão de calcular trajeto. Essa observação confirma o que postulam Halim e Ismail (2017) sobre o algoritmo vizinho mais próximo ser um dos computados mais rapidamente na análise comparativa dos autores e ainda pelo autor Bisbo (2015) que atesta que o algoritmo desempenha rapidamente o seu papel.

5. Considerações finais

Este trabalho buscou o desenvolvimento de uma solução para realizar o cálculo de rotas turísticas utilizando o problema do caixeiro viajante sendo validado através de um protótipo de aplicativo móvel.

O protótipo do aplicativo desenvolvido se mostrou capaz de executar as funções de maneira satisfatória possibilitando uma micro experiência do usuário. O algoritmo foi capaz de gerar um roteiro com pontos turísticos de interesse do usuário através do método do vizinho mais próximo, apresentando características dos métodos heurísticos sendo a solução próxima da ideal. Com isso foi possível verificar a viabilidade de utilização desta aplicação.

Para trabalhos futuros, pretende-se explorar a combinação de métodos do problema do caixeiro viajante para aperfeiçoar a solução agregando as qualidades das heurísticas a fim de obter um resultado cada vez mais próximo do ideal. Ainda, almeja-se prosseguir o desenvolvimento da aplicação incluindo funcionalidades e aprimorando o layout.

6. Referências Bibliográficas

Applegate, D .L., Bixby, R. E., Chvátal, V. e Cook, W. (2007) "The Traveling Salesman Problem: A Computational Study". Princeton University Press, Princeton, NJ.

Bispo, R. C. (2018), "Planejador de Roteiros Turísticos: Uma Aplicação do Problema do Caixeiro Viajante na Cidade do Recife". Universidade Federal Rural de Pernambuco UFRPE, Curso de Bacharelado em Sistemas de Informação, Recife. (Trabalho de conclusão de curso).

Boaventura Netto, P. O. e Jurkiewicz, S. (2009) Grafos: Introdução e Prática. São Paulo, Editora Blucher, 2ª ed.

Brucato, Corinne (2013) "The Traveling Salesman Problem". University of Pittsburgh, Department of Mathematics, Pittsburgh. (Master thesis).

Chen, R.-M., Huang, Y.-M. (2001) "Competitive neural network to solve scheduling problems". Neurocomputing, Volume 37, Issues 1–4, Pages 177-196.

Dekkers A. and Aarts E. (1991) "Global optimization and simulated annealing". Mathematical Programming 50 North-Holland 367-393.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996) "Ant System: Optimization by a Colony of Cooperating Agents". IEEE Transactions On Systems, Man, And Cybernetics-part B Cybernetics, Vol 26, No 1, February.

Dumitrescu I e Stützle T. (2003) "A survey of methods that combine local search and exact algorithms".

Englert, M., Röglin, H. and Vöcking, B. (2014) "Worst Case and Probabilistic Analysis of the 2-Opt Algorithm for the TSP". Algorithmica 68, 190–264.

Faizah, S., Novianti, L., Kom, S., Kom, M.; Novita, N. (2019) "The Implementation of Ant Colony Algorithm In Finding The Shortest Travel Route of Palembang Tourism By Android Based". Journal of Physics: Conference Series, vol. 1167.

FLUTTER (2021) Disponível em https://flutter.dev/docs.

Gelfand, S. B. and Mitter, S. K. (1985) "Analysis Of Simulated Annealing For Optimization". Proceedings of the IEEE Conference on Decision and Control.

Glover, F. (1989) "Tabu Search-Part I". ORSA Journal on Computing Vol. 1, No. 3, Summer.

Halim, A.H., Ismail, I. (2017) "Combinatorial Optimization: Comparison of Heuristic Algorithms in Travelling Salesman Problem". Archives of Computational Methods in Engineering 26, 367–380 .

Halim, A.H., Ismail, I. (2019) "Tree physiology optimization on SISO and MIMO PID control tuning". Neural Comput & Applic 31, 7571–7581.

Hu Y, Zhang Z, Yao Y et al (2021) "A bidirectional graph neural network for traveling salesman problems on arbitrary symmetric graphs". Engineering Applications of Artificial Intelligence, Vol. 97.

Huang, W., Yu, J.X. (2017) "Investigating TSP Heuristics for Location-Based Services". Data Science and Engineering 2, pg 71–93.

Hui, Wang (2012) "Comparison of several intelligent algorithms for solving TSP problem in industrial engineering". Systems Engineering Procedia vol 4 pg 226 – 235.

Hurkens, C.A.J. and Woeginger G. J. (2004) “On the nearest neighbor rule for the traveling salesman problem”. Operations Research Letters, Volume 32, Issue 1, Pages 1-4.

Ignarra, L. R. (2013) Fundamentos do turismo. 3 ed. São Paulo, Cengage Learning, pg 76.

Johnson D.S. (1990) “Local optimization and the Traveling Salesman Problem”. In: Paterson M.S. (eds) Automata, Languages and Programming. ICALP 1990. Lecture Notes in Computer Science, vol 443. Springer, Berlin, Heidelberg.

Karamcheti, V. and Malek, M. (1991) “A TSP engine for performing tabu search”. Proceedings of the International Conference on Application Specific Array Processors, pp. 309-321.

Karp, R. M. (1972) “Reducibility among combinatorial problems”. In: Complexity of computer computations. [S.l.]: Springer p. 85–103.

Malek, M., Guruswamy, M., Pandya, M. et al. (1989) “Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem”. Ann Oper Res 21, 59–84.

Mandziuk, J., Macuk, B. (1992) A neural network designed to solve the N-Queens Problem. Biol. Cybern. 66, 375–379.

Montoya, M., Palma, B., Sandoval, N., Fúnez, H. (2020) “Optimization of tourism route for Honduras Optimización de ruta para recorrido turístico en Honduras”. 18 Th LACCEI International Multi-Conference for Engineering, Education, and Technology.

Mukhairez, H. H. A. and Maghari, A. Y. A. (2015) “Performance Comparison of Simulated Annealing, GA and ACO Applied to TSP”. International Journal of Intelligent Computing Research (IJICR), Volume 6, Issue 4, December.

Município de Veranópolis. (2021) Turismo. Disponível em http://www.veranopolis.rs.gov.br/cidade/12/turismo, julho.

Oliveira, A. (2015) “Extensões do Problema do Caixeiro Viajante”. Universidade de Coimbra, Departamento de Matemática, Coimbra. (Dissertação de mestrado).

Pasquier, J. L., Balich, I. K., Carr D. W. and López-Martín, C. (2007) “A Comparative Study of Three Metaheuristics Applied to the Traveling Salesman Problem,” Sixth Mexican International Conference on Artificial Intelligence, Special Session (MICAI), pp. 243-254.

Potvin, J. -Y. (1996) “Genetic algorithms for the traveling salesman problem”. Annals of Operations Research.

Štencek, J. (2013), “Traveling salesman problem Degree Programme in Software engineering”. JAMK University of Applied sciences, Degree Programme in Software engineering (Bachelor’s Thesis).

Yai-Fung Lim, Pei-Yee Hong, Razamin Ramli, Ruzelan Khalid and Md. Azizul Baten, (2016) “Performance Evaluation of Heuristic Methods in Solving Symmetric Travelling Salesman Problems”. Journal of Artificial Intelligence, 9: 12-22.

Zacarias, F., Cuapa, R., Ita, G. De, Torres D.(2015) Smart Tourism in 1-Click. Procedia Computer Science, Volume 56, pg 447-452.