Coordenação de Múltiplos Veículos Autônomos de Entrega Usando *K-Means* e Algoritmos Bio-Inspirados

Clênio Eduardo da Silva

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2020

Clênio Eduardo da Silva

Coordenação de Múltiplos Veículos Autônomos de Entrega Usando *K-Means* e Algoritmos Bio-Inspirados

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Prof. Dr. Jefferson Rodrigo de Souza
Coorientador: Prof. Dr. Raulcézar Maximiano Figueira Alves

Uberlândia
2020

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

S586 Silva, Clênio Eduardo da, 1994-
2020 Coordenação de Múltiplos Veículos Autônomos de Entrega Usando K-Means e Algoritmos Bio-Inspirados [recurso eletrônico] / Clênio Eduardo da Silva. - 2020.

Orientador: Jefferson Rodrigo de Souza.
Coorientador: Raulcézar Maximiano Figueira Alves.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2020.563
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Souza, Jefferson Rodrigo de,1985-, (Orient.). II. Alves, Raulcézar Maximiano Figueira,1984-, (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. IV. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Mestrado Acadêmico, 24/2020, PPGCO				
Data:	29 de julho de 2020	Hora de início:	**13h32min**	Hora de encerramento:	**17h45min**
Matrícula do Discente:	11822CCP004				
Nome do Discente:	Clênio Eduardo da Silva				
Título do Trabalho:	Coordenação de Múltiplos Veículos Autônomos de Entrega Usando K-means e Algoritmos Bio-inspirados				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Inteligência Artificial				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Luiz Gustavo Almeida Martins - FACOM/UFU; Geraldo Pereira Rocha Filho - DCC/UNB; Raulcézar Maximiniano Figueira Alves - CTI/UFU (coorientador) e Jefferson Rodrigo Souza - FACOM/UFU, orientador do candidato.

Os examinadores participaram desde as seguintes localidades: Geraldo Pereira Rocha Filho - Brasília -DF; Luiz Gustavo Almeida Martins, Raulcézar Maximiniano Figueira Alves e Jefferson Rodrigo de Souza - Uberlândia-MG. O discente participou da cidade de Uberlândia-MG.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Jefferson Rodrigo de Souza, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos,

conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Jefferson Rodrigo de Souza**, **Professor(a) do Magistério Superior**, em 30/07/2020, às 10:15, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Raulcézar Maximiano Figueira Alves**, **Analista de Tecnologia da Informação**, em 30/07/2020, às 10:28, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Luiz Gustavo Almeida Martins**, **Professor(a) do Magistério Superior**, em 30/07/2020, às 16:02, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Geraldo Pereira Rocha Filho**, **Usuário Externo**, em 26/08/2020, às 21:10, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2144147** e o código CRC **57B6B3EF**.

Referência: Processo nº 23117.040956/2020-94 SEI nº 2144147

Dedico este trabalho à minha mãe Ângela, exemplo de pessoa em minha vida e cujo empenho em nos educar sempre veio em primeiro lugar, ao meu querido pai Manoel (in memoriam), cuja a vida dedicou ao trabalho em prol do nosso bem estar. Também aos meus irmãos, Cleiber, Clésio e Luzia, por todo o apoio ao longo de minha vida.

Agradecimentos

Primeiramente, agradeço a Deus pelo dom da vida e por todas as oportunidades a mim concedidas. Creio que o Senhor vem me acompanhando e guiando cada passo de minha trajetória. Agradeço a mãe de Jesus Cristo que vêm me cobrindo com seu manto sagrado e intercedendo por mim.

Agradeço aos meus pais e irmãos por serem fundamentais em toda minha vida e torcerem por mim em todos os momentos.

Gostaria de agradecer à minha namorada Taís por compreender a importância do meu sonho em cursar um mestrado e, principalmente, por estar ao meu lado e torcendo por mim. Agradeço também à sua mãe Geralda por todo carinho e orações.

Ao Pessoal do Laboratório de Robótica Móvel do ICMC/USP, Tiago, Iago, Júnior que colaboraram no experimento com o veículo CaRINA II. Agradeço também ao Lucas que ofereceu estadia nos dias em que estive em São Carlos.

Ao Professor Jefferson pela orientação, gentileza e compreensão durante o desenvolvimento desse trabalho. Obrigado!

Ao Professor Raul pela coorientação, por me atender prontamente, mesmo em seu horário de trabalho, nas vezes em que fui no CTI-UFU para esclarecimento de dúvidas, por acreditar em mim para continuação dessa pesquisa. Meu muito obrigado!

Aos Professores da Pós Graduação da FACOM/UFU pelos ensinamentos e conhecimentos compartilhados.

Aos secretários de pós-graduação, Sônia e Erisvaldo, pela prestatividade e boa vontade que sempre me auxiliaram.

Aos Professores da Graduação do IFTM Campus Patrocínio pelos ensinamentos e conhecimentos compartilhados e por terem me iniciado no campo da pesquisa científica.

Aos meus amigos do apartamento, Douglas, Leonard e Raphael, os quais fizeram o início desse período de adaptação em Uberlândia mais fácil me proporcionando momentos de distração e parceria.

Aos meus amigos da graduação e equipe das maratonas de programação.

Aos meus amigos da pós-graduação, que tive a oportunidade de troca de conhecimento e pensamentos.

Aos amigos do trabalho, onde pude aprender muito e aplicar um pouco do conhecimento que adquiri no decorrer de minha vida acadêmica.

As crianças do CRAS Geraldo Tuniquinho por entenderem a minha saída em prol dos estudos. Desejo que todos estejam muito bem.

Enfim, agradeço a todos aqueles que sempre me apoiaram e torceram por mim.

"Entrega o teu caminho ao Senhor, confia Nele, e o mais Ele fará."
(Salmo 37.5)

Resumo

Com o surgimento de carros autônomos, várias tarefas podem ser automatizadas, além do transporte de pessoas, como a entrega de mercadorias. Para reduzir custos e esforços, essa tarefa pode ser atribuída a uma frota de carros que deve cobrir um conjunto de locais de entrega. Esta dissertação apresenta o desenvolvimento de uma abordagem híbrida como uma solução para o problema de múltiplos caixeiros viajantes, (do inglês, *multiple Traveling Salesman Problem* - mTSP) aplicado ao escalonamento de rotas para veículos autônomos. Inicialmente, usamos o *K-means* como um pré-processamento para gerar rotas que distribuem os locais de entrega entre os carros. Em seguida, essas rotas são definidas como população inicial para os algoritmos bio-inspirados: Algoritmo Genético (GA) e Colônia de Formigas em sua versão (ACS). Esses algoritmos executam um processo evolutivo para encontrar uma rota que minimize a distância geral, mantendo o equilíbrio das rotas individuais de cada carro. Os experimentos foram conduzidos em um sistema de escalonamento de rotas em ambientes virtuais (simulação) e em um estudo de caso no Campus 2 da Universidade de São Paulo. Nos experimentos, foram realizadas comparações das abordagens híbridas, K-means-GA e K-means-ACS com as suas versões sem pré-processamento, com a geração da população inicial de forma aleatória. Além das comparações com GA e ACS foram realizadas comparações com o algoritmo de Otimização por Enxame de Partículas (PSO). Os resultados apontam que à medida que o número de carros e locais de visita aumentam, as abordagens híbridas superam suas versões clássicas e o PSO. Para avaliação dos resultados foram aplicados um teste não paramétrico *kruskal wallis* seguido de um teste de comparações múltiplas *Dunn-Bonferroni*.

Palavras-chave: Algoritmos Bio-Inspirados, mTSP, K-Means, Planejamento de Caminhos, Sistema de Escalonamento de Rotas, Veículos Autônomos.

Abstract

With the emergence of self-driving cars, several tasks can be automated in addition to transporting people, such as delivering goods. To reduce costs and efforts, this task can be assigned to a fleet of cars that must cover a set of delivery locations. This dissertation presents the development of a hybrid approach as a solution for the multiple Traveling Salesman Problem (mTSP) applied to the route scheduling for self-drive cars. Initially, we used K-means as pre-processing to generate routes that distribute delivery locations between cars. Then, these routes are defined as the initial population for the bio-inspired algorithms: Genetic Algorithm (GA) and Ant Colony in its version (ACS). These algorithms perform an evolutionary process to find a route that minimizes the overall distance, maintaining the balance of the individual routes of each car. The experiments were conducted in the route scheduling system in virtual environments (simulation) and in a case study at Campus 2 of the University of São Paulo. In the experiments, comparisons of the hybrid approaches, K-means-GA and K-means-ACS were made with their versions without pre-processing, with the initial population generation at random. In addition, to comparisons were also made with Particle Swarm Optimization (PSO). The results show that as the number of cars and places increases, the hybrid approaches surpass their classic versions and also the PSO. To evaluate the results, a nonparametric test *kruskal wallis* followed by a test of multiple comparison test *Dunn-Bonferroni* were applied.

Keywords: Bio-Inspired algorithms, mTSP, K-means, Path Planning, Route Scheduling System, Autonomous Vehicles.

Lista de ilustrações

Lista de tabelas

Lista de siglas

ACS *Ant Colony System*

ACO *Ant Colony Optimization*

AS *Ant System*

AM Aprendizado de Máquina

AOBI Algoritmos de Otimização Bio-Inspirados

CaRINA Carro Robótico Inteligente para Navegação Autônoma

CE Computação Evolutiva

GA *Genetic Algorithm*

IA Inteligência Artificial

IC Inteligência Coletiva

ICMC Instituto de Ciências Matemáticas e de Computação

LRM Laboratório de Robótica Móvel

mTSP *multiple Traveling Salesmen Problem*

MCBGA *Modified Chu-Beasley Genetic Algorithm*

PSO *Particle Swarm Optimization*

RTE Regra de Transição de Estados

STI Sistemas de Transportes Inteligentes

TSP *Traveling Salesmen Problem*

USP Universidade de São Paulo

VRP *Vehicle Routing Problem*

VANT Veículo Aéreo Não Tripulado

Sumário

Capítulo 1

Introdução

Veículos autônomos é um tópico importante para o campo de Sistemas de Transportes Inteligentes (STI) e suas aplicações em cidades inteligentes, como a melhoria do fluxo de tráfego, gerenciamento automatizado de interseções, e redução de acidentes (BAGLOEE et al., 2016) (ELLIOTT; KEEN; MIAO, 2019). Embora o principal objetivo dos veículos autônomos em ambiente urbano seja o transporte de pessoas, outras tarefas podem ser automatizadas, como a entrega de mercadorias (TOOR; WARREN, 2017). Por exemplo, empresas podem entregar seus produtos de forma autônoma aos clientes em vários locais, 24 horas por dia, 7 dias por semana. Além disso, para aumentar a eficiência das entregas, estratégias de roteamento de veículos podem ser aplicadas para escalonar visitas nos locais de entrega (GUNAWAN; SUSYANTO; BAHAR, 2019).

O problema de escalonar visitas para um único veículo pode ser visto como uma instância do problema do caixeiro viajante (do inglês, *Traveling Salesmen Problem* (TSP)), que é um problema de otimização combinatória (GUNAWAN; SUSYANTO; BAHAR, 2019). Problemas de otimização combinatória são problemas que tentam encontrar uma solução ideal a partir de um conjunto finito de soluções (PARDALOS; DU; GRAHAM, 2013). No TSP, dado um conjunto de n cidades e o custo de viajar (ex: distância) entre cada possível par de cidades, o ojetivo é encontrar a melhor maneira possível de visitar todas elas, o que minimiza o custo final de viagem (DAVENDRA, 2010). Enquanto o TSP é restrito a um único caixeiro, o problema de múltiplos caixeiros viajantes, (do inglês, *multiple Traveling Salesmen Problem* (mTSP)) generaliza o problema para vários caixeiros, o que é comum em aplicações do mundo real, como roteamento de ônibus escolar (BAYKASOĞLU; ÖZBEL, 2016) (ELLEGOOD et al., 2020), gerenciamento de coleta e entrega (GUNAWAN; SUSYANTO; BAHAR, 2019), roteamento de veículos (BRAEKERS; RAMAEKERS; NIEUWENHUYSE, 2016) (MASUTTI; CASTRO, 2018) (DHEIN; NETO; ARAÚJO, 2018) (GUNAWAN; SUSYANTO; BAHAR, 2019), entre outros. Assim, o mTSP consiste em encontrar rotas para m caixeiros, que iniciam e terminam suas rotas em um único ponto, conhecido como depósito (BEKTAS, 2006). Além disso, cada cidade deve ser visitada exatamente uma vez por algum caixeiro, e a distância total percorrida para visitar todas

as cidades deve ser minimizada (BEKTAS, 2006).

No entanto, se for considerado apenas a minimização da distância total como objetivo de otimização, as distâncias percorridas individualmente pelos caixeiros podem tornar-se desequilibradas (ALVES; LOPES, 2015). Por exemplo, quando um caixeiro percorre longas distâncias, o custo de usar esse mesmo caixeiro para visitar todas as cidades daquela região é menor do que o custo de enviar outro caixeiro para visitar algumas delas. Como resultado, alguns caixeiros podem viajar muito, enquanto outros ficam ao redor do depósito. Portanto, para alguns problemas, também é fundamental equilibrar as rotas dos caixeiros (ALVES, 2017).

O TSP pertence à classe de problemas NP-completos [1], uma vez que o tempo aumenta exponencialmente para cada aumento em n. Mas o *mTSP* é ainda mais complexo, pois requer a avaliação e classificação de nós na rota de cada caixeiro, o que torna-o NP-difícil (KIRáLY; ABONYI, 2010). Neste sentido, algoritmos bio-inspirados, como Enxame de Partículas (*Particle Swarm Optimization* (PSO)), Colônia de Formigas (*Ant Colony Optimization* (ACO)) e Algoritmo Genético (*Genetic Algorithm* (GA)), são frequentemente usados para resolver esses problemas (LATAH, 2016) (XU et al., 2018) (GULCU; ORNEK, 2019) (HU et al., 2020).

Assim, o presente trabalho de mestrado apresenta o desenvolvimento de uma abordagem híbrida como uma solução para mTSP com aplicação em um sistema centralizado para escalonamento de rotas para uma frota de carros. Neste caso, cidades são locais de entrega a serem visitados e caixeiros são carros. O algoritmo *K-means* é usado como um pré-processamento para geração da população inicial com bons indivíduos para os algoritmos bio-inspirados, GA e ACS. Desse modo, são propostas duas soluções híbridas, K-means-GA e K-means-ACS. O sistema usa os algoritmos desenvolvidos para gerar soluções que minimizam a distância total viajada pela frota de veículos e também o balanceamento das rotas dos veículos. Experimentos foram conduzidos de modo a comparar os algorimos híbridos com os algoritmos PSO, ACS e o GA mono-objetivo proposto em (ALVES; LOPES, 2015). Ademais, um experimento simulado foi realizado usando modelos 3D[2] de veículos autônomos em uma cidade virtual e um experimento real na Universidade de São Paulo com 2 carros, sendo um autônomo e outro de direção manual. Os resultados apontam que à medida que o número de carros e locais de visita aumentam, as abordagens híbridas superam suas versões com implementação tradicional. Além disso, foi observado que o K-means-GA consegue atingir a convergência com um número menor de iterações se comparado ao GA.

[1] Complexidade de um problema computacional. <https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/NPcompleto.html>.

[2] Computação gráfica tridimensional.

1.1 Motivação

Com o desenvolvimento de pesquisas em veículos autônomos, sistemas de transportes inteligentes e cidades inteligentes, várias atividades hoje executadas por humanos poderão ser automatizadas (BAGLOEE et al., 2016). Apesar do foco em carros autônomos ser o transporte de pessoas, outras tarefas poderão ser realizadas, como entrega de mercadorias (TOOR; WARREN, 2017).

Recentemente, as empresas Ford e Domino's Pizza se uniram para testar carros de entrega de pizza autônomos em Michigan-EUA, como parte de um esforço para entender melhor como os clientes respondem e interagem com veículos autônomos (TOOR; WARREN, 2017).

Figura 1 – Carro Autônomo da Ford para entrega de pizzas Domino's (TOOR; WARREN, 2017).

Uma das vantagens de se eliminar o fator humano neste cenário é poder realizar entregas todos os dias a qualquer hora, sem se preocupar se o motorista está cansado ou precisa fazer paradas durante a jornada.

Com a realização de testes como esse, é possível imaginar que eventualmente existirão frotas de veículos autônomos fazendo entregas em diversas cidades em um futuro próximo. Com isso, surge a dificuldade de se coordenar todas essas entregas em diferentes locais utilizando um conjunto de veículos autônomos, ou seja, dado uma frota de veículos autônomos que se encontram em uma empresa de entrega e vários pedidos de clientes espalhados em diferentes pontos de uma cidade, como organizar as entregas de modo eficiente considerando que um veículo pode sair para entregar vários pedidos?

Um dos critérios seria tentar reduzir os custos da empresa de entrega gerando rotas para os veículos de forma que ao final tem-se a menor distância viajada pela frota. Contudo, esta minimização pode levar alguns veículos a viajar mais que outros, dependendo da disposição dos locais de entrega na cidade e do número de veículos disponíveis. Sendo assim, clientes que tem suas mercadorias transportadas por veículos que viajam longas distâncias podem demorar a receber suas mercadorias. Em termos práticos, por exemplo

no caso da entrega de pizza, um cliente ficaria insatisfeito ao receber sua pizza fria, além de passar algum tempo esperando a entrega.

O problema neste cenário é que a minimização da distância total impacta no equilíbrio das rotas e vice-versa, ou seja, tem-se dois critérios de otimização opostos, o que é um desáfio para algoritmos de otimização. Contudo, esses critérios de otimização não são conflitantes, assim a aplicação de algoritmos bio-inspirados da classe mono-objetivo são mais adequados que os da classe multi-objetivo (HASHIMOTO, 2004).

O escalonamento de visitas para mais de um veículo pode ser visto como uma instância mTSP (MASUTTI; CASTRO, 2018) (XU et al., 2018) (DHEIN; NETO; ARAÚJO, 2018), e sendo o mTSP um problema da classe NP-difícil, a aplicação de algoritmos bio-inspirados é adequada na busca por soluções aproximadas (KIRáLY; ABONYI, 2010). O escalonamento da ordem de visitas para robôs com a aplicação de dois algoritmos genéticos é apresentado em (ALVES, 2017). No trabalho, também é apresentada uma função de aptidão que realiza a minimização da distância total viajada pelos robôs e equilíbrio dessas distâncias. Já nos trabalhos de (LATAH, 2016)(XU et al., 2018)(YANG; SZETO, 2019) (MA et al., 2019), o mTSP é resolvido com a aplicação de um método de transformação, o qual objetiva simplificar o mTSP em vários TSP com a dinâmica de dividir para conquistar, e para cada TSP aplicar um algoritmo de otimização.

Sendo assim, a motivação para a presente pesquisa de mestrado estende os trabalhos (ALVES; LOPES, 2015) e (ALVES, 2017), e propõe o uso do *K-means*, como meio de pré-processamento para geração da população inicial dos algoritmos GA e ACS. Sendo o foco na modelagem e construção de uma população inicial mTSP e execução única dos algorimos bio-inspirados para o problema. Ressalta-se também a aplicação em um sistema centralizado para escalonamento de rotas para veículos, o qual utiliza os algoritmos desenvolvidos para escalonar a ordem de visita para cada veículo.

1.2 Objetivos

Os objetivos gerais e específicos deste trabalho são:

- Geral
  - Propor uma abordagem *mTSP* para coordenação de uma frota de veículos autônomos de entrega em um sistema centralizado.
- Específicos
  - Desenvolver e comparar os algoritmos bio-inspirados, PSO, ACS e GA (ALVES; LOPES, 2015), para resolver o *mTSP*, com aplicação em um sistema centralizado de coordenação de veículos;

- Investigar a aplicação do algoritmo *k-means* como gerador da população inicial e meio de acelaração da convergência para os algoritmos bio-inspirados, de modo a desenvolver abordagens híbridas;
- Comparar as abordagens híbridas com os algoritmos com implementações tradicionais dos algoritmos bio-inspirados;
- Conduzir um estudo usando veículos autônomos com o sistema de escalonamento em um cenário real.

1.3 Hipótese

- ❑ Como coordenar veículos autônomos de entrega reduzindo a distância total viajada pela frota e equilibrando as rotas individuais simultaneamente?
- ❑ É possivel acelerar a convergência de algoritmos bio-inspirados mantendo ou obtendo melhores resultados?
- ❑ Com o aumento de locais de entrega e de veículos, os algortimos bio-inspirados apresentam o mesmo desempenho em relação a minimização da função de aptidão?

1.4 Contribuições

- ❑ Algoritmo *k-means* como um gerador de população;
- ❑ Aceleração da convergência de algoritmos bio-inspirados para o mTSP;
- ❑ Comparação entre as implementações tradicionais dos algoritmos bio-inspirados GA, ACS e PSO e as abordagens híbridas desenvolvidas k-means-GA e k-means-ACS;
- ❑ Proposta de um sistema centralizado para escalonar rotas para entrega;
- ❑ Condução de um estudo de caso em um cenário real.

1.5 Organização da Dissertação

Esta dissertação esta organizada em seis capítulos, como mostrado a seguir:

- ❑ Capítulo 2: apresenta os conceitos teóricos dos componentes de um sistema inteligente para escalonamento de rotas para veículos autônomos: algoritmo de planejamento A*, algoritmos bio-inspirados e *K-means*. Além disso, apresenta conceitos sobre testes estatísticos e o teste utilizado para avaliação dos experimentos;
- ❑ Capítulo 3: apresenta trabalhos correlatos a proposta da presente dissertação;

- ❏ Capítulo 4: apresenta a abordagem que trabalha para minimização da distância de uma frota e balanceamento das distâncias viajadas por cada carro;
- ❏ Capítulo 5: apresenta os resultados realizados para a avaliação dos algoritmos desenvolvidos e compara com o GA(ALVES; LOPES, 2015) em quatro experimentos;
- ❏ Capítulo 6: apresenta a conclusão desta dissertação com as considerações finais e propostas para possíveis trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Sabendo que veículos autônomos é um tópico importante para o campo de STI e o uso de técnicas computacionais podem contribuir para a tarefa de coordenação de uma frota de tais veículos. Neste capítulo são apresentados os componentes teóricos de um sistema inteligente para escalonamento de rotas para veículos.

O capítulo está organizado da seguinte maneira: a Seção 2.1 contextualiza o campo de veículos autônomos e apresenta dois veículos desenvolvidos pelo Laboratório de Robótica Móvel (LRM-ICMC/USP) parceiro neste trabalho; a Seção 2.2 descreve as principais técnicas para coordenação de agentes móveis; a Seção 2.3 apresenta os problemas TSP e mTSP e descreve os principais métodos que vem sendo abordados para resolução desses problemas; a Seção 2.4 apresenta a teoria sobre o problema de planejamento de caminhos e o algoritmo A*; a Seção 2.5 apresenta os principais algoritmos de otimização bio-inpirados usados para escalonamento de rotas em abordagens TSP e mTSP; a Seção 2.6 apresenta o algoritmo de Aprendizado de Máquina (AM) não supervisionado *k-means*; e por fim, a Seção 2.7 apresenta a teoria sobre testes estatísticos.

2.1 Veículos Autônomos

Os veículos autônomos são previstos para diminuir o congestionamento nas estradas, melhorar a segurança eliminando erros humanos e liberando os motoristas da responsabilidade da direção do veículo, permitindo maior produtividade e /ou tempo para descanso, juntamente com uma infinidade de outros benefícios (PENDLETON et al., 2017).

Os carros autônomos já são uma realidade no contexto atual. Montadoras como Volvo [1], BMW [2], Mercedes-Benz [3], Ford [4] e Tesla [5] já testam veículos autônomos com

[1] Fabricante de veículos comerciais. Fonte: volvocars.com
[2] Empresa alemã, fabricante de automóveis e motocicletas. Fonte: bmw.com.br
[3] Marca alemã de automóveis. Fonte: mercedes-benz.com.br
[4] Fabricante de automóveis multinacional sediada em Dearborn, Michigan. Fonte: ford.com.br
[5] Empresa automotiva norte americana, que produz e vende automóveis elétricos de alto desempenho. Fonte:tesla.com

o objetivo de oferecer uma tecnologia autônoma independente de um condutor humano (FUSSY; OLIVEIRA, 2015) e (DINO, 2019). Com as inúmeras possibilidades de execução e automatização de serviços com uso desses veículos inteligentes, empresas de tecnologia como Google, Uber e outras realizam pesquisas e testes com propósito de adotar os carros autônomos para execução de alguns de seus serviços (WADE, 2018).

Pesquisas com o uso de Inteligência Artificial (IA) e suas variáveis técnicas vem sendo conduzidas com o propósito de melhorar a navegação e detecção de objetos em vias urbanas para carros autônomos (FERNÁNDEZ et al., 2013) (HATA; WOLF, 2015) (RIDEL et al., 2017). Esse tipo de tarefa se encontra no campo da robótica móvel, sendo a navegação autônoma um dos principais problemas da área (SOUZA, 2014). Para um agente autônomo compreender o ambiente é essencial o uso de sensores e atuadores. Como sensores compreende-se: câmeras de vídeo, radares, GPS, laser e unidades de medida inercial, e como atuadores: pernas, garras e rodas (SOUZA, 2014) (FERNANDES et al., 2014). A Figura 2 apresenta um agente autônomo interagindo com o seu ambiente por meio de sensores, os quais permitem a percepção do ambiente e realização de ações através dos seus atuadores.

Figura 2 – Agente móvel autônomo interagindo com o ambiente através de sensores e atuadores. Fonte: (WOLF et al., 2009).

Para uma navegação autônoma alguns requisitos são necessários como computadores com capacidade de interpretação dos dados de percepções e atuadores que em um veículo autônomo permite a direção, aceleração e frenagem no ambiente em que ele se encontra. Desse modo, um sistema de software de um veículo autônomo pode ser categorizado em três categorias, percepção, planejamento e controle, como apresentado na Figura 3.

A Figura 3 mostra uma visão geral dos sistemas de veículos autônomos, apresentando os componentes de *hardware*: sensores, comunicação V2V (*vehicle to vehicle*) e atuadores. Como componentes de *software* são apresentadas três categorias que possibilitam a coleta de dados e tomada de decisões do sistema: percepção, planejamento e controle.

1. **Percepção:** compreende a percepção do ambiente, capacidade de um sistema autônomo reunir e extrair conhecimentos do ambiente, e a localização, habilidade do

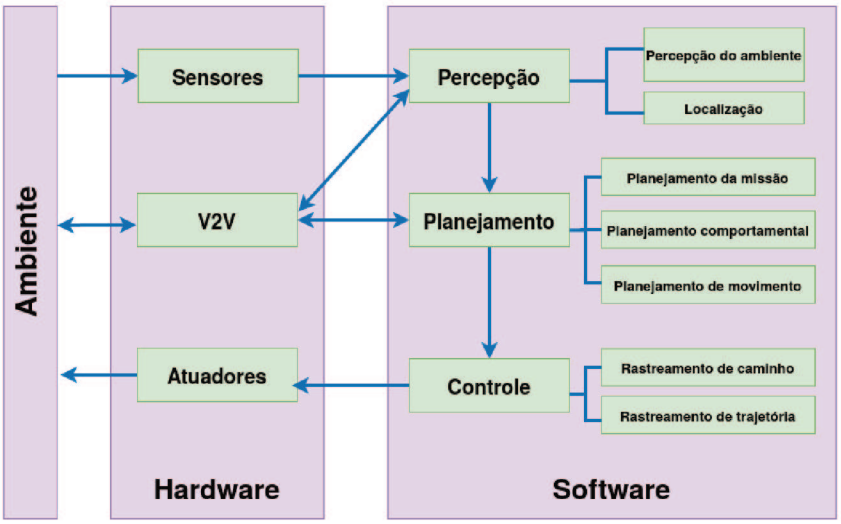


Figura 3 – Uma visão geral típica do sistema de veículos autônomos. Imagem adaptada de (PENDLETON et al., 2017).

agente autônomo determinar sua posição com relação ao ambiente.

2. **Planejamento:** processo de tomada de decisões para atingir um objetivo, tal como movimentar o veículo de um local de início para um local de objetivo, evitando obstáculos e otimizando as heurísticas desenvolvidas.

3. **Controle:** capacidade do agente autônomo executar as ações planejadas que foram geradas pelos outros processos.

Em (FERNANDES et al., 2012), é apresentado uma plataforma e arquitetura de um veículo inteligente, descrevendo o sistema de controle e módulos, os quais, permitem a navegação autônoma. Essa plataforma foi nomeada como CaRINA I (Figura 4), e pode funcionar em ambientes urbanos e estradas, sendo as estradas mais indicadas pelos autores do trabalho.

Figura 4 – Plataforma robótica veicular CaRINA I. Fonte: (FERNANDES et al., 2012).

A plataforma CaRINA I inclui bases e estruturas de suporte construídas especialmente para a instalação de *lasers*, câmeras e outros sensores (FERNANDES et al., 2012).

Posteriormente em (FERNANDES et al., 2014) é apresentado a arquitetura da plataforma CaRINA II, um (Fiat Palio *Adventure*) modificado para navegação autônoma (Figura 5). O trabalho também apresenta o veículo como o primeiro veículo comercial da América Latina capaz de executar navegação autônoma nas ruas. Umas das principais diferenças entre o CaRINA I e o CaRINA II é o mecanismo usado para alternar entre direção manual e direção controlada pelo computador. Enquanto o CaRINA I usa um dispositivo mecânico, o sistema CaRINA II possui um sistema eletromecânico controlado por um dispositivo baseado em um sistema de acoplamento magnético (FERNANDES et al., 2014).

Figura 5 – Plataforma robótica veicular CaRINA II. Fonte: (LRM, 2011).

As Plataformas CaRINA I e CaRINA II possuem uma arquitetura de sistema, na qual os módulos específicos para cada atuador do motor do veículo foram desenvolvidos e são responsáveis pela conversão dos comandos abstratos (pontos de ajuste) em seus valores equivalentes para cada plataforma, produzindo a ação desejada (FERNANDES et al., 2014).

A Figura 6 apresenta a arquitetura do sistema CaRINA. Os módulos em verde compreendem a tomada de decisões, tendo como subordinados os módulos em laranja; Os módulos em vermelho compreendem a percepção do ambiente. Eles são responsáveis por adquirir dados sensoriais e fornecê-los para uma representação que permita o reconhecimento de forma simples, rápida e confiável; Em azul são apresentados os sensores que possibilitam a coleta de dados do ambiente de modo a fornecer essas informações para os módulos em vermelho; Em roxo são apresentados os atuadores que possibilitam a execução de ações com base nas decisões fornecidas pelos módulos em laranja; Em cinza são apresentados os módulos referentes ao subsistema de segurança e *backup*. Eles são é responsáveis por manter registros de decisões e operações realizadas pelo sistema, permitindo melhores análises após o processamento ou até a criação de dados para simulação; Por fim, em amarelo são apresentados os barramentos de comunicação e segurança com outros veículos.

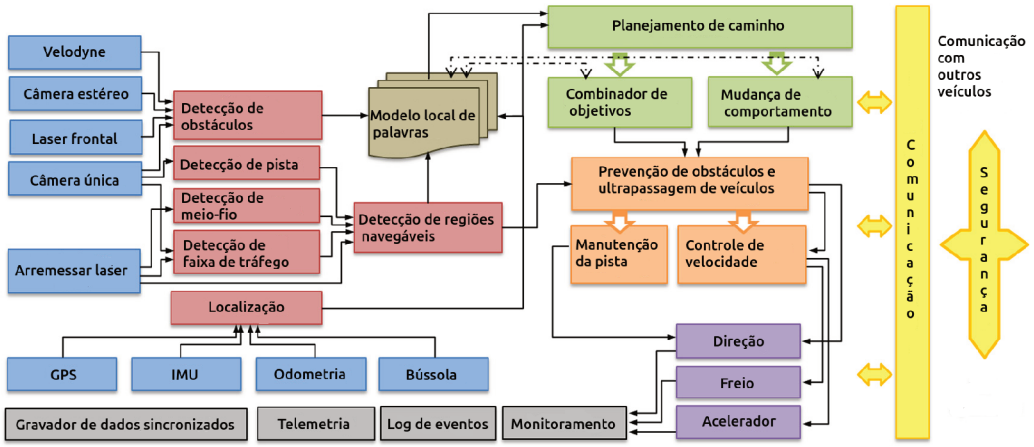


Figura 6 – Arquitetura do sistema CaRINA. Imagem adaptada de (FERNANDES et al., 2014).

2.2 Coordenação para Agentes Autônomos

Veículos autônomos são conceitualmente agentes inteligentes que percebem e agem sobre o ambiente para executar algum tipo de tarefa (RUSSELL; NORVIG, 2013). Neste sentido, um time de agentes que trabalha em paralelo tem potencial para finalizar um dada tarefa mais rápida que um único agente (ALVES, 2017). Além disso, times de agentes autônomos introduzem redundância no sistema que o torna mais robusto à falhas. Quando um time de agentes resolve uma tarefa em conjunto, agentes que possivelmente sofram falhas podem ser substituídos por outros, sendo umas das vantagens de se considerar mais de um agente para uma mesma tarefa.

De modo semelhante a sociedade humana, existem tipos de comportamento em ambientes com agentes autônomos, os quais podem ser competitivo ou cooperativo. Competitivo apresenta situações nas quais os agentes autônomos competem uns contra os outros para melhor atender seus objetivos particulares. Já o comportamento cooperativo está relacionado a situações onde agentes autônomos precisam atuar juntos a fim de completar uma tarefa enquanto otimizam uma função de utilidade do sistema.

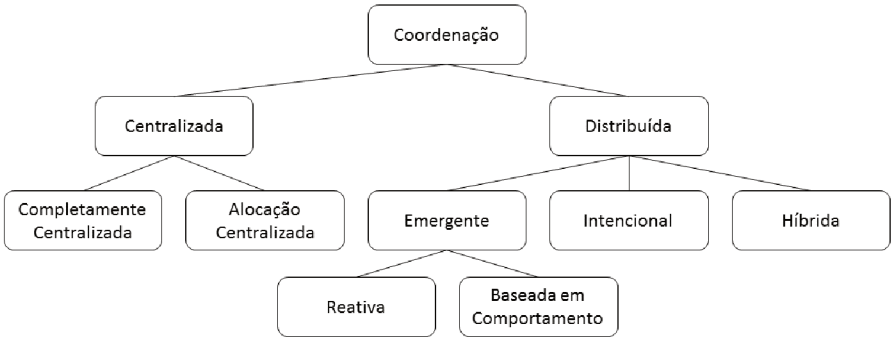


Figura 7 – Taxonomia das abordagens de coordenação para agentes autônomos. Imagem adaptada de (ALVES, 2017).

Além do tipo de comportamento, competitivo ou cooperativo, existe uma taxonomia geral com relação a coordenação de múltiplos agentes autônomos conforme ilustrado na Figura 7. Na abordagem centralizada, existe um controle central que possui informações gerais do ambiente e de todos os agentes autônomos. Tal controle central pode ser um computador ou até mesmo algum dos próprios agentes autônomos com capacidade de comunicação. Uma vantagem dessa abordagem é que se tem a visão geral do ambiente no mesmo local, onde serão produzidos planos otimizados que atendem aos requisitos de todos os agentes autônomos.

De modo contrário, abordagens distribuídas não possuem essa visão geral centralizada, o que torna os processos de coordenação e de tomada de decisão mais complexos e sujeitos a erros. A Tabela 1 mostra vantagens e desvantagens de cada abordagem apresentada nesta taxonomia.

Abordagem	Descrição	Vantagens	Desvantagens
Completamente Centralizada	único agente planeja para o time inteiro;	potencial para ser ótimo; coordenação implícita;	geralmente intratável computacionalmente; único ponto de falha;resposta lenta à mudanças;
Alocação Centralizada	único agente aloca tarefas para membros do time; membros do time completam tarefas individuais;	execução distribuída; alocação pode ser ótima;	computacionalmente ainda é cara; ainda possui um ponto central de falha;
Reativa	agentes possuem um pequeno laço de perceber e agir;	extremamente rápida e simples;	não consegue lidar com tarefas complexas;
Baseada em Comportamento	usa informações sobre o estado atual para tomada de ações;	rápida e simples; agentes podem contribuir com múltiplas tarefas;	mais cara que a reativa; agentes ainda não conseguem planejar;
Intencional	comunicação com intenção de coordenação;	facilita escalonamento e planejamento;	lenta em situações de tempo crítico; muito dependente de comunicação;
Híbrida	maior intenção de coordenação;	permite uma melhor distribuição e planejamento de recursos; possui uma pequena coordenação;	não consegue realizar iterações complexas;

Tabela 1 – Comparativo das abordagens Centralizadas vs. Distribuídas. Tabela adaptada de (ALVES, 2017).

Na abordagem completamente centralizada, um único agente planeja a execução das tarefas para o time inteiro, sendo a execução também de modo centralizado. Nesta abordagem, os demais agentes não têm total autonomia durante a execução. Já na alocação centralizada um único agente determina a execução das tarefas para o time inteiro, porem a execução é distribuída, assim cada agente tem autonomia para tomar decisões durante a execução.

A abordagem que melhor se adéqua ao problema tratado nesta pesquisa é a alocação centralizada, na qual um sistema centralizado escalonará a ordem das entregas de cada veículo autônomo. Uma vez que essa informação chega aos veículos, cada um tem autonomia de tomar decisões durante a execução das trajetórias como desviar de obstáculos, parar no semáforo, etc.

2.3 Problema de Múltiplos Caixeiros Viajantes

Nesta dissertação é abordado o problema de escalonar visitas de uma frota de veículos autônomos até locais de entrega usando a abordagem de alocação centralizada. Para isso foi adotada uma variante do problema dos caixeiros viajantes (do inglês *Traveling Salesman Problem* - TSP), para mais de um caixeiro. Esse tipo de variante é chamado *multiple Traveling Salesman Problem* (mTSP), o qual generaliza o problema para mais de um caixeiro.

O TSP tradicional é um problema que tenta definir a menor rota para percorrer um conjunto de cidades (visitando uma única vez cada uma delas), retornando à cidade ponto de partida (DAVENDRA, 2010). Ele é um problema de otimização motivado pela necessidade dos caixeiros viajantes realizarem entregas em diversos locais (as cidades) percorrendo o menor caminho possível, reduzindo o tempo necessário para a viagem e os possíveis custos com transporte e combustível (DAVENDRA, 2010).

A representação do TSP é adequada para modelar problemas práticos em diversas áreas como jogos, logística e robótica (ALVES, 2017). Em tais problemas, uma cidade poderia ser um ponto específico em que um agente deve deslocar-se, e a distância poderia ser o tempo ou o custo de deslocamento do agente para esse ponto. Enquanto o TSP é limitado a um único agente, o mTSP estende o problema para vários agentes, o que é mais comum em aplicações do mundo real, como roteamento de ônibus escolar (BAYKASOĞLU; ÖZBEL, 2016) (ELLEGOOD et al., 2020), gerenciamento de coleta e entrega (GUNAWAN; SUSYANTO; BAHAR, 2019) e roteamento de veículos (MASUTTI; CASTRO, 2018) (DHEIN; NETO; ARAÚJO, 2018).

No mTSP, é apresentado um conjunto de cidades e todas elas devem ser visitadas somente uma vez por algum caixeiro, os quais iniciam e terminam suas viagens em um local comum de origem chamado depósito. O número de cidades é representado por n e o número de caixeiros por m. O objetivo é encontrar rotas para os m caixeiros, de modo que o custo total de viagem (o custo de visitar todas as cidades) seja minimizado. A medida de custo pode ser definida em termos de distância, tempo, etc. Algumas variações possíveis do problema são apresentadas de acordo com (ALVES, 2017):

- Vários depósitos: caso o problema tenha vários depósitos com um número de caixeiros localizados em cada um, um caixeiro poderá retornar a qualquer depósito com as limitação de que o número inicial de caixeiros em cada depósito permaneça o mesmo após todas as viagens.

- Número de caixeiros: podendo ser um número fixo ou uma variável limitada.

- Carga fixa: se o número de caixeiros for uma variável limitada, geralmente o uso de cada caixeiro na solução terá um custo fixo associado. Nesse caso, a minimização dessa variável limitada pode estar envolvida na otimização.

- Janelas de tempo: usada quando certas cidades devem ser visitadas em determinados períodos.

- Outras restrições: restrições adicionais baseiam-se na distância máxima ou mínima, ou na duração da viagem que um caixeiro percorre ou em outras restrições especiais.

Alguns trabalhos apresentam a formulação do mTSP onde, $G(V, A)$ é um grafo não direcional completo com um conjunto de vértices $V = \{0, 1, 2, ..., n\}$ e um conjunto de arestas $A = \{(i, j) : i, j \in V, i \neq j\}$. Caso o grafo não seja completo, a falta de alguma aresta é trocada por uma de tamanho infinito. Algumas variáveis também são introduzidas:

- n = número de cidades a serem visitadas;

- m = número de caixeiros.

- C = matriz de custo, onde:

$$\begin{gathered} C_{ij} = C_{ji} \\ C_{ij} + C_{jk} \geq C_{ik}, \forall ijk = 1, 2, 3, ..., n \end{gathered} \tag{1}$$

$$X_{ij} = \begin{cases} 1 \text{ Se o caixeiro viaja diretamente de } i \text{ para } j \\ 0 \text{ caso contrário} \end{cases} \tag{2}$$

O mTSP pode ser resolvido através de algumas principais estratégias. Essas estratégias são conhecidas como métodos exatos, heurísticos e de transformação.

- **Métodos Exatos:** Os métodos exatos tem como característica a garantia da obtenção da solução ótima do problema. Contudo, essa garantia aplica-se apenas em problemas com pequenas e médias instâncias, pois o tempo de execução tende a aumentar com relação ao tamanho da instância do problema, restringindo o uso prático destes algoritmos (STEFANELLO, 2011).

- **Métodos Heurísticos:** Diferente dos métodos exatos, os métodos heurísticos ou aproximados buscam encontrar boas aproximações da solução ótima em um tempo de processamento aceitável (DESALE et al., 2015). As heuríticas procuram resolver os problemas de maneira racional, explorando a estrutura do problema para buscar boas soluções, mas sem garantir a otimalidade (QUEIROZ, 2011).

- **Transformações:** Técnicas de transformação tem como objetivo transformar o problema na versão tradicional do TSP e assim utilizar outras soluções já conhecidas.

A Tabela 2 apresenta algoritmos pertencentes a essas estratégias que se propõem a resolver o mTSP.

Estratégia	Algoritmo
Métodos Exatos	Integer linear programming formulatios (KULKARNI; BHAVE, 1985) (KARA; BEKTAS, 2006)
	Cutting plane (LAPORTE; NOBERT, 1980)
	Branch and Bound (ALI; KENNINGTON, 1986) (GROMICHO; PAIXAO; BRONCO, 1992)
	Lagrangean relaxation + branch and bound (GAVISH; SRIKANTH, 1986)
Métodos Heurísticos	Heurísticas Simples (RUSSELL, 1977) (POTVIN; LAPALME; ROUSSEAU, 1989)
	Evolutionary algorithm (FOGEL, 1990)
	Simulated annealing (SONG; LEE; LEE, 2003)
	Tabu search (RYAN et al., 1998)
	Genetic algorithms (ZHANG; GRUVER; SMITH, 1999) (YU et al., 2002) (TANG; LIU; YANG, 2000)
	Neural networks (MODARES; SOMHOM; ENKAWA, 1999) (TORKI; SOMHON; ENKAWA, 1997) (VAKHUTINSKY; GOLDEN, 1994)
Transformações	mTSP assimétrico para TSP assimétrico (BELLMORE; BELLMORE, 1974)
	Symmetric mtSP to symmetric TSP (HONG; PADBERG, 1977) (RAO, 1980) (JONKER; VOLGENANT, 1988)
	Multidepot mTSP to TSP (LAPORTE; NOBERT; TAILLEFER, 1988) (GUO-XING, 1995)

Tabela 2 – Estratégias e algoritmos para mTSP. Fonte: (BEKTAS, 2006).

Existem vários problemas reais baseados no TSP tradicional, como o problema de roteamento de veículos (*Vehicle Routing Problem* (VRP)) (BRAEKERS; RAMAEKERS; NIEUWENHUYSE, 2016). O VRP estende o TSP de uma forma que exista um conjunto de clientes ou paradas (locais de entrega), e eles têm que ser visitados por um veículo, o qual deve iniciar e terminar sua viagem no depósito. Embora os métodos exatos resolvam pequenas instâncias desses problemas com bastante eficiência, ainda existem dificuldades para instâncias maiores. Por outro lado, existem várias abordagens meta-heurísticas capazes de encontrar soluções razoáveis em menos tempo, incluindo: otimização de enxame de partículas, colônia de formigas e algoritmo genético (KARAKATIč; PODGORELEC, 2015).

2.4 Planejamento de Caminhos

Planejamento de caminhos (do inglês, *path planning*) é um tópico relevante de pesquisa em aŕeas como robótica móvel e suas aplicações em cidades inteligentes (LAUMOND et al., 1998) (HU; YANG, 2004) (AL-TURJMAN, 2016). Segundo (ZHANG; ZHAO, 2014):

> planejamento de caminhos refere-se a tarefa de planejar uma trajetória em um ambiente com obstáculos estáticos e dinâmicos, em que o robô pode encontrar um caminho do começo ao fim para atender a certos critérios de avaliação,

enquanto o mesmo deve evitar obstáculos com segurança e confiabilidade nas viagens.

Um problema clássico na literatura é citado como *Piano Mover's Problem* (Problema do Movimento do Piano) (DAVENPORT, 1986). Tal problema consiste em determinar a trajetória do piano de um cômodo de uma casa para outro sem acertar nenhum obstáculo. Problemas como esse são comuns na robótica móvel, por exemplo, movimentação de um robô de entrega de documentos em um departamento público.

Um problema de planejamento de caminhos também pode ser definido como um problema de busca, onde podemos fazer uma busca através do espaço de estados, partindo do estado inicial à procura de um estado objetivo (RUSSELL; NORVIG, 2013). Em (RUSSELL; NORVIG, 2013), problemas de busca são definidos com a modelagem de estados, estado inicial, ações, modelo de transição, teste de objetivo e custo de caminho:

- **Estados:** parte do ambiente em que um agente se encontra após a execução de uma ação;
- **Estado inicial:** estado em que o agente se encontra no início da execução da busca;
- **Ações:** conjunto de comportamentos que um agente pode desempenhar para se deslocar de um estado para os demais;
- **Modelo de transição:** determina o estado resultante ao realizar uma determinada ação a partir do estado atual;
- **Teste objetivo:** avaliação de um estado frente ao estado objetivo;
- **Custo de caminho:** número de passos do caminho.

Dois algoritmos clássicos para resolução de problemas de planejamento são: busca em espaço de estados para a frente (progressão) e busca para trás (regressão) de estados relevantes (RUSSELL; NORVIG, 2013). Esses algoritmos não são eficientes sem uma boa função heurística. Uma função heurítica $h(s)$ tem como papel estimar a distância de um estado s para o estado objetivo (RUSSELL; NORVIG, 2013). Um algoritmo tipicamente utilizado para a tarefa de planejamento de caminhos é o algoritmo A*, visto que ele apresenta condições para a definição de boas heurísticas (YAO et al., 2010) (DUCHOŇ et al., 2014) (LIU et al., 2017) e (GUNAWAN et al., 2019).

2.4.1 Algoritmo A*

O algoritmo A*, pronuncia-se A estrela, é um algoritmo de busca informada, o qual utiliza o conhecimento do problema, além da sua definição e pode encontrar soluções de forma mais eficiente que uma estratégia de busca sem informação (RUSSELL; NORVIG,

2013). O A* avalia os nós (estados) no espaço de busca através da combinação de $g(n)$, o custo para alcançar o nó, e $h(n)$, o custo para ir do nó ao objetivo.

$$f(n) = g(n) + h(n) \tag{3}$$

Se $g(n)$ é o custo do caminho saindo do estado inicial até o estado n e $h(n)$ é o custo estimado do caminho de menor custo de n até o objetivo, então $f(n)$ = custo estimado da solução de menor custo através de n.

Para que uma função heurística seja vista como ótima, duas condições devem ser respeitadas. A primeira condição é que $h(n)$ seja uma heurística admissível, em outros termos, nunca ultrapasse o custo de atingir o objetivo. A segunda condição é que ela seja consistente, essencial em aplicações do A*. Assim, $h(n)$ será consistente se, para cada nó n e para todo próximo n' de n gerado por uma ação a, o custo estimado de alcançar o objetivo de n não for maior do que o custo do passo de chegar a n' mais o custo estimado de alcançar o objetivo de n' (RUSSELL; NORVIG, 2013).

$$h(n) \leq c(n, a, n') + h(n') \tag{4}$$

O algoritmo A* tem as seguintes características: a versão de busca em árvore de A* é ótima se $h(n)$ for admissível, enquanto a versão de busca em grafos de A* é ótima se $h(n)$ for consistente (RUSSELL; NORVIG, 2013).

2.5 Algoritmos de Otimização Bio-Inspirados

Algoritmos de Otimização Bio-Inspirados (AOBI) são métodos meta-heurísticos frequentemente utilizados em pesquisas da Ciência da Computação e afins (BARBOSA, 2017). Tais algoritmos procuram compreender os padrões encontrados na natureza, de modo a aplicá-los no desenvolvimento de ferramentas computacionais e resolver problemas complexos (BARBOSA, 2017). Esses algoritmos fazem parte dos métodos de computação natural, ciência que utiliza a natureza como fonte de inspiração para desenvolvimento de técnicas computacionais na resolução de problemas complexos (YANG, 2010). A Figura 8 apresenta uma taxonomia geral com relação a computação natural.

A computação inspirada na natureza ou bio-inspirada compreende estratégias desenvolvidas inspiradas em algum recurso biológico ou natural (GOLDBERG, 1989). Como exemplos dessas estratégias têm-se a computação evolutiva (BACK; FOGEL; MICHALEWICZ, 2000), inteligência coletiva (BONABEAU; DORIGO; THERAULAZ, 1999), (KENNEDY; EBERHART; SHI, 2001), redes neurais artificiais (SIMON, 1999), sistemas imunológicos artificiais (DASGUPTA, 1999) e sistemas endócrinos artificiais (WEI; QIANG; GAO, 2006), (TIMMIS; NEAL; THORNILEY, 2009).

Já a simulação e emulação de fenômenos naturais abrange a aplicação de técnicas computacionais com o objetivo de resumir comportamentos naturais, padrões e processos

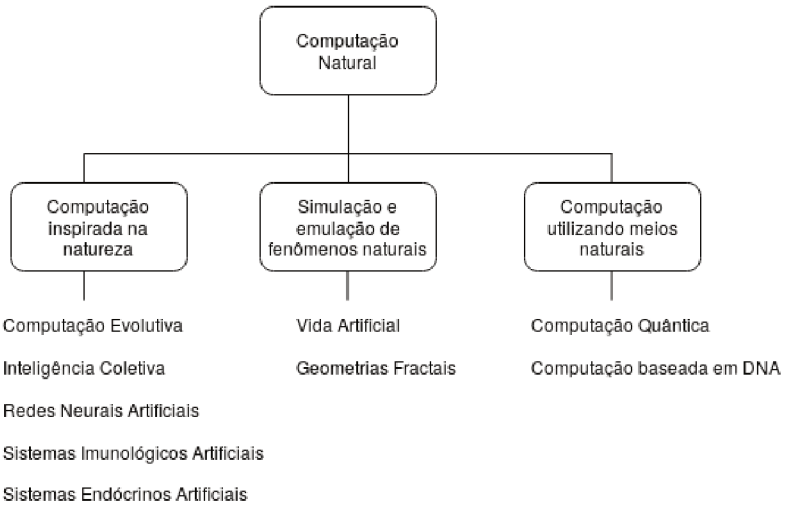


Figura 8 – Taxonomia dos tópicos de pesquisa em computação natural. Imagem adaptada de (PAPPA, 2005)

biológicos (CASTRO et al., 2004). Suas principais estratégias são vida artificial, organismos artificiais (ADAMI, 1998) e geometria fractal (MANDELBROT, 1983).

Por fim, a computação utilizando meios naturais refere-se ao um paradigma onde recursos naturais, como, cadeias de DNA e *bits* quânticos são usados como estruturas de dados no desenvolvimento de computadores naturais (PăUN; ROZENBERG; SALOMAA, 1998) (NIELSEN; CHUANG, 2002) (CASTRO et al., 2004).

Dentre as estratégias bio-inspiradas, enfatiza-se a Computação Evolutiva (CE) e a Inteligência Coletiva (IC), cujo os algoritmos foram tópicos de estudo para a condução desta dissertação.

Os algoritmos de CE, no geral, buscam soluções para problemas de otimização baseando-se em conceitos apresentados na teoria da evolução, escrita por Charles Darwin, compreendendo a existência de uma população de indivíduos que competem entre si, processos de modificação e reprodução, entre outros (POLI et al., 2008) (BÄCK; FOGEL; MICHALEWICZ, 1997).

A IC ou inteligência de enxames também é vista como um ramo da computação natural (SERAPIÃO, 2009). A abordagem é uma alternativa computacional do processo de extrair ideias da natureza, como comportamento de um bando de pássaros, cardume de peixes e cooperação das colônias de formigas, para desenvolver sistemas computacionais (SERAPIÃO, 2009), (DORIGO, 1992) e (KENNEDY; EBERHART, 1995).

2.5.1 Algoritmo Genético

Um dos métodos de CE mais usado são os Algoritmos Genéticos (do inglês, *Genetic Algorithm* - GAs), os quais foram fundamentados inicialmente com o trabalho (HOLLAND et al., 1992), e posteriormente aprimorados por (GOLDBERG, 1989). No geral os GAs

aplicam os princípios da seleção natural, como a sobrevivência do mais apto, na busca de boas soluções em espaços de busca complexos.

Na busca por boas soluções, um GA utiliza uma estrutura denominada cromossomo ou indivíduo, que caracterizam possíveis soluções no espaço de busca do problema. Além do cromossomo, um GA compreende alguns operadores genéticos como: seleção, reprodução ou cruzamento, mutação e reinserção (GOLDBERG, 1989). A Tabela 3 apresenta o significado da terminologia técnica da evolução natural aplicada aos GAs.

Evolução natural (termo)	Algoritmos Genétivos (significado)
Cromossomo	indivíduo
Gene	característica
Alelo	valor de uma característica
Locus	posição
Genótipo	estrutura codificada
Fenótipo	conjunto de parâmetros

Tabela 3 – Definição da terminologia da evolução netural aplicada aos GAs (GOLDBERG, 1989).

Embora a literatura apresente diferentes implementações de GAs, eles geralmente compartilham a seguinte estrutura (MITCHELL, 1997): O algoritmo opera por iterações também chamadas de gerações atualizando ativamente a população de indivíduos. Em cada iteração, todos os indivíduos da população são avaliados de acordo com a função de aptidão (*fitness function*). Uma nova população é então gerada pela seleção probabilística dos indivíduos mais aptos da população atual. Alguns desses indivíduos selecionados são levados adiante na próxima geração da população intacta. Outros são usados como base para a criação de novos indivíduos descendentes, aplicando operações genéticas como cruzamento e mutação. Uma estrutura básica de um GA é apresentada no Algoritmo 1.

Uma população contendo p indivíduos é incializada. Em cada iteração, uma população sucessora P_s é formada probabilisticamente selecionando os indivíduos correntes de acordo com seus valores de aptidão. Novos indivíduos são criados aplicando o operador de cruzamento a pares de indivíduos mais adequados e criando mutações na geração resultante. Esse processo é iterado até atingir uma condição de parada pré-estabelecida.

2.5.1.1 Representação dos Indivíduos

Um indivíduo ou cromossomo é uma parte essencial na implementação de um GA. A estrutura do cromossomo deve ser formulada seguindo as especificações do problema abordado. Existem diversas formas de representação para os cromossomos, como a representação binária, inteira e real (HOLLAND et al., 1992) (MITCHELL, 1997) (DAVIS, 1991). Na representação binária cromossomos são codificados por uma sequência de dígitos binários (0 e 1), sendo assim, um gene pode ser representado por 0 ou 1 (HOLLAND et al., 1992). Na representação em ponto flutuante cada gene de um cromossomo é codificado

Algoritmo 1 Estrutura básica do GA (MITCHELL, 1997)

Entrada: o tamanho da população p, o número de gerações T, taxa de cruzamento r, taxa de mutação m.

Saída: O melhor indivíduo.

1: $P \leftarrow$ Gerar p indivíduos aleatórios (Passo 1)
2: **for** cada h em P **do**
3: calcular o $Fitness(h)$ (Passo 2)
4: **end for**
5: $t \leftarrow 0$
6: **while** $t < T$ **do**
7: Selecione $(1-r)p$ indivíduos de P para adicionar a P_s. A probabilidade $Pr(h_i)$ de selecionar um indivíduo h_i de P é dada por: (Passo 3)

$$Pr(h_i) \leftarrow Fitness(h_i)$$

8: Selecione $\frac{r.p}{2}$ pares de indivíduos de P, de acordo com $Pr(h_i)$ obtidos acima. Para cada par, $(h1, h2)$, produzir dois filhos aplicando o operador de cruzamento. Adicionar todos os filhos a P_s. (Passo 4)
9: Escolha m por cento dos membros de P_s com probabilidade uniforme. Para cada um, inverta um bit selecionado aleatoriamente em sua representação. (Passo 5)
10: atualizar $P \leftarrow P_s$. (Passo 6)
11: Avaliação (Passo 2)
12: **end while**
13: **return** O indivíduo de P com o melhor valor de aptidão.

com um valor real em ponto flutuante, tal representação é geralmente aplicada a problemas contínuos com exigência de grau de precisão elevado (JANIKOW; MICHALEWICZ, 1991).

Além das representações de indivíduos já descritas, outra forma de codificar um cromossomo é a representação inteira, em que cada cromossomo é uma conjunto de números inteiros representando uma posição ou valor em uma sequência. Essa representação pode ser usada em problemas de otimização combinatória como o TSP e VRP (DAVIS, 1991). No TSP cromossomos com representação inteira descrevem a ordem em que o caixeiro (*salesmen*) visitará as cidades (Figura 9).

Na Figura 9, o cromossomo representa a ordem de visita em cada vértice (cidade) do grafo e o retorno para o vértice inicial. Essa representação é frequentemente utilizada para o TSP simétrico, o qual considera um caminho hamiltoniano permitindo passar por todos os vértices de um grafo G, não repetindo nenhum, ou seja, passar por todos uma só vez por cada.

2.5.1.2 Função de Aptidão

A Função *Fitness* ou função de aptidão é utilizada para avaliar cada indivíduo da população. Tal função pode ser modelada de acordo com o objetivo do problema. Em

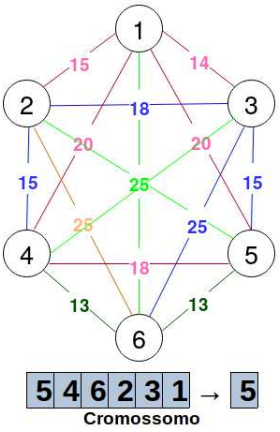


Figura 9 – Exemplo da representação de um cromossomo baseada em permutação para o problema do TSP simétrico.

(MITCHELL, 1997), a função *fitness* é definida como um critério para *rankear* potenciais indivíduos para a seleção probabilistica e inclusão na população da próxima geração.

No TSP, o peso de ponderação de uma aresta pode estar associado a distância percorrida entre as cidades (DAVENDRA, 2010). Para modelar uma função *fitness* para a representação do cromossomo apresentado na Figura 9, onde para cada aresta ligando um vértice i aos $n-1$ vértices, tem-se um valor ponderando essa ligação, é necessário calcular o somatório das distâncias. Na Equação 5 é apresentado uma função *fitness* para avaliar uma população de indivíduos representados como na Figura 9.

$$f(h) = \sum_{i=1}^{n-1} d(v_i, v_{i+1}) \tag{5}$$

Onde, h representa o indivíduo a ser avaliado, v_i o i-esimo vértice do grafo, n a quantidade de vértices e $d(v_i, v_{i+1})$ a distância associada entre 2 vértices.

Um outro exemplo de função *fitness* é apresentado em (DAVENDRA, 2010) o qual podemos representar com a Equação 6:

$$f(h) = \frac{1}{\sum_{i=1}^{n} d(v_i, v_{i+1})} \tag{6}$$

2.5.1.3 Operador de Seleção

A operação de seleção indica quais indivíduos (cromossomos) serão selecionados para fase de reprodução. No processo de seleção, indivíduos com os melhores valores de aptidão possuem maiores probabilidades de serem selecionados (DRÉO et al., 2006). Os operadores tradicionais de seleção são:

- **Seleção proporcional (método da roleta):** Neste método os indivíduos são selecionados para a fase de reprodução conforme os valores de aptidão extraídos da função objetivo durante a fase de avaliação. A probabilidade de seleção para um

indivíduo é definida pela Equação 7.

$$P_{(i)} = \frac{f_{(i)}}{\sum_{j=1}^{n} f_{(j)}} \tag{7}$$

Em que $P_{(i)}$ é a probabilidade de seleção do indivíduo i da população corrente, dada pela divisão do seu valor de aptidão ($f_{(i)}$) pela soma total de todos os valores de aptidão da população corrente. A seleção é baseada no método roleta, onde é definida uma roleta em que cada "fatia"contém o tamanho proporcional do valor de aptidão de um indivíduo i. A roleta tem tamanho N, sendo $N = \sum_{i=1}^{n} P_{(i)}$. A roleta é girada T vezes, onde T é a quantidade de indivíduos selecionados. Para cada giro da roleta, um valor é sorteado e o indivíduo com a fatia que contém esse valor na roleta é selecionado e submetido a fase de reprodução. Na Figura 10 é apresentado um esquema de seleção pelo método da roleta.

Indivíduo (Pi)	Aptidão f(Pi)	P(i)
P1	3,45	0,185
P2	8,15	0,437
P3	2,05	0,110
P4	3,33	0,179
P5	1,65	0,089

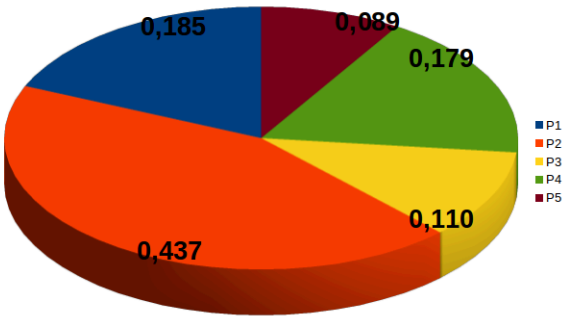


Figura 10 – Exemplo ilustrativo de seleção por roleta.

Como desvantagem, o método de seleção por roleta pode resultar em uma rápida estagnação do processo de busca, convergência prematura.

- **Seleção por rank:** Basicamente para cada geração, os indivíduos presentes na população corrente são ordenados de acordo com os valores de aptidão. O indivíduo com o pior valor de aptidão obtém *rank* 1 e o melhor obtém *rank* N, onde N é o tamanho da população. A probabilidade de seleção para um indivíduo é dada por um ranqueamento linear (Equação 8) (CHAKRABORTY; CHAKRABORTY, 1999).

$$p(x_i^{(t)}) = \frac{1}{N}\left(min + \frac{(max - min)(rank\left[x_i^{(t)}\right] - 1)}{N - 1}\right) \tag{8}$$

Onde $max + min = 2$ e $1 \leq max \leq 2$. Assim, $p(x_i^{(t)})$ corresponde a distribuição de probabilidade adequada para $x_i^{(t)}$, em um espaço amostral de N indivíduos.

- **Seleção por torneio:** Neste método, seleciona-se n indivíduos da população corrente aleatoriamente. Um torneio é conduzido, onde o indivíduo com a melhor aptidão é selecionado de maneira determinística ou probabilística. Tal procedimento

é repetido t vezes, onde t é igual ao número de indivíduos da população. Na Figura 11 é apresentado um esquema de seleção por torneio.

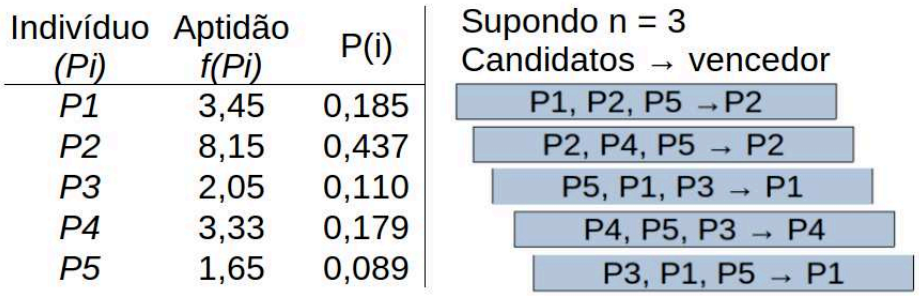


Figura 11 – Exemplo ilustrativo de seleção por torneio.

No torneio simples a seleção é feita sem considerar a aptidão relativa e no torneio estocástico a seleção é feita através do método da roleta. O parâmetro n permite definir a pressão seletiva durante a evolução, quanto maior o número de indivíduos que participam do torneio, maior será a pressão seletiva. Caso a pressão seletiva for muito baixa, a convergência se torna lenta, ou seja, o tempo para encontrar uma boa solução aumenta. Do contrário, se a pressão seletiva é muito elevada, ocasionará uma convergência prematura.

- **Elitismo:** Uma parte da população (os melhores pais) é mantida para a próxima geração. O objetivo é evitar que indivíduos com bons valores de aptidão sejam perdidos durante a fase de reprodução e mutação, onde são realizadas modificações nos genes dos indivíduos selecionados.

2.5.1.4 Operador de Cruzamento

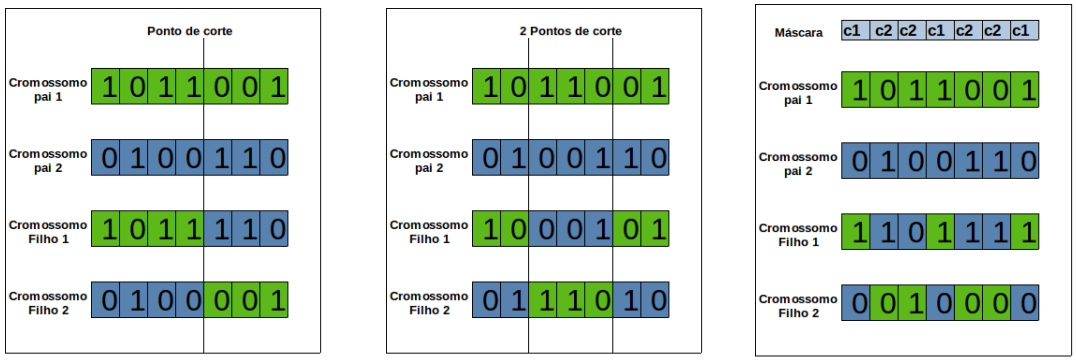
De acordo com a seleção natural, os indivíduos escolhidos com o operador de seleção, denominados pais, realizam um processo de cruzamento para reprodução de indivíduos filhos, os quais serão submetidos a próxima geração. O operador de reprodução conhecido como cruzamento (*crossover*) é responsável pela reprodução dos indivíduos através da troca de informação genética entre eles (DEEP; THAKUR, 2007). Para realização do *crossover* é necessário a utilização do parâmetro taxa de cruzamento, que determina a proporção de indivíduos que serão submetidos ao cruzamento (DRÉO et al., 2006). Segundo (DRÉO et al., 2006) exitem três operadores de cruzamento considerados clássicos na literatura:

- **Cruzamento de um ponto simples:** Inicialmente proposto por (HOLLAND et al., 1992), aplica um ponto de corte aleatoriamente em um par de cromossomos. Tal ponto particiona ambos cromossomos em duas partes. Sucessivamente, a primeira

parte do primeiro cromossomo pai é unida à segunda parte do segundo cromossomo pai, formando o primeiro cromossomo filho. O segundo cromossomo filho é gerado com a junção da primeira parte do segundo cromossomo pai e a segunda parte do primeiro cromossomo pai. A Figura 12-(a) ilustra o cruzamento de um ponto.

- **Cruzamento múltiplo:** São definidos de maneira aleatória múltiplos pontos de cortes. Os genes contidos entre os pontos de corte são trocados alternadamente entre os cromossomos pais (DRÉO et al., 2006). A Figura 12-(b) ilustra um cruzamento com dois pontos de corte.

- **Cruzamento uniforme:** Nesse tipo de cruzamento é utilizado uma estrutura de máscara do tamanho dos cromossomos e sorteia-se os genes que cada cromosso pai fornecerá ao primeiro cromossomo filho; o outro cromosso filho é gerado pelo complemento da máscara. A Figura 12-(c) ilustra o cruzamento uniforme.



(a) Cruzamento um ponto de corte (b) Cruzamento dois pontos de corte (c) Cruzamento uniforme

Figura 12 – Exemplos clássicos de operadores de cruzamento genético.

Além dos operadores clássicos de cruzamento, a literatura apresenta outros operadores baseados em ordem, também conhecido como operadores de permutação ou representação baseada em percurso, sendo um dos modos mais utilizados para representação de um cromossomo para instâncias do TSP no contexto dos GAs (LARRANAGA et al., 1999). Dois operadores bem conhecidos de cruzamento cromossômico para GAs baseados em permutação são o *partially mapped crossover* (PMX) e *cycle crossover* (CX) (LARRANAGA et al., 1999).

- ***Partially Mapped Crossover*:** O PMX foi proposto por (GOLDBERG; LINGLE et al., 1985), ele faz uso de dois cromossomos pais para gerar duas soluções filhas. Uma parte das cidades contidas na estrutura cromossômica do indivíduo pai é selecionada de forma aleatória e mapeada em conjunto com as cidades restantes do outro cromossomo pai que não estão contidas na primeira seleção para gerar um filho. Um exemplo de funcionamento do PMX é ilustrado na Figura 13.

  Na Figura 13, são apresentados dois cromossomos pais. O primeiro ponto de corte em ambos cromossomos foi definido entre o primeiro e o segundo gene e o segundo

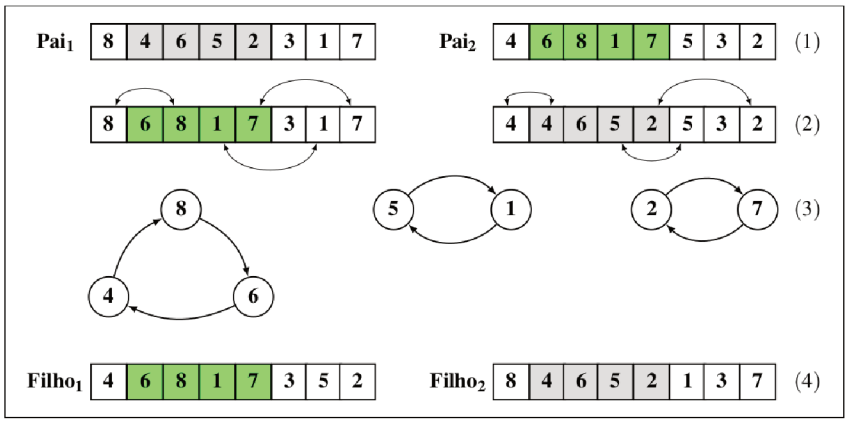


Figura 13 – Exemplo ilustrativo do *Partially Mapped Crossover* - PMX. Imagem extraída de (NERY, 2017).

ponto de corte entre o quinto e o sexto gene de cada solução. Assim a sublista de genes em cinza no cromossomo **(Pai 1)** é herdada pelo **(Filho 2)**, e a sublista de genes em verde cromossomo **(Pai 2)** é herdada pelo **(Filho 1)**, passo (2). Após o passo (2) os genes externos as sublistas nos filhos com valores duplicados são alterados conforme a ordem do mapeamento realizado no passo (3). Exemplo, o primeiro cromossomo do passo (2) contém genes com valores **1**, **7** e **8** duplicados. Então, é necessário trocar esses genes nas posições repetidas externas a região de corte por genes que estão nas mesmas posições das repetidas internas na região de corte do cromossomo **(Pai 1)**. Nesse caso, o gene com valor **1** que ocupa a sétima posição deve ser substituido pelo gene com valor **5** que ocupa a mesma posição no cromossomo **(Pai 1)** do valor repetido **1** dentro da região de corte, o mesmo se aplica às outros valores repetidos. Ao final do processo teremos como resultado os filhos **1** e **2** sem repetição de genes passo (4).

- ***Cycle Crossover*:** O CX também utiliza dois cromossomos pais para gerar sua descendência. Cada posição na estrutura cromossômica de um descendente deve ser preenchida por um gene na mesma posição correspondente há um dos pais (OLIVER; SMITH; HOLLAND, 1987). Assim, o operador CX garante que os genes presentes nos cromossomos pais serão herdados pelos filhos com a mesma ordem de ocorrência.

  O algoritmo CX recebe como entrada duas soluções pais **(P1)** e **(P2)**. No primeiro passo, seleciona-se de uma solução pai, o gene que encontra-se na primeira posição e insere este gene na primeira posição do filho. Posteriormente, seleciona-se a posição na qual o gene do passo anterior se encontra na cadeia de genes do outro pai e adiciona no filho a posição como um gene na mesma posição do pai. O processo é repetido até que a primeira posição do filho seja novamente selecionada.

  Na Figura 14, o filho **(F1)** é inicialmente gerado recebendo o gene **1** de **(P1)**. No próximo passo é verificada a posição do gene **1** de **(F1)** em **(P2)**, a posição é igual

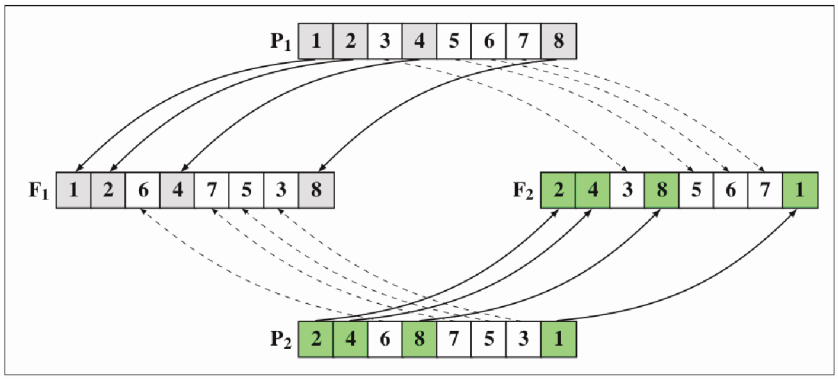


Figura 14 – Exemplo ilustrativo do *Cycle Crossover* - CX. Imagem extraída de (NERY, 2017).

a **8**, assim o valor **8** é adicionado na oitava posição de **(F1)**. Posteriormente é verificada a posição do gene **8** de **(F1)** em **(P2)**, a posição é igual a **4**, assim o valor **4** é adicionado na quarta posição de **F1**. Repetindo o mesmo processo, sera inserido o gene **2** na segunda posição de **(F1)** e o primeiro ciclo estará formado. O segundo ciclo é iniciado a partir de **(P2)**. Assim, a terceira posição de **(F1)** será preenchida com o valor da terceira posição de **(P2)**, onde o valor da terceira posição de **(P2)** é igual a **6**. O ciclo é finalizado pois, o gene **6** de **(F1)** já se encontra na sexta posição de **(P1)**. Outro ciclo é inciando onde a quinta posição de **(F1)** será igual a quinta posição de **(P2)**. Novamente o ciclo é finalizado pois, o gene **7** de **(F1)** já se encontra na sétima posição de **(P1)**. Com base nessa dinâmica os próximo genes de **(F1)** são **5** e **3**, finalizando assim a solução **(F1)**. A solução **(F2)** é gerada seguindo a mesma dinâmica, porem o primeiro ciclo é inciado de **(P2)** e os posteriores de **(P1)**.

2.5.1.5 Operador de Mutação

O processo de mutação ocorre com alteração de modo aleatório de algumas características genéticas de um conjunto de cromossomos selecionados de forma probabilística, sendo a probabilidade de seleção com valor pequeno (GOLDBERG, 1989). O processo de mutação esta associado a diversidade genética, com a possibillidade de exploração de novos pontos no espaço de busca, onde uma taxa de mutação devidamente equilibrada previne efeitos negativos ocasionados por uma pressão seletiva elevada (DRÉO et al., 2006).

Considerando uma representação binária, o operador de mutação padrão troca o valor de um gene em um cromossomo (HOLLAND et al., 1992). Desse modo, se o gene selecionado tem o valor 0, o seu valor passará a ser 1, caso contário o valor será 0. Já em problemas com represenção em ponto flutuante, os operadores de mutação mais frequentemente usados são mutação uniforme e mutação gaussiana (MICHALEWICZ;

SCHOENAUER, 1996).

Um operador de mutação comum em representação por permutação que pode ser aplicado ao TSP é o conhecido como 2-*opt* (LIN; KERNIGHAN, 1973). Neste procedimento são selecionados dois pontos de um cromossomo e revertido os genes entre os pontos. Exemplo: [1 | 2 3 4 5 | 6] → [1 | 5 4 3 2 | 6]. Este operador também pode ser extendido para *k*-*opt*, onde k pontos são selecionados e as sub-sequências são revertidas.

Um outro caso particular da mutação 2-*opt* é o que duas posições (alelos) são selecionados e seus genes trocados, conhecida como mutação baseada em ordem (SYSWERDA, 1991).

2.5.1.6 Operador de Reinserção

Após os processos de seleção, cruzamento, mutação e avaliação dos cromossomos descendentes da população corrente, a próxima etapa é selecionar os mais aptos para a próxima geração. Tal processo consiste em definir um operador de reinserção. Segundo (SILVA, 2011) os operadores de reinserção frequentemente aplicados nos GAs são:

- **Reinserção pura:** é realizada a substituição de todos os cromossomos pais da população corrente por todos os cromossomos descendentes. Frequentemente acompanhado por elitismo.
- **Reinserção uniforme:** seleciona os cromossomos mais aptos da população total (pais + filhos) para a próxima geração atráves de algum operador de seleção tradicional.
- **Reinserção por elitismo:** uma porção dos cromossomos pais mais aptos da população corrente são mantidos para a próxima geração.
- **Reinserção baseada na aptidão:** ordena-se todos os cromossomos da população corrente (pais e filhos) de acordo com suas aptidões e os t mais aptos são selecionados diretamente para a próxima geração.

2.5.1.7 Trabalhos com Aplicação do GA ao TSP e MTSP

Em particular, GAs são frequentemente usados para resolver variantes do TSP e VRP. Por exemplo o problema de coleta e entrega que combina roteamento de veículos com distribuição de objetos. O objetivo é encontrar rotas ideais de visitas para veículos que devem transportar passageiros ou mercadorias de locais de captação para locais de entrega. Esse problema é abordado em (LIAO; CHIEN; TING, 2014), no qual os autores desenvolveram duas técnicas para melhorar o desempenho de um GA.

Um GA é proposto em (YU; LU, 2014) para planejar as rotas de uma máquina automática de corte de roupa. O objetivo foi reduzir o tamanho do caminho de corte e

melhorar a suavidade do movimento. Os autores fizeram algumas melhorias em relaçao aos métodos de mutação e *crossover* para acelerar a convergência e evitar ótimos locais. Em (SAKURAI et al., 2010a) e (SAKURAI et al., 2010b) são apresentados GAs de tempo-real para escalonar rotas de entrega usando a representação do TSP.

Para resolver o mTSP, um algoritmo híbrido que combina GA com busca local guiada pela heurística 2-opt é proposto por (SEDIGHPOURA; YOUSEFIKHOSHBAKHT; DARANI, 2011). Outra abordagem para o planejamento de rotas de entrega com o GA, usando a representação do mTSP, é apresentada por (SADIQ, 2012), que aplica uma fase de agrupamento para alocar pacotes semelhantes juntos antes da geração das rotas.

(KIRáLY; ABONYI, 2010) propõe uma forma de modelar um indivíduo para o mTSP usando um cromossomo para cada caixeiro. Em (ARYA; GOYAL; JAISWAL, 2014), o indivíduo tem um único cromossomo de comprimento $p+q$, onde os nós são representados por uma permutação de inteiros de 1 a p. Essa permutação é particionada em sub-viagens q pela inserção de inteiros negativos de q (de 1 a q) que representam a mudança de um caixeiro para o próximo.

(ZHOU; SONG; PEDRYCZ, 2018) apresentam um estudo comparativo de melhoria de um GA e PSO para resolver o mTSP com múltiplos depósitos, caminho fechado e exigência de um número mínimo de cidades que cada caixeiro percorreria. Os autores propõem dois GAs, sendo um GA com método de seleção roleta e elitista, na qual quatro novos tipos de operação de mutação foram propostos. O outro GA vincula a seleção e a mutação. Eles também usaram um novo operador de seleção e mutação mais abrangente. Para a análise comparativa eles adotaram o PSO e um método chamado de Algoritmo de Otimização Invasivo de Ervas Daninhas (IWO) aplicados ao mTSP. Os algoritmos foram validados com as instâncias do TSPLIB[6] onde foram realizados experimentos comparativos.

2.5.2 Otimização por Enxame de Partículas

O algortimo de Otimização por Enxame de Partículas (do inglês, *Particle Swarm Optimization* - PSO), foi inicialmente proposto por (KENNEDY; EBERHART, 1995), que com base em observações do comportamento social dos animais, como, pássaros e peixes, desenvolveram a primeira versão da meta-heurística para resolver problemas de otimização contínua.

2.5.2.1 PSO Aplicado a Problemas de Otimização com Espaços Contínuos

Em (KENNEDY; EBERHART, 1995) foi introduzido um método para otimização de funções não lineares contínuas descrevendo os conceitos do PSO, algoritmo que simula um comportamento social. Basicamente, um bando de pássaros voando aleatoriamente no espaço de busca, onde cada pássaro é uma solução (partícula).

[6] Instâncias de amostras para o TSP e problemas relacionados.

No PSO alguns termos são empregados para representação dos compontentes técnicos do algoritmo. O termo partícula refere-se a cada indivíduo do grupo, um grupo de indivíduos é conhecido como enxame ou população. As partículas voam através das coordenadas do espaço de busca n-dimensional, quando se movem, enviam suas coordenadas para uma função de aptidão. A posição e velocidade de cada partícula são definidas pelos vetores $x_i^{(t)} = (x_{i1}^{(t)}, x_{i2}^{(t)}, ...x_{in}^{(t)})$ e $v_i^{(t)} = (v_{i1}^{(t)}, v_{i2}^{(t)}, ...v_{in}^{(t)})$, respectivamente. Usando a função de aptidão (*fitness*), cada partícula é avaliada em cada estágio do algoritmo. Cada partícula (indivíduo) tem seu movimento baseado em três parâmetros: fator de sociabilidade, fator de individualidade e velocidade. Combinando os três parâmetros com um valor gerado aleatoriamente o algoritmo determina o próximo local da partícula no espaço de busca da solução (OLIVEIRA; SILVA; ALOISE, 2004).

- ❑ **Fator de sociabilidade:** define a inclinação das partículas para melhor posição encontrada por qualquer indivíduo do enxame;

- ❑ **Fator de individualidade:** define a inclinação da partícula com sua melhor posição já descoberta;

- ❑ **Velocidade maxíma:** define o movimento da partícula.

Com base no fator social e fator individual, cada partícula possui um peso para se basear em melhores posições e também nas melhores posições do enxame. O fator de individualidade é responsável por gerar uma diversificação. Nele tem-se a variavél *pbest*, que é basicamente a memória de uma melhor posição encontrada até o momento, e a velocidade associada à essa variável auxilia a partícula a retornar a uma boa posição passada. Já no fator de sociabilidade tem-se a variável *gbest* que é basicamente o conhecimento público, com uma velocidade que leva a melhor posição no espaço de busca de soluções que uma partícula entre todas do enxame já esteve. Para a atualização de uma partícula é necessário avaliar se a solução encontrada é melhor que *pbest* e *gbest*.

A velocidade de uma partícula é obtida por três termos: inércia, cognitivo e social. Cada termo da velodicade da partícula é representado por uma parte da Equação 9 que apresenta o cálculo da velocidade da partícula. A inércia é dada por $w^t.v_i^t$, o termo cognitivo é dado por $c1.r1.(pbest_i)$ e o termo aprendizado social é dado por $c2.r2.(gbest - x_i^t)$.

$$v_i^{t+1} = w^t.v_i^t + c1.r1.(pbest_i - x_i^t) + c2.r2.(gbest - x_i^t) \tag{9}$$

Os parâmetros $r1$ e $r2$ são números aleatórios obtidos em uma distribuição uniforme $[0, 1]$, $c1$ e $c2$ são constantes de aceleração positivas associadas ao fatores cognitivo e social. O termo w é uma constante de inércia na geração t que controla a capacidade da busca

local e global. Essa constante pode variar ao longo das iterações de acordo com a equação 10.

$$w^t = w_{max} - \frac{w_{max} - w_{min}}{t_{max}} \tag{10}$$

em que w_{max} e w_{min} representam os valores máximos e mínimos da constante de inércia, sendo $w_{max} = 0.9$ e $w_{min} = 0.4$, de acordo com (JORDEHI; JASNI, 2013), e t_{max} é o número máximo de gerações.

A Equação 11 limita a velocidade de v_i^{t+1} evitando que as partículas saiam do espaço de busca.

$$v_i^{t+1} = \begin{cases} v_i^{t+1} & se \quad -v_{max} \leq v_i^{t+1} \leq v_{max} \\ -v_{max} & se \quad v_i^{t+1} < -v_{max} \\ v_{max} & se \quad v_i^{t+1} > v_{max} \end{cases} \tag{11}$$

O deslocamento de uma partícula no espaço de busca é efetuado pela Equação 12.

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{12}$$

Em que x_i^{t+1} é a nova posição da partícula no espaço de busca dada pela adição da nova velocidade v_i^{t+1} a posição atual x_i^t. Assim como a velocidade, é necessário limitar os valores que a nova posição x_i^{t+1} apresentará com a movimentação da partícula (Equação 13).

$$x_i^{t+1} = \begin{cases} x_i^{t+1} & se \quad x_{min} \leq x_i^{t+1} \leq x_{max} \\ x_{max} & se \quad x_i^{t+1} > x_{max} \\ x_{min} & se \quad x_i^{t+1} < x_{min} \end{cases} \tag{13}$$

A representação da movimentação de uma partícula pode ser visualiza na Figura 15, supondo $w = 1$, $c1.r1 = 1$ e $c2.r2 = 1$.

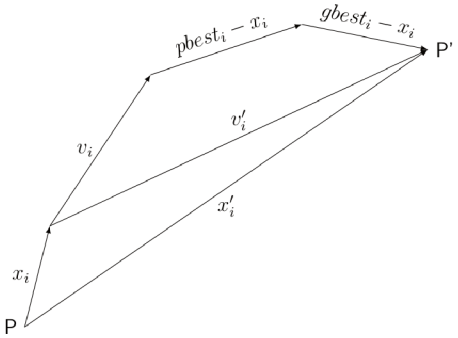


Figura 15 – Movimentação de uma partícula de acordo com a sua velocidade no espaço de busca. Imagem extraída de (PINOTTI, 2017).

A cada geração t do Algoritmo 2, o vetor de velocidade para cada partícula é modificado com base nos três termos definidos anteriormente. Então, com base na velocidade determinada para cada partícula, ocorre uma movimentação para uma nova posição no

espaço de busca. Eventualmente, é provável que o enxame, como um todo, se aproxime de um nível ótimo (KENNEDY; EBERHART, 1995)

Algoritmo 2 PSO - minimização em espaço contínuo

Entrada: o tamanho da população n, o número de gerações T, constante de aceleração cognitivo *c1*, constante de aceleração social *c2*, peso de inércia w.

Saída: O melhor indivíduo *gbest*.

1: $p \leftarrow$ Gerar n (Passo 1)
2: **for** cada i em p **do**
3: Inicializar o vetor posição x_i com valores aleatórios (Passo 2)
4: Inicializar o vetor velocidade v_i com vazio (Passo 3)
5: **end for**
6: $t \leftarrow 0$
7: **while** $t < T$ **do**
8: **for** cada i em p **do**
9: Calcular o valor de aptidão $f(p_i)$ (Passo 4)
10: **if** $f(p_i) < f(pbest)$ **then**
11: $pbest \leftarrow p_i$ (Passo 5)
12: **end if**
13: **if** $f(p_i) < f(gbest)$ **then**
14: $gbest \leftarrow p_i$ (Passo 6)
15: **end if**
16: **end for**
17: **for** cada i em p **do**
18: $v_i^{t+1} = w^t.v_i^t + c1.r1.(pbest_i - x_i^t) + c2.r2.(gbest - x_i^t)$ (Passo 7)
19: $x_i^{t+1} = x_i^t + v_i^{t+1}$ (Passo 8)
20: **end for**
21: **end while**
22: **return** *gbest*.

O Algoritmo 2 apresenta estrutura do PSO com aplicação em problemas de minimização em espaços de buscas contínuos.

2.5.2.2 PSO Aplicado a Problemas de Otimização com Espaços Discretos

O PSO apresentando anteriormente é restrito a espaços de busca contínuos, ou seja, números reais. Contudo, muitos problemas de otimização apresentam caracteristicas discretas. Com isso, viu-se a necessidade de desenvolver novas variações do PSO para resolver problemas de otimização combinatória como o TSP (PANG et al., 2004), VRP (CHEN; YANG; WU, 2006) e problemas de escalonamento (LIAO; TSENG; LUARN, 2007).

A primeira versão do PSO aplicado a espaços de busca discretos é apresentada em (KENNEDY; EBERHART, 1997). Nessa versão o PSO difere da versão original (contínua) em dois aspectos. O primeiro aspecto é que as partículas são compostas por variáveis binárias. No segundo, a velocidade deve ser transformada na mudança de probabilidade, que é a chance da variável binária assumir o valor um. Uma posição,

$X_i^t = (x_{i1}^t, x_{i2}^t, ..., x_{iD}^t), x_{id}^t \in \{0, 1\}$ pertence a partícula i com D *bits* na iteração t, onde X_i^t é vista como uma pontecial solução com uma taxa de mudança denominada velocidade. A velocidade é apresentada por $V_i^t = (v_{i1}^t, v_{i2}^t, ..., v_{iD}^t), v_{id}^t \in R$. $P_i^t = (p_{i1}^t, p_{i2}^t, ..., p_{iD}^t)$ é a melhor solução que a partícula i obteve até a iteração t, e $P_g^t = (p_{g1}^t, p_{g2}^t, ..., p_{gD}^t)$ é a melhor solução obtida por P_i^t na população (*gbest*). Assim como no PSO contínuo, cada partícula ajusta sua velocidade de acordo com a Equação 9.

Uma outra versão do PSO aplicada em problemas discretos é apresentada em (CLERC, 2004). Nessa versão foram introduzidos alguns conceitos e modificações nos operadores do PSO contínuo. Uma modificação foi feita no vetor velocidade, a qual passa a ser uma lista de transposições. Uma velocidade nula (vazia) é uma lista sem transposições. Sendo P uma posição e V uma velocidade, logo, P' será uma nova posição em que $P' = P + V$, $P + V$, faz referência a aplicação de tranposições no vetor velocidade em P (OLIVEIRA; SILVA; ALOISE, 2004).

Para obter a velocidade é realizado a subtração entre duas posições P_1 e P_2, sendo $V = P_1 - P_2$. A seguir é apresentado um exemplo extraído de (OLIVEIRA; SILVA; ALOISE, 2004): Dadas duas posições A e B:

A: (1 2 3 4 5)
B: (2 3 1 5 4)

- A(1) = B(3) = 1. Assim, a primeira transposição sera (1,3) então teremos um B' = B + S(1,3) → B' = (1 3 2 5 4)
- A(2) = B'(3) = 2. Assim, a segunda transposição sera (2,3) então teremos um B" = B' + S(2,3) → B" = (1 2 3 5 4)
- Assim, o terceiro operador será (4, 5), logo teremos a lista de transposição (V), que é igual: S(1,3), S(2,3),S(4,5)
- Desse modo, A - B = V = [(1,3),(2,3),(4,5)].

A adição entre duas velocidades equivale a concatenação de listas de transposições, tendo assim uma nova lista. A multiplicação do coeficiente pela velocidade pode se dá em quatro casos:

1. C = 0, teremos C * V = nulo;
2. C pertence [0, 1], truncamos V por C*[V];
3. C > 1, aplicamos V, C vezes mais a parte decimal de C *[V];
4. C < 0, não é definido pra este tipo de implementação.

Outras versões do PSO para resolução de problemas em espaços discretos vem sendo apresentadas na literatura, como a versão apresentada em (LIAO; TSENG; LUARN, 2007) que extende a versão de (KENNEDY; EBERHART, 1997) para resolver problemas de escalonamento. Também são apresentados algoritmos híbridos com a combinação do PSO com os operadores genéticos do GA como em (ABDEL-KADER, 2011). E uma versão conhecida como S-PSO (*Set-Based Particle Swarm Optimization*) que usa um conjunto de representações e caracteriza o espaço de busca discreto do contínuo, sendo a solução candidata e a velocidade definidas como um conjunto com possibilidades respectivas (CHEN et al., 2009).

2.5.2.3 Trabalhos com Aplicação do PSO ao TSP e MTSP

A aplicação do PSO ao VRP tem sido constatemente discutida e apresentada na literatura, destaca-se os seguintes trabalhos:

Em (WU; TAN, 2009) uma versão híbrida do PSO com ACO é aplicada ao VRP para otimização da logística de grãos. O objetivo é usar uma frota de veículos com capacidade para atender uma série de clientes com restrições fixas de demanda e janela de tempo. O algoritmo foi desenvolvido em duas etapas. Na primeira etapa é aplicado o PSO e sucessivamente na segunda etapa o ACO, funcionando como uma busca local atráves da estratégia desenvolvida no trabalho denominada mecanismo guiado por feromônio.

Outro exemplo de desenvolvimento de uma meta-heurítica híbrida é apresentado em (MARINAKIS; MARINAKI, 2010), onde os autores desenvolveram um algoritmo para resolver o VRP combinando PSO (KENNEDY; EBERHART, 1995), com GA, MPNS-GRASP (MARINAKIS; MIGDALAS; PARDALOS, 2009) e estratégias de expansão da vizinhança de busca (MARINAKIS; MIGDALAS; PARDALOS, 2005).

No trabalho de (OKULEWICZ; MAŃDZIUK, 2013) foi desenvolvido um PSO em dois estágios, e aplicado ao problema de roteamento dinâmico de veículos para resolver o problema de atribuição do cliente e otimizar as rotas. Na primeira etapa o PSO é aplicado para encontrar solicitações de clientes que devem ser atendidas pelo mesmo veículo, já na segunda etapa várias outras instâncias do PSO são aplicadas, uma para cada um dos veículos atribuídos a solicitações específicas (resolvendo o TSP).

Em (GOKSAL; KARAOGLAN; ALTIPARMAK, 2013) é apresentado uma versão híbrida do PSO discreto com o algoritmo de busca local descida em vizinhança variável (do inglês, *Variable Neighborhood descent* - VND) aplicado ao VRP com coleta e entrega simultâneas. No trabalho também é implementado uma estratégia semelhante ao algoritmo de recozimento simulado para preservar a diversidade das soluções.

(GULCU; ORNEK, 2019) propõem 2 algoritmos baseados no PSO para resolver o mTSP. Os algoritmos apresentados pelos autores são chamados de APSO e HAPSO. O algoritmo APSO é baseado no PSO e no algoritmo 2-opt, com operadores de reconexão e troca. Já o algoritmo HAPSO é baseado nos algoritmos: GRASP, PSO e 2-opt, com

operadores de reconexão e troca. Nos experimentos foram usadas 5 instâncias do TSP como mTSP e realizada uma comparação com os algoritmos GA e ACO. De acordo com os resultados apresentados no trabalho, o algoritmo HAPSO superou os demais algoritmos em mais instâncias.

2.5.3 Otimização por Colônia de Formigas

O algoritmo de Otimização por Colônia de Formigas (do inglês, *Ant Colony Optimization - ACO*, foi proposto por Marco Dorigo (DORIGO, 1992). O algoritmo tem sua ideia no conceito do comportamento cooperativo de algumas espécies de formigas. As formigas depositam e são atraídas por uma subtância chamada feromônio, que concentradas em trilhas, possibilitam o reforço das mais percorridas, as quais são possivelmente as melhores, pois a presença de ferômonio em grande quantidade significa que ainda não ocorreu a evaporação e sim a passagem de mais formigas em pouco tempo. Esse mecanismo torna o ACO um algoritmo poderoso para solucionar problemas de otmização combinatória, pois, combinando o depósito de feromônio (reforço positivo) e a evaporação (reforço negativo), é possivel que se evite, em grande parte dos problemas uma convergência prematura do algoritmo para soluções não ótimas (DORIGO; CARO, 1999).

Em (DORIGO; MANIEZZO; COLORNI, 1996) foi proposto uma heurística com objetivo de resolver problemas de otimização combinatória conhecida como *Ant System* (AS). A aplicação do AS ao TSP apresentou resultados encorajadores, e desde de então o algoritmo e suas variações vem sendo aplicados com sucesso em vários problemas NP-difíceis de otimização combinatória (DORIGO; STUTZLE, 2004).

A versão AS apresentada em (DORIGO; MANIEZZO; COLORNI, 1996) aponta bons resultados para instâncias do TSP com poucas cidades. Porém, os autores vieram apresentando trabalhos com objetivo de mostrar as melhorias feitas na heurística, apresentado em (DORIGO; GAMBARDELLA, 1997) o *Ant Colony System* ACS, uma meta-heurística que resultou em uma melhoria do AS, possuindo um desempenho superior e aplicabilidade a problemas de tamanhos mais consideráveis.

O ACS apresenta duas diferenças para o AS. A primeira diferença se refere ao conceito e forma de atualização dos feromônios. Na versão AS, a atualização é feita de forma global e única, após o término de cada iteração, onde é realizada de maneira simultânea a evaparação e depósito de feromônio em todos os caminhos (arcos do grafo) percorridos de forma relativa a qualidade do caminho (JÚNIOR, 2013). Já no ACS, a atualização dos feromônios acontece na fase local após a ação de cada formiga, caracterizando a evaporação dos feromônios, e na fase global após o fim de cada iteração, compensando apenas o melhor caminho encontrado pelas formigas com depósito de uma quantidade proporcional a qualidade da solução encontrada (JÚNIOR, 2013).

Inicialmente o ACS faz uma atualização local em cada arco, ou seja, sempre que uma formiga realiza ação de atravessar um arco, a quantidade de feromônio em tal arco é

alterada de acordo com a Equação 14

$$\tau_{ij}(t) \leftarrow (1 - \xi)\tau_{ij}(t) + \rho\tau_0 \tag{14}$$

Na Equação 14, τ_0 é a quantidade de feromônio em cada arco. Esta quantidade inicial de feromônio é definida como o inverso do comprimento da solução encontrada atráves do método *Nearest Neighbor* com $1/L_{mn}$, onde L_{mn} corresponde ao custo da solução encotrada podendo ser multiplicado pelo número de nós (cidades) do problema $(1/n.L_{mn})$ diminuindo ainda mais a concentração incial de ferômonio (JÚNIOR, 2013). Os parâmetros ξ e ρ definem a velocidade com que a densidade de feromônio decai, sendo ξ relacionado a evaporação local e ρ a evaporação global.

No AS a regra de atualização do feromônio é única possuindo o princípio de evaporação e depósito na mesma regra (DORIGO; MANIEZZO; COLORNI, 1996). Ou seja, cada formiga é responsável por depósitos e pela evaporação. No caso do ACS, a regra é diferente, sendo o reforço da trilha (depósito) não mais realizado por cada formiga, e sim de forma unificada e centralizada (fase de atualização global) (JÚNIOR, 2013). Ocorrendo no final de cada iteração, onde o melhor caminho encontrado é o unico que tem sua densidade de feromônio incrementada, o que difere do AS. A regra de atualização global do ACS é apresentada na Equação 15:

$$\tau_{ij}(t) \leftarrow (1 - \rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t) \tag{15}$$

Na Equação 15, $\Delta\tau_{ij}(t)$ é igual ao inverso do comprimento do melhor caminho. Ou seja, quanto melhor for a qualidade da solução, ou se tratando do TSP, quanto menor for o comprimento desta solução, maior será o depósito de feromônio adicionado aos arcos desta solução (JÚNIOR, 2013).

A segunda e principal diferença entre os algoritmos esta na fase de construção da solução, onde é usada uma Regra de Transição de Estados (RTE) ou regra de decisão das formigas. A RTE é uma regra que determina a probabilidade de um vértice ser escolhido por uma formiga. Em que, i representa o vértice no qual a formiga k é encontrada e j um dos possíveis vértices para os quais ela pode se mover (BARBOSA; KASHIWABARA et al., 2015). A formiga k deve apenas visitar os vértices ainda não visitados até a fase atual da construção da solução. No algoritmo AS, a regra é considerada unicamente probabilística ou estocástica, sendo que os caminhos com maiores concentrações de feromônios sejam mais prováveis de serem escolhidos pelas formigas (BARBOSA; KASHIWABARA et al., 2015). No ACS a regra é tida como pseudoaleatória, onde envolve um parte determinística, ou exploratória, e uma parte probabilística (RTE do AS) (BARBOSA; KASHIWABARA et al., 2015). Devido a RTE inserida no ACS, o algoritmo apresenta um parâmetro extra denominado $q0$, tal parâmetro varia de 0 a 1 e define o poder de exploração de novos caminhos. Antes de uma formiga realizar a escolha por um caminho através da RTE, um número aleatório q de 0 a 1 é sorteado. Se o número sorteado

for maior que $q0$, a RTE a ser usada para a decisão da formiga é a apresentada no AS (Equação 16), ou seja, a formiga decide para qual caminho irá se direcionar com base nas probabilidades calculadas para cada caminho factível. Caso o número sorteado q for menor que $q0$, a formiga tomará sua decisão baseando-se no conhecimento disponível sob a forma de depósitos de feromônios e distâncias. Ou seja, a escolha da formiga é baseada em um j que maximize o valor de $\tau_{i\iota}[\eta_{i\iota}]^{\beta}$ (Equação 17).

$$p_{ij}^{k} = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}} & \text{se } j \text{ é uma aresta permitida à } k \\ 0 & \text{caso contrário} \end{cases} \tag{16}$$

Na Equação, 16 i representa o vertice em que a formiga k se encontra, j um dos possíveis vértices para o deslocamento, η_{ij} corresponde a visibilidade da aresta, α e β são parâmetros que definem o peso da trilha de feromônio e visibilidade, respectivamente.

$$j = \begin{cases} \arg\max_{\iota \in S_i^k}\{\tau_{i\iota}[\eta_{i\iota}]^{\beta}\}, & \text{if } q \leq q0 \\ J, & \text{caso contrário} \end{cases} \tag{17}$$

Na Equação, 17 j representa a cidade escolhida por uma formiga k que se encontra no vértice i. A equação é conhecida como regra proporcional pseudo-aleatória, já que o parâmetro $q0$ define a porcentagem de escolhas que serão feitas deterministicamente pelas formigas, quando $0 \leq q0 \leq 1$. Se q é uma variável aleatória distribuída em $[0, 1]$ e atualizada a cada movimento, e $q0 = 1$, as escolhas das formigas serão determinadas deterministicamente pelo maior valor de $\tau_{i\iota}[\eta_{i\iota}]^{\beta}$, onde $\iota \in S_i^k$ corresponde ao conjunto de cidades que podem ser visitadas por uma formiga k no movimento. Se $q0 = 0$, as escolhas das formigas serão realizadas de maneira estocástica com J sendo a cidade escolhida através do cálculo da probabilidade descrito na Equação 16.

O ACS é uma das várias melhorias aplicadas ao AS. Na literatura são apresentados outros algoritmos buscando aperfeiçoamentos para o AS, entre eles o *Elitist AS*, *Ant-Q*, *MAX-MIN AS* e o *Rank-based AS* (DORIGO; STUTZLE, 2004). O Algoritmo 3 apresenta o pseudocódigo do ACS.

2.5.3.1 Trabalhos com Aplicação do ACO ao TSP e MTSP

Na literatura são apresentados diversos trabalhos que aplicam o ACS no problema de roteamento de veículos VRP. A seguir são resumidos alguns desses trabalhos:

Uma aplicação do ACS ao VRP é apresentada em (GAMBARDELLA; TAILLARD; AGAZZI, 1999), o qual aplica dois algoritmos ACS visando dois objetivos, sendo o primeiro com função de *fitness* que busca a minimização do número de veículos enquanto o segundo a minimização da distância viajada. A cooperação entre os dois algoritmos é realizada pela troca de informações através da atualização dos feromônios.

Em (REIMANN; DOERNER; HARTL, 2004) é apresentado uma versão do ACS baseado no trabalho de (REIMANN; STUMMER; DOERNER, 2002). Os autores realizaram

Algoritmo 3 ACS adaptado de (DORIGO; GAMBARDELLA, 1997)

Entrada: o tamanho da população T_p, o número de iterações T, influência do feromônio α, informação heurística β, taxa de evaporação local ξ, taxa de evaporação global ρ, valor do feromônio inicial τ_0, parâmetro de escolha RTE q_0.

Saída: A melhor solução.

1: Inicializar a população $P \leftarrow T_p$ formigas (Passo 1)
2: Inicializar a tabela de feromônio $\tau_{ij} \leftarrow \tau_0$ (Passo 2)
3: $f(S_k) \leftarrow \infty$
4: $t \leftarrow 0$
5: **for** cada t em T **do**
6: **for** cada k em P **do**
7: $q \leftarrow$ *sorteio()*(Passo 3)
8: $j = \begin{cases} \arg\max_{\iota \in S_i^k}\{\tau_{i\iota}[\eta_{i\iota}]^\beta\}, & \text{if } q \leq q0 \\ J, & \text{caso contrário} \end{cases}$ (Passo 4)
9: $\tau_{ij}(t) \leftarrow (1-\xi)\tau_{ij}(t) + \rho\tau_0$ (Passo 5)
10: Avaliar solução construída para $k \leftarrow f(S_k)$ (Passo 6)
11: **if** $S_k < L_{melhor}$ **then**
12: $L_{melhor} \leftarrow S_k$(Passo 7)
13: **end if**
14: **end for**
15: $\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t)$(Passo 8)
16: **end for**
17: **return** S_k

um estudo computacional e análise estatística em instâncias de benchmark padrão e em novas instâncias em grande escala definidas para VRP . Como resultado apresentaram as melhorias em relação ao algoritmo e uma ferramenta para solução o VRP em tamanho real.

Em (MONTEMANNI et al., 2005) é proposto uma versão do ACS aplicado a problemas dinâmicos de roteamento de veículos, também chamado de roteamento on-line. A abordagem proposta baseia-se na partição do dia útil em intervalos de tempo. Os autores modelaram o problema de modo a remover a necessidade de que os veículos tivessem que retornar ao depósito. O algoritmo desenvolvido foi integrado a um sistema centralizado de modo que pudesse funcionar de maneira centralizada pelas pessoas encarregadas das ordens de despacho. Uma análise comparativa foi realizada com o algoritmo GRASP aplicado em instâncias criadas pelos próprios autores, e realização de um estudo de caso na cidade de Lugano, sul da Suíça.

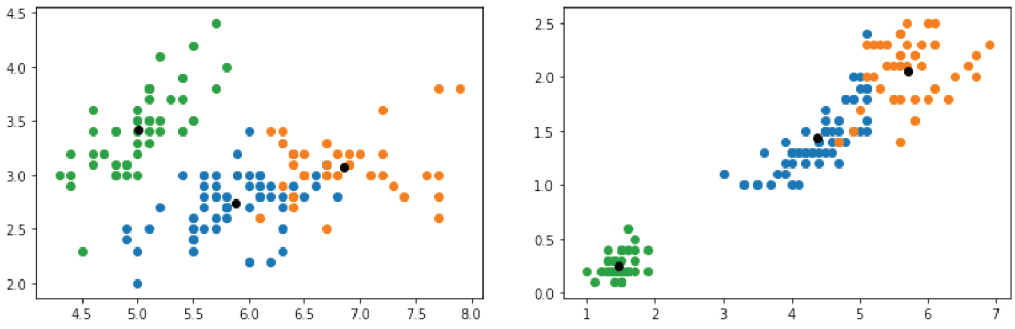
Em (LI; TIAN, 2006) é apresentado um ACS híbrido com busca local aplicado ao VRP. Além disso, um procedimento de pós-otimização foi incorporado ao algoritmo proposto buscando uma melhor condução no processo de busca de soluções otimas. Em (LI; TIAN; LEUNG, 2009), os autores desenvolveram e aplicaram ao VRP um ACS combinado com busca tabu. Outra abordagem híbrida aplicada ao VRP é apresentada

em (CHENGMING, 2008), onde os autores combinam um método estocástico de busca local com o algoritmo ACS. A abordagem apresentou bons resutados e proposta de uma nova abordagem estocástica para otimização do VRP.

Em (CHEN et al., 2018) é apresentado o problema de ter uma equipe de robôs móveis para visitar um conjunto de localizações. O problema é formulado como mTSP com um depósito único e depósito múltiplo que é um problema determinístico polinomial difícil. No trabalho os autores propõem um algoritmo biométrico baseado no ACO para resolver o problema. No algoritmo, um multi-ACO simples é integrado a um sistema sequencial. Os autores também propõem um método de otimização local para mTSP com 2 objetivos para melhorar as soluções candidatas.

2.6 Algoritmo K-means

K-médias ou *K-means* é uma técnica de agrupamento de dados (*data clustering*) proposta em (MACQUEEN et al., 1967). Tal técnica consiste em construir k partições dos dados utilizando um critério de similaridade (TAN; STEINBACH; KUMAR, 2016). Basicamente, o algoritmo atribui a cada ponto o centro mais próximo e escolhe o próximo centro (centróide) do grupo (*cluster*) como a média dos pontos que foram atribuídos a ele. Na Figura 16 é apresentado um exemplo de agrupamento realizado com *K-means*.



(a) **(b)**

Figura 16 – Agrupamento de dados com aplicação do K-means ao conjunto de dados Iris. (a) Eixo Y: comprimento da sépala. Eixo X: largura da sépala. (b) Eixo Y: comprimento da pétala. Eixo X: largura da pétala. Centróides representados pelos pontos pretos.

Para a Figura 16, os dados foram agrupados em três partições, sendo k igual a três. Os grupos estão representados nas cores azul, verde e laranja, e os centróides em preto.

A estratégia do algoritmo é agrupar instâncias de dados de maneira que a distância entre os pontos pertencentes a cada grupo seja minimizada. Sendo assim o *k-means* tenta encontrar os melhores centróides dos grupos. O Algoritmo 4 apresenta o pseudocódigo do *k-means*.

Algoritmo 4 K-means (MITCHELL, 1997)

Entrada: o número de grupos k, o conjunto de dados x.
Saída: k grupos e centróides.

```
1: Definição dos k centros aleatoriamente
2: while houver mudanças nos centróides do
3:    for cada i em x do
4:       Atribuir x_i ao grupo associado com o centroíde c_j mais próximo
5:       Atualizar os centróides c_j de cada cluster
6:       for cada j em c do
7:          c_j = Σ_{j=1}^{k} ||x_i − c_j|| / N
8:       end for
9:    end for
10: end while
```

O Algoritmo 4 é dividido em dois estágios: um estágio inicial e um iterativo. O primeiro estágio envolve a definição dos k centróides; e o segundo estágio consiste no cálculo dos k novos centróides (MITCHELL, 1997). Para o cálculo dos novos centróides é necessário realizar o cálculo da medida de proximidade dos dados entre os pontos de dados (Equação 18), sendo o objetivo da clusterização minimizar o quadrado da distância de cada ponto até o seu centro mais próximo.

$$d(x_p\dot{c}_j) = \sqrt{\sum_{i=1}^{N}(x_{pi} - c_{ji})^2} \tag{18}$$

Em que, $d(x_p\dot{c}_j)$ é a distância euclidiana entre o ponto x e o *cluster* c. Após o cálculo efetuado pela medida de proximidade, o centróide será atualizado com a Equação 19:

$$c_j = \frac{1}{m_j}\sum_{x \in c_j} x \tag{19}$$

O algoritmo é finalizado quando um certo critério de parada é encontrado, podendo ser o número de iterações, ou caso não ocorra mudanças nas posições dos centróides. Desse modo, dado um conjunto x de amostras, onde o objetivo é classificar os dados em k *clusters*, o algoritmo tende a minimizar uma função de erro, tal como o erro médio quadrático (Equação 20).

$$E = \sum_{j=1}^{k}\sum_{i=1}^{N}\|x_i - c_j\|^2 \tag{20}$$

Em que, k representa o número de *clusters*, N números de instâncias de dados, x_i instância de dado e c_j é o centro de j grupos.

2.7 Testes Estatísticos

Testes estatísticos são procedimentos que possiblitam avaliar, com auxílio de uma amostra se determinadas hipóteses podem ser rejeitadas ou não (FARIA, 2017). Uma

hipótese estatística é considerada uma suposição que pode ser verdadeira ou não referente a uma ou mais populações. A Hipótese nula (H_0) ou hipótese de igualdade consiste em afirmar que os parâmetros ou características matemáticas de duas ou mais populações são idênticos (FARIA, 2017). Em contrapartida, a hipótese alternativa (H_1) necessariamente, difere de H_0. Com isso, os testes estatísticos tem a capacidade de apontar quais resultados de um experimento podem levar à rejeição de H_0 a um nível de significância (α) pré-estabelecido. O Nível de significância (α) é um limiar de confiança que informa se vamos ou não rejeitar H_0 (FARIA, 2017).

Na literatura estão disponíveis diversos tipos de testes estatísticos que devem ser escolhidos de acordo com uma análise prévia dos dados e da quantidade de grupos avaliadas (FARIA, 2017), (BARBETTA; REIS; BORNIA, 2004). Estes testes são divididos em duas categorias: Testes paramétricos e não paramétricos. Testes paramétricos validam hipoteses sobre parâmetros específicos , tais como média, desvio padrão ou proporção, já os não paramétricos testam hipóteses sobre parâmetros, distruibuições ou classes de amostras relacionadas ou não (SIEGEL; JR, 1975). A Tabela 4 apresenta uma relação de testes paramétricos e não paramétricos.

Testes paramétricos	**Testes não paramétricos**
Teste t para 1 amostra	Teste do Sinal para 1 Amostra, teste de Wilcoxon para 1 Amostra
Teste t para 2 Amostras	Teste de Mann-Whitney
ANOVA com um Fator	Kruskal-Wallis, Teste de Mood para a Mediana
DOE fatorial com um fator e um bloco	Teste de Friedman

Tabela 4 – Relação de testes paramétricos (médias) e não paramétricos (medianas). Tabela adaptada de (MINITAB, 2019).

Na presente dissertação, o teste *Kruskal-Wallis* (KRUSKAL; WALLIS, 1952) foi utilizado para validar as comparações efetuadas em relação ao desempenho dos algoritmos bio-inspirados e suas versões com pré-processamento por meio do *software IBM SPSS* (IBMCORP, 2011).

Antes da aplicação do teste não paramétrico *Kruskal-Wallis* foram realizados testes com as amostras para identificar qual tipo de teste seria correto. Nesses testes foi descartado o uso do teste Análise de Variância (*ANOVA a um fator*) (HEIBERGER; NEUWIRTH, 2009). Onde, este teste exige que duas premissas sejam atendidas, como: distribuição normal dos resíduos e homogeneidade de variância. Para avaliar essas duas premissas foi usado um teste de *Levene* (LEVENE, 1960) que verifica se há homogeneidade de variância, ou homocedasticidade, e para verificar a normalidade dos resíduos foi utilizado o teste de *Shapiro-Willk*, cuja hipótese H_0 é que os dados seguem uma distribuição normal (SHAPIRO; WILK, 1965). Ressalta-se que as amostras analisadas são de grupos independentes, ou seja, em cada grupo os participantes são diferentes. Neste trabalho, cada grupo corresponde a um algoritmo e os casos correspondem as execuções dos algoritmos.

2.7.1 *Kruskal-Wallis*

O *Kruskal-Wallis* é um método não paramétrico aplicado em contextos de teste de hipóteses, os quais envolvem dois ou mais grupos. Tal método é equivalente ao *ANOVA a um fator* (Tabela 4), sendo aplicado quando as premissas necessárias para uso do *ANOVA* não são satisfeitas. Desse modo, a significância estatística é realizada atráves das medianas dos grupos, ao contrário do *ANOVA a um fator*, que realiza a avaliação sobre as médias dos grupos. O teste retorna um *valor-p* que permite verificar a diferença entre os grupos a um nível de significância α.

No cálculo do teste *Kruskal-Wallis* cada observação é substituída por um posto (representação em nível ordinal). Dessa forma, os *scores* das k amostras combinadas são dispostos em uma única serie de postos (VIALI, 2008). Os postos são atribuídos de acordo com os *score*, onde para o menor é atribuído o posto 1, para o seguinte posto 2 e assim sucessivamente até o maior posto que é n (número total de observações independentes nas k amostras) (VIALI, 2008).

Com isso é realizada a soma dos postos em cada amostra. O cálculo testa se as somas apresentam diferenças entre si. Se H_0 é verdadeira, então H tem distribuição qui-quadrado com $gl = k-1$, considerando que os tamanhos das k amostras não sejam muito pequenos. A Equação 21 apresenta o cálculo da estatística de teste, onde H é uma medida de variância das somas dos postos (VIALI, 2008).

$$H = \frac{12}{n(n+1)} \sum_{j=1}^{k} \frac{R_j^2}{n_j} - 3(n+1) \tag{21}$$

- k = número de amostras;
- n_j = número de elementos na amostra j;
- R_j = soma dos postos na amostra j;
- $n = \sum n_j$ = número total de elementos em todas as amostras combinadas.

H assume um valor pequeno quando as populações são semelhantes, do contrario, H tende a assumir um valor maior. Desse modo, a distribuição de H sob H_0 equivale a qui-quadrado com $(k-1)gl$, em que gl é o grau de liberdade, ou seja, número de determinações independentes menos o número de parâmetros avaliados na população.

O *valor-p* é dado pela conversão de H em uma probabilidade condicional representada por $P\left[\chi^2_{(k-1)} > H\right]$, em que χ^2 representa a tabela qui-quadrado. Sendo o *valor-p* menor que o nível de significância α, H_0 é rejeitada.

Através do *Kruskal-Wallis*, não é possível identificar os grupos que apresentam diferença significativa quando a hipótese nula é rejeitada. Desse modo, é frequentemente seguido de um *post-hoc* em que cada par de grupos é submetido a um processo de comparação para que seja realizada uma análise do padrão de diferença em suas médias. Então,

um Teste de *Dunn-Bonferroni* (DUNN, 1964) é realizado entre cada par de grupos. Tal teste funciona como o *Teste de Tukey* (KESELMAN; ROGAN, 1977) no *ANOVA a um fator*.

Capítulo 3

Trabalhos Correlatos

Neste capítulo serão apresentados os trabalhos que foram utilizados como base teórica para o desenvolvimento da proposta dessa dissertação. São abordados trabalhos referentes à aplicação de algoritmos bio-inspirados ao mTSP, técnicas para balanceamento das rotas, além de sistemas para escalonamento de rotas que serviram como base para o método proposto nessa dissertação.

3.1 Minimização e Balanceamento das Rotas

O trabalho de (ALVES; LOPES, 2015) apresenta o desenvolvimento de dois GAs para resolver o mTSP. Os algoritmos realizam a minimização da distância total viajada pelos caixeiros (*salesmen*) e fazem o balancemento de suas rotas. Os autores desenvolveram e avaliaram duas abordagens em conjunto com uma função *fitness* proposta, um GA multi-objetivo (SPEA2) e um GA mono-objetivo com uma função *fitness* que combina dois objetivos. Diferentes métodos de cruzamento e seleção foram testados para as cinco instâncias do mTSP definidas em uma aplicação desenvolvida no trabalho. Como resultado é apresentado que o GA multi-objetivo gerou soluções com melhor balanceamento das rotas para os caixeiros.

O trabalho (XU et al., 2018) apresenta uma abordagem mTSP em duas fases. O trabalho introduz o algoritmo *K-means* para criar grupos de cidades com base na proximidade entre elas. Além disso, foi considerado como critérios de minimização o balanceamento das rotas entre os caixeiros e a distância total. Assim, foi proposto um método de balanceamento, que realiza a minimização e balanceamento das rotas dos caixeiros. Tal método limita a capacidade de cada grupo pelo cálculo da divisão do número de cidades pelo número de grupos (caixeiros). Portanto, cada grupo tem um limite máximo de cidades. Após a aplicação do método de balanceamento utilizando *K-means* e restrição de grupos é iniciada a segunda fase, onde, cada grupo é apresentado como entrada para um GA de forma independente. Os resultados apresentados demonstram que o método proposto alcançou um bom balanceamento, juntamente com a minimização da distância.

Considerando o mTSP como um sistema multiagente, o trabalho (YANG; SZETO, 2019) teve como objetivo simplificar e reduzir o custo computacional enquanto resolvia o mTSP. Para isso, as cidades do problema foram divididas em N grupos usando o *K-means*. Após a divisão das cidades em grupos, o mTSP é decomposto em um conjunto de N TSP, com cada grupo tendo um número máximo de cidades baseado na restrição que é a divisão do número de cidades pelo número de caixeiros. Essa restrição tem como objetivo balancear as cidades entre os N grupos. Para cada grupo, um GA é executado com o operador *2-opt* na busca por uma rota ideal. O desempenho e a complexidade do método foram avaliados mostrando a viabilidade da aplicação do *K-means* na divisão de grandes mapas.

O problema de minimização de equipes de inspeção em linhas de transmissão de energia é resolvido em (HU et al., 2020). Dado as linhas de transmissão de destino e as horas de trabalho restritas para cada inspetor, o objetivo principal é minimizar e equilibrar as horas de trabalho de cada equipe de inspeção com o número mínimo de equipes. Como as tarefas de inspeção podem ser realizadas por Veículos Aéreos Não Tripulados (VANTs), no trabalho foi investigado o tempo gasto sobre o caminho ideal definindo o tempo de inspeção como uma constante. O problema de menor caminho de inspeção com várias equipes de inspeção foi tratado como uma instância do mTSP. Para equilibrar as horas de trabalho e mostrar o caminho quase ótimo das equipes de inspeção, foi proposto um algoritmo *K-means* aprimorado. No algoritmo é mantido a fase de inicialização do *K-means* tradicional, onde são selecionados k postes do conjunto de postes de forma aleatória, e os demais postes são atribuídos aos k conjuntos, k representa o número de equipes de inspeção. Posteriormente, o tempo total gasto e minimização dos caminhos de inspeção de cada equipe é computado pelos algoritmos GRASP, ACO e algoritmo de recozimento simulado (do inglês, *simulated Annealing* (SA)). Os experimentos foram conduzidos com exploração de duas linhas reais de transmissão de energia em uma província na China. Os resultados comparativos demonstram que o algoritmo SA apresentou desempenho superior ao ACO e GRASP.

3.2 Algoritmos Modificados para o mTSP

Em (BOLANOS et al., 2016), um GA modificado denominado *Modified Chu-Beasley Genetic Algorithm* (MCBGA) é aplicado ao mTSP. A principal característica do algoritmo é a construção de uma população inicial com boa qualidade e a implementação de três operadores de busca local, facilitando a exploração de boas regiões no espaço de busca. O algoritmo é dividido em duas etapas. Na primeira etapa, o MCBGA produz uma população inicial completamente diversa considerando três heurísticas para o TSP. Assim, a população é construída com *Nearest Neighbor Heuristics* (GUTIN; YEO; ZVEROVICH, 2002) para 30% da população, *Christofides heuristic* (CHRISTOFIDES,

1976) para outros 30% e o restante da população com a heurística definida em (LIN; KERNIGHAN, 1973). Na segunda etapa, cada solução da população foi dividida aleatoriamente em grupos de acordo com o número de caixeiros e submetida ao GA. Desse modo, o mTSP foi transformado em n TSP, sendo n o número de grupos criados com a divisão aleatória de cada solução. Como experimentos, os autores conduziram uma análise comparativa com os algoritmos ACO apresentado em (JUNJIE; DINGWEI, 2006) e o MGA apresentado em (SEDIGHPOURA; YOUSEFIKHOSHBAKHT; DARANI, 2011). Os resultados apresentados mostram que o MCBGA obteve os melhores resultados para as seis instâncias utilizadas no experimento.

Um ACO modificado é aplicado ao mTSP em (LATAH, 2016). O algoritmo *K-means* foi utilizado para agrupar as cidades do problema em grupos de acordo com a adjacência entre as cidades. O número de grupos foi definido de acordo com o número de caixeiros. A modificação proposta no ACO foi conduzida após a atualização do feromônio com um cruzamento de dois pontos. Após o agrupamento das cidades foi executado em cada grupo o ACO modificado. Além disso, foi realizada uma comparação do algoritmo com as versões clássicas do ACO e GA. Os resultados dos experimentos mostram que o ACO modificado superou o ACO em sua implementação tradicional e apresentou resultados competitivos com GA com implementação típica.

Em (SATHYA; MUTHUKUMARAVEL, 2017), é apresentado o desenvolvimento de um GA e um ACO aplicado ao problema do caminho mais curto, do inglês (*Shortest Path* (SP). O problema foi modelado como mTSP, e para sua resolução um método dividido em duas fases foi aplicado. Na primeira fase, o algoritmo *K-means* foi usado para agrupar as cidades de acordo com a adjacência. Na segunda fase, cada grupo de cidades foi apresentado como entrada para um GA e um ACO conduzirem um processo de otimização das rotas de cada grupo. Para avaliar os algoritmos foi usado um mapa com 180 cidades, e o número de caixeiros definido como seis. Os resultados mostram que o ACO obteve um melhor desempenho na minimização das distâncias.

Em (MA et al., 2019) é proposto um algoritmo de otimização coordenada que combina um GA com *K-means* aplicado ao problema de atribuição e escalonamento da ordem de execução de tarefas para múltiplos Veículos Aéreos Não Tripulados (VANTs). O objetivo principal é determinar de forma efetiva a quantidade de VANTs que satisfazem uma restrição de tempo de missão, e encontrar a melhor ordem de voo para as tarefas de cada VANT, simultaneamente. O problema é abordado como uma instância do mTSP e resolvido em dois procedimentos. Inicialmente, o conjunto de tarefas é dividido em grupos supondo um número de *clusters* para o algoritmo *K-means*. São realizados vários agrupamentos variando o número de *clusters*, assim o agrupamento que apresenta menor custo de tempo tem o número de *clusters* utilizado como número de VANTs. Para cada grupo de saída do *K-means* é atribuido um VANT. O segundo procedimento é o escalonamento da ordem de execução das tarefas para cada grupo com o GA. Após a realização deste pro-

cedimento é avaliada a distância total dos VANTs e se a restrição de tempo foi satisfeita. Caso a restrição de tempo não seja satisfeita, retorna-se ao primeiro procedimento onde é suposto um número de *clusters* e execução do *K-means*. Assim, o mTSP é simplificado ao TSP. Os resultados comparativos e de simulação demonstram que o algoritmo é eficaz para resolver este tipo de problema de atribuição de tarefas.

3.3 Sistema de Escalonamento de Rotas para Robôs

Em (ALVES, 2017) é apresentado um sistema centralizado para coordenação de uma equipe de robôs móveis em ambientes internos. No sistema é possível selecionar um mapa, selecionar o depósito de saída e retorno dos robôs e os locais de visita. As trajetórias são definidas com a realização de um planejamento de caminhos com o algoritmo A*, e o escalonamento das rotas para os robôs são definidos com os algoritmos GA mono-objetivo e GA multi-objetivo apresentados em (ALVES; LOPES, 2015).

Na presente dissertação é proposto o uso do *K-means* como um pré-processamento para geração da população inicial para os algoritmos bio-inspirados GA e ACS. Comparações do GA mono-objetivo de (ALVES; LOPES, 2015) são realizadas com os algoritmos bio-inspirados PSO, ACS e dois outros algoritmos propostos com a junção do *K-means* com GA e ACS. Os algoritmos foram implementados no sistema de escalonamento detalhado em (ALVES, 2017), o qual foi adaptado para o escalonamento de rotas para veículos. Outro aspecto importante que vale ser ressaltado é que esta abordagem difere-se dos trabalhos correlatos, onde esses trabalhos utilizam o *K-means* para agrupar as cidades em n grupos, onde n representa o número de caixeiros, decompondo o mTSP em n TSP. Desse modo, para cada grupo é necessário executar um algoritmo bio-inspirado. Já a abordagem descrita nesta dissertação utiliza o *K-means* para criar bons indivíduos, sendo o foco na modelagem e construção de uma população inicial mTSP e execução única dos algorimos bio-inspirados para o problema. A Tabela 5 apresenta uma comparação entre os trabalhos correlatos e a proposta desta dissertação.

Referência	**Algoritmos**	**Balanceamento**	**Experimentos**	**mTSP decomposto em TSP**
(ALVES; LOPES, 2015) e (ALVES, 2017)	mono-GA e multi-GA	função *fitness*	Simulação em sistema de escalonamento de rotas.	-
(XU et al., 2018)	GA	K-means com restrição nos grupos.	Simulação em aplicativo móvel.	Execução do algoritmo em cada grupo de saída do k-means.
(YANG; SZETO, 2019)	GA com 2-opt	K-means com restrição nos grupos.	-	Execução do algoritmo em cada grupo de saída do k-means.
(BOLANOS et al., 2016)	MCBGA	-	6 instâncias TSPLIB.	Execução do algoritmo em cada grupo gerado com a divisão aleatória da solução de acordo com número de caixeiros.
(LATAH, 2016)	ACO com inserção de operador de cruzamento de dois pontos.	-	5 instâncias TSPLIB.	Execução do algoritmo em cada grupo de saída do k-means.
(SATHYA; MUTHUKUMARAVEL, 2017)	GA e ACO	-	Mapa com 180 cidades e 6 caixeiros.	Execução do algoritmo em cada grupo de saída do k-means.
(MA et al., 2019)	GA	-	Simulação baseada em instância com 100 tarefas	Execução do algoritmo de otimização em cada grupo de saída do k-means.
(HU et al., 2020)	GRASP, ACO e SA	K-means aprimorado	Duas linhas reais de transmissão de energia	Execução do algoritmo de otimização em cada equipe de inspeção.
Nossa proposta	mono-GA, PSO, ACS, k-means-GA e k-means-ACS	função *fitness*	Simulaçao em sistema de escalonamento de rotas e estudo de caso em ambiente real.	-

Tabela 5 – Comparação entre os principais trabalhos correlatos a proposta de trabalho.

Capítulo 4

Materiais e Métodos

A tarefa de otimização de rotas para veículos pode ser vista com uma instância do mTSP, em que cada carro é um caixeiro e locais de entrega são cidades. Neste capítulo é apresentado a abordagem que trabalha para minimização da distância de uma frota e balanceamento das distâncias viajadas por cada carro.

4.1 Escalonamento com Algoritmos Bio-Inspirados

Os algoritmos bio-inspirados apresentam uma arquitetura genérica. Para resolver um problema usando esta arquitetura, alguns componentes devem ser projetados considerando o contexto do problema. Como componentes da arquitetura, temos a estrutura do indivíduo e função de aptidão nos algoritmos GA, PSO e ACS. Outros componentes são mais específicos: cruzamento e mutação para o GA, no PSO temos métodos de transposição e adição da velocidade da partícula, e regras de transição de estado que determinam a construção das rotas para o ACS. Esta seção descreve como estes componentes foram definidos para os algoritmos e como cada algoritmo foi implementado para resolver o mTSP.

4.1.1 Estrutura do Indivíduo

O mTSP estende o tradicional TSP para um modelo multiagente, em que vários caixeiros podem ser usados para visitar n cidades descritas no problema. Assim, cada cidade deve ser visitada apenas uma vez por um dos m caixieros. Para indicar as posições das cidades em um plano ou um mapa, eles devem ter coordenadas. As coordenadas são essenciais principalmente para estimar a distância do caminho entre um par de cidades. Assim sendo, um indivíduo pode ser modelado como a solução apresentada na Figura 17.

O tamanho do indivíduo é igual ao número de cidades, como todas elas devem ser visitadas apenas uma vez. Cada coluna de um indivíduo relaciona um caixeiro com uma

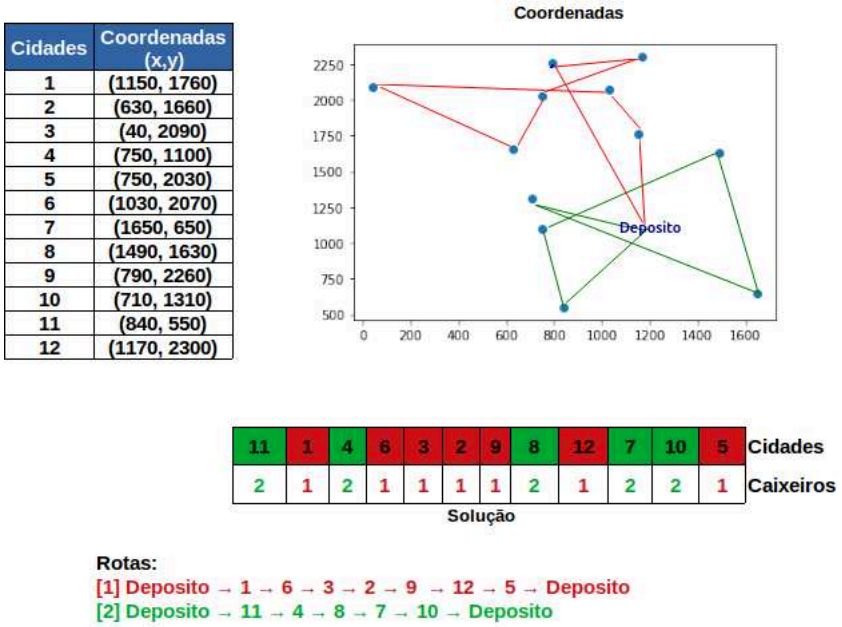


Figura 17 – Modelagem solução mTSP. O indivíduo é a solução do problema. Este cenário mostra um caso em que dois caixeiros devem visitar doze cidades e retornarem ao depósito.

cidade que deve ser visitada. No caso da solução apresentada na Figura 17 são dois caixieros sendo representados pelos números 1 e 2 na segunda linha da estrutura. A ordem das cidades em um indivíduo é também essencial, pois mostra a sequência em que os caixeiros irão realizar as suas visitas. Na primeira linha da estrutura estão as cidades relacionadas a cada caixeiro, formando assim a sua rota individual. A definição do depósito no mapa de coordenadas é única para todos os caixeiros definidos no problema (mTSP com depósito único).

4.1.2 Malha de Tráfego e Grafo Completo

Diferentemente do exemplo da Figura 17, onde os caminhos entre os locais são representados por linhas retas em um plano, o escalonamento de rotas para veículos em mapas de estradas requer o cálculo de caminhos viáveis. Portanto, antes de escalonar as rotas é criado um matriz de caminho P usando o algoritmo A* para todos os pares de locais, incluindo a localização do depósito, conforme ilustrado pela Figura 18.

Assim, a dimensão de P é $\|L\| + 1 \times \|L\| + 1$ em que L é a lista de locais a serem visitados. Para otimizar o cálculo, os elementos da diagonal principal P são definidos como vazios, o que corresponde a caminhos onde a origem e o destino são a mesma localização. Além disso, quando o elemento P_{ij} é calculado, o que corresponde ao caminho de i a j, o caminho inverso é automaticamente armazenado em P_{ji}. Essa matriz representa um grafo completo conectando todos os locais desejados com caminhos reais (malha de tráfego), que são usados posteriormente pelo processo de escalonamento.

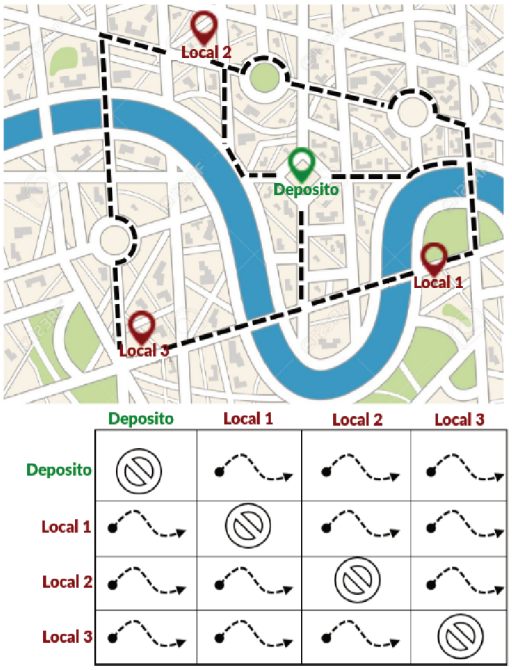


Figura 18 – Matriz de caminho. Essa matriz armazena o caminho calculado com A* com base nos mapas de estradas. Os caminhos são calculados para todos os pares de locais a serem visitados, incluindo o depósito.

4.1.3 Objetivos

O objetivo principal do mTSP é minimizar a distância total percorrida por todos os caixeiros (DAVENDRA, 2010). No entanto, isso pode causar um desequilíbrio na distância percorrida por eles. Isso ocorre quando um caixeiro viaja longe para visitar uma cidade. Como o mTSP visa reduzir a distância total, este caixeiro tende a cobrir todas as cidades ao redor já que enviar outro caixeiro para aquele local, para visitar apenas algumas cidades, pode aumentar a distância total. Como resultado, alguns caixeiros podem viajar muito mais que outros.

Para alguns domínios reais, esse comportamento não é adequado. Por exemplo, em uma empresa de entrega pode haver alguns veículos sobrecarregados fazendo muitas entregas e se deslocando por longas distâncias, enquanto outros permanecem nas proximidades da empresa. Como resultado, alguns clientes precisam esperar mais tempo do que outros para receber suas entregas, o que é um problema para alguns serviços, como a entrega de alimentos. Por isso, é proposto o uso de dois objetivos para o problema. O primeiro é responsável por minimizar a distância total percorrida pelo conjunto de caixeiros, enquanto o segundo visa reduzir a diferença da distância percorrida individualmente por eles.

Dada a matriz de caminho P, a distância da rota percorrida por um caixeiro i é calculada pela soma dos comprimentos de caminho dentro de sua rota R_i:

$$R_i = \sum_{j=1}^{n-1} |P_{j,j+1}|, \forall_j \in Cities_i \tag{22}$$

A distância total percorrida por todos os caixeiros em uma determinada solução é:

$$\Delta = \sum_{i=1}^{m} R_i \tag{23}$$

O equilíbrio é medido pelo desvio padrão dos comprimentos das rotas, onde $\bar{R}$ é o comprimento médio das rotas.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{m}(R_i - \bar{R})^2}{m-1}} \tag{24}$$

4.1.4 GA - Cruzamento e Mutação

O mTSP é um problema de permutação devido as cidades não se repiterem em um cromossomo. Portanto, métodos de cruzamento e mutação devem considerar este contexto. Os métodos *Partially Matched Crossover* (PMX) e *Cycle Crossover* (CX) defindos na subseção 2.5.1 do capítulo 2 são amplamente usados em tais casos.

No PMX, uma parte do cromossomo (através de 2 pontos de corte) é selecionada onde o material genético de ambos parentes será totalmente trocado. Então, algumas mudanças são feitas na descendência para eliminar genes repetidos.

Em relação à mutação, uma simples permutação é realizada entre dois genes aleatórios para garantir que cada cidade apareça apenas uma vez no cromossomo. A Figura 19 apresenta como é realizada a mutação em um indivíduo:

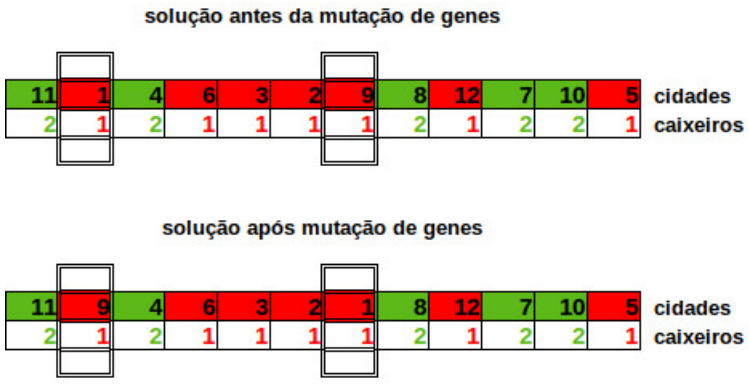


Figura 19 – Exemplo de mutação entre dois genes de um indivíduo.

O Algoritmo 5 apresenta o GA mono-obejtivo adaptado de (ALVES; LOPES, 2015).

No Algoritmo 5, uma população inicial é gerada aleatoriamente com T_p indivíduos e submetida ao processo de avaliação. Cada indivíduo representa um escalonamento de rotas para m caixeiros que passam por n cidades sem repetição.

A função *fitness* F usada neste GA combina os dois objetivos, somando a distância total Δ e o desvio padrão σ, conforme mostrado na fórmula a seguir:

$$F = \Delta + \sigma \tag{25}$$

Algoritmo 5 GA mono-objetivo (ALVES; LOPES, 2015)

Entrada: o tamanho da população T_p, o número de gerações T, taxa de cruzamento r, taxa de mutação m.

Saída: O melhor indivíduo.

```
1: P ← Gerar T_p indivíduos aleatórios
2: for cada p em P do
3:    calcular o Fitness(p)
4: end for
5: t ← 0
6: while t < T do
7:    Selecione r indivíduos de P para adicionar a P_s
8:    Para cada par, (P_s), produzir dois filhos aplicando o operador de cruzamento
9:    Realizar a mutação com a probabilidade m
10:   calcular o Fitness(p)
11:   Selecionar os melhores T_p indivíduos
12: end while
13: return Os melhores indivíduos.
```

Para a próxima etapa são selecionadas r indivíduos da população corrente, para o cruzamento. Para este trabalho, o operador de seleção utilizado foi o torneio estocástico e o de cruzamento usado foi o PMX, com base nos experimentos comparativos de (ALVES; LOPES, 2015). Em seguida alguns novos indivíduos são submetidos ao processo de mutação com prababilidade m. O método de mutação aplica uma simples permutação entre dois genes aleatórios do indivíduo, o que também evita a geração de indivíduos inválidos.

Finalmente, os novos indivíduos são avaliados e inseridos na população atual. Essa população é classificada de acordo com os valores de *fitness*. Os melhores T_p indivíduos em cada iteração são selecionados para formar a nova população que executará novamente os processos de seleção, cruzamento e mutação.

4.1.5 PSO - Velocidade e Posição

Na adaptação do PSO para o TSP a velocidade é uma lista de trocas (*swaps*). Para o mTSP as trocas devem ser feitas para as cidades e caixeiros, e as cidades não podem ser repetidas. A lista de trocas é obtida pela subtração das posições e o movimento é realizado pela adição. Considerando P como uma posição e V como uma velocidade, então P' é a nova posição dada por $P' = P + V$, na qual todas as trocas pertecentes a V devem ser aplicadas. Na Figura 20, a velocidade V é obtida subtraindo uma posição da outra, $V = A - B$. Através dessa diferença obtém-se a lista de trocas, resultando na equação $A = B + V$.

No passo 1 da Figura 20, não foi necessário realizar a troca de caixeiros no indivíduo B, pois o caixeiro na primeira posição do indivíduo B é igual ao caixeiro na primeira posição do indivíduo A. Desse modo, não há necessidade de realizar a troca em B. Sendo

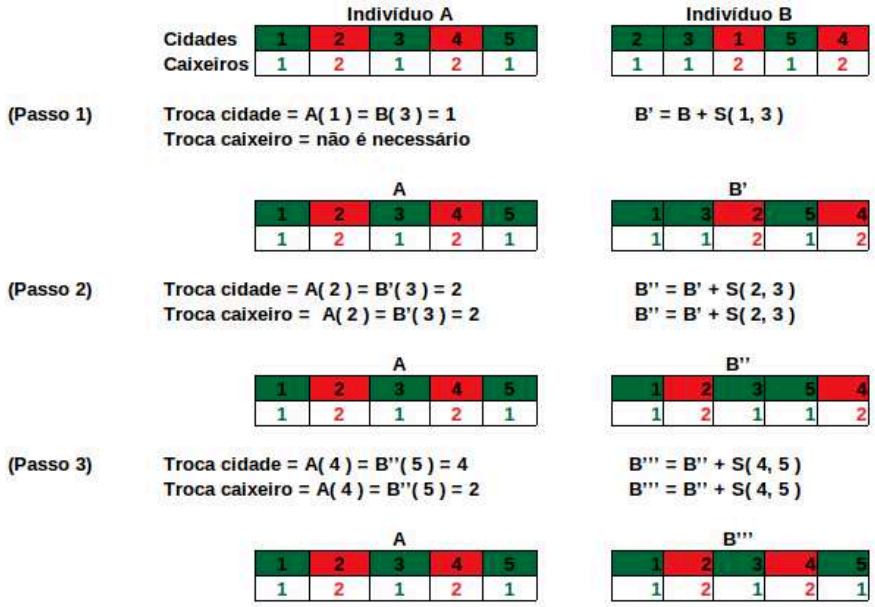


Figura 20 – Exemplo de trocas (*swaps*) nas posições de um indivíduo modelado no PSO. Os blocos pintados de verde representam as rotas para o caixeiro 1 e as rotas pintadas de vermelho, as rotas do caixeiro 2. No final das trocas tem-se $V = [(1,3),(2,3),(4,5)]$.

assim, após a troca das cidades no passo 1, o indivíduo B' na primeira posição sera igual ao indivíduo A na primeira posição.

Esse esquema é aplicado nas partes $(pbest_i - x_i^t)$ e $(gbest - x_i^t)$ da equação de atualização da velocidade apresentada na subseção 2.5.2 do capítulo 2. Nesse caso, podendo o indivíduo A ser considerado $pbest_i$ ou $gbest$ e o indivíduo B, x_i^t.

O algoritmo do PSO discreto não sofre alteração em sua estrutura. Desse modo, a estrutura do nosso algoritmo é semelhante a do Algoritmo 2. A modificação é apenas na velocidade e posição que são conduzidos pelos respectivos cálculos de atualização introduzidos no PSO discreto.

Inicialmente, uma população com um número de partículas (indivíduos) é gerada, sendo a posição das partículas iniciadas de forma aleatória. A posição de uma partícula é representada pela estrutura de indivíduo apresentada na Figura 17 e a velocidade é uma lista de trocas (*swaps*) inicializada como vazia. A avaliação é executada pela função de aptidão apresentada na Equação 25. Após a avaliação, é realizado a atualização da melhor partícula local *pbest* e da melhor partícula global *gbest*. As partículas *pbest* e *gbest* influenciam no cálculo da velocidade de cada partícula, sendo elas que direcionam a atualização das posições. As Equações 9 e 12 são usadas para a realização dos cálculos de atualização de velocidade e posição, respectivamente.

4.1.6 ACS - Construção da Solução

Na modelagem do ACS para o mTSP foi utilizado o paradigma orientado a objetos, onde temos a classe caixeiro composta pelo depósito e uma lista inicialmente vazia, a qual deverá conter a rota para o caixeiro, e a classe solução composta por uma lista de objetos instanciados da classe caixeiro e os atributos para avaliação da solução, desvio padrão e distância total.

Algoritmo 6 ACS- mTSP

Entrada: o tamanho da população T_p, o número de iterações T, influência do feromônio α, informação heurística β, taxa de evaporação local ξ, taxa de evaporação global ρ, valor do feromônio inicial τ_0, parâmetro de escolha RTE q_0.

Saída: A melhor solução.

1: Inicializar a lista $P \leftarrow T_p$ instâncias da classe solução (formigas)
2: Inicializar a tabela de feromônio $\tau_{ij} \leftarrow \tau_0$
3: $f(S_k) \leftarrow \infty$
4: $t \leftarrow 0$
5: **for** cada t em T **do**
6: **for** cada k em P **do**
7: $C \leftarrow$ Instanciar e adicionar os m caixeiros a lista de caixeiros da solução
8: $V \leftarrow$ Gerar a lista de cidades não visitadas
9: **while** $tamanho(V) \neq 0$ **do**
10: $c \leftarrow$ seleciona um m_i caixeiro aleatoriamente
11: $q \leftarrow$ *sorteio()*
12: $j = \begin{cases} \arg\max_{\iota \in S_i^k}\{\tau_{i\iota}[\eta_{i\iota}]^\beta\}, & \text{if } q \leq q0 \\ J, & \text{caso contrário} \end{cases}$
13: $C_c \leftarrow C_c \cup j$
14: $V \leftarrow V - j$
15: $\tau_{ij}(t) \leftarrow (1-\xi)\tau_{ij}(t) + \rho\tau_0$
16: **end while**
17: Adicionar depósito ao final da rota do caixeiro c
18: Avaliar solução construída para $k \leftarrow f(S_k)$ (Passo 6)
19: **if** $S_k < L_{melhor}$ **then**
20: $L_{melhor} \leftarrow S_k$(Passo 7)
21: **end if**
22: **end for**
23: $\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t)$(Passo 8)
24: **end for**
25: **return** S_k

No algoritmo desenvolvido, cada formiga é uma solução, sendo assim uma população é uma lista de instâncias da classe solução. Para cada formiga (solução), são instanciados os m caixeiros. As rotas são construídas com a RTE do ACS para cada caixeiro da solução com base nas cidades disponíveis para visita. No iníco do processo, sorteia-se um caixeiro m_i da lista de caixeiros da solução; e sua rota atual é enviada para a RTE juntamente com

a lista de cidades disponíveis para visita. A partir da última cidade visitada pelo caixiero m_i, a RTE calcula a probabilidade definida na regra do ACS, retornando a próxima cidade a ser visitada pelo caixeiro, e removendo-a da lista de cidades disponíveis. Para cada nova cidade adicionada a rota do caixeiro m_i, ocorre a atualização local na tabela de feromônio entre a cidade de saída i e a nova cidade j, sendo a atualização na posição (i, j) da matriz de feromônio. Esse processo se repete até que a lista de cidades disponíveis fique vazia. Ao final do processo, uma solução é montada e uma estrutura similar a apresentada na Figura 17 é obtida.

Após o processo de construção da solução, ocorre a avaliação definida pela função *fitness* representada na Equação 25 e a atualização da melhor solução local. E por fim, ao final de cada iteração ocorre a atualização global da trilha de feromônio. No ACS, diferentemente do AS, a regra de atualização do feromônio determina que ocorra uma evaporação e depósito de feromônio ao final de cada iteração nas arestas pertencentes a melhor solução encontrada até o momento. O Algoritmo 6 apresenta o pseudocódigo da implementação realizada para o mTSP.

4.2 Algoritmos Híbridos: K-means-GA e K-means-ACS

Além do GA, PSO e ACS, foram desenvolvidas duas versões híbridas entre o algoritmo *K-means* e os algoritmos GA e ACS. Nesta seção são descritos os algoritmos e como cada um foi adaptado ao mTSP.

4.2.1 K-Means como Pré-Processamento

No GA as soluções (indivíduos) iniciais são geradas de forma aleatória, ou seja, em um mapa com sete cidades, as ordens das cidades são embaralhadas em um indivíduo e os m caixeiros adicionados aleatoriamente para cada cidade. Já no ACS é selecionado um m_i caixeiro da lista de caixeiros pertencentes a solução, e para esse é selecionado a próxima cidade a ser visitada de acordo com a RTE. Essas formas tradicionais de gerar a população nos dois algoritmos podem ocasionar um número maior de iterações ou gerações até que cada algortimo alcance a convergência para boas soluções, as vezes nem conseguindo obter tais soluções.

Com o objetivo de melhorar a convergência dos algoritmos, assim como o balanceamento e a qualidade das soluções, foi proposto a utilização de um método de pré-processamento na criação dos indivíduos para os algoritmos bio-inspirados GA e ACS. Tal pré-processamento é realizado com a aplicação do *K-means* no agrupamento das cidades para auxílio na criação dos indivíduos.

O *K-means* é um popularmente conhecido algoritmo de agrupamento de dados com a tarefa de particionar um conjunto de dados em k partições de acordo com a similaridade entre os dados. Nesta dissertação, o *K-means* é aplicado na geração da população inicial para o GA e na criação da lista de vizinhos para a construção da solução no ACS. Desse modo, dois algoritmos são propostos, K-means-GA e K-means-ACS. O Esquema de geração de indivíduos é apresentado na Figura 21.

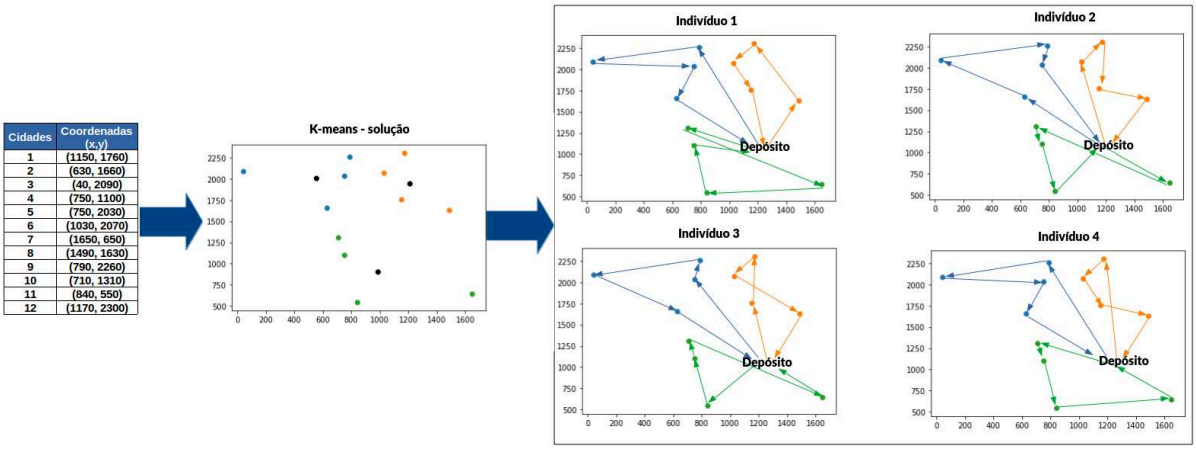


Figura 21 – Esquema de geração de indivíduos com o pré-processamento realizado pelo K-means. Os pontos pretos na solução K-means representam os centróides encontrados pelo algoritmo.

Para geração da população inicial, o *K-means* é executado baseado no mapa de coordenadas com a definição dos pontos de visita. O *K-means* é executado uma vez, onde o número de grupos k é definido de acordo com número de caixeiros. Em seguida, a construção dos indivíduos com os k grupos é iniciada. Para uma população com T_p indivíduos, em cada geração, a sequência das coordenadas será embaralhada dentro dos grupos. Desse modo, não é necessário executar o *K-means* T_p vezes para cada geração de indivíduo. Embaralhando a sequência de coordenadas em cada grupo, é permitido uma maior diversidade de soluções, mantendo a qualidade obtida pelo K-means, como apresentado na Figura 21.

É importante ressaltar que, para uma aplicação em um domínio de vias e estradas, seria interessante usar como medida de distância para o *k-means* o algoritmo A*. Essa implementação foi realizada, porem como os centroides mudam a cada iteração, sendo necessário computar o A* para cada iteração. Portanto, decidimos usar a distância euclidiana como medida de distância com finalidade de termos uma proximidade estimada entre as coordenadas (pontos de visita).

4.2.2 K-means-GA

No K-means-GA, a primeira etapa é a geração de uma populaçao com T_p indivíduos baseados nos resultados do *K-means*. Cada indivíduo é gerado com o embaralhamento da sequência de coordenadas dentro de cada grupo k definido com o número de caixeiros. Esses indivíduos são submetidos a etapa (2) do Algoritmo 5, onde é realizado a avaliação dos indivíduos. A partir dessa etapa continua-se com as etapas do Algoritmo 5.

4.2.3 K-means-ACS

No K-means-ACS, a etapa de geração da população incial com *K-means* é realizada na construção da solução do ACS. Ao contrário do GA, no ACS para cada solução (formiga) k é contruída uma rota baseada na solução resultante do *K-means*. O *K-means* constroi a lista de vizinhos para cada caixeiro m_i e RTE do ACS apenas tem o papel de calcular a probabilidade de escolha da próxima cidade j (local de visita) da lista estipulada pelo *K-means* para o caixeiro m_i. A Figura 22 ilustra o esquema de geração de uma solução com o K-means-ACS.

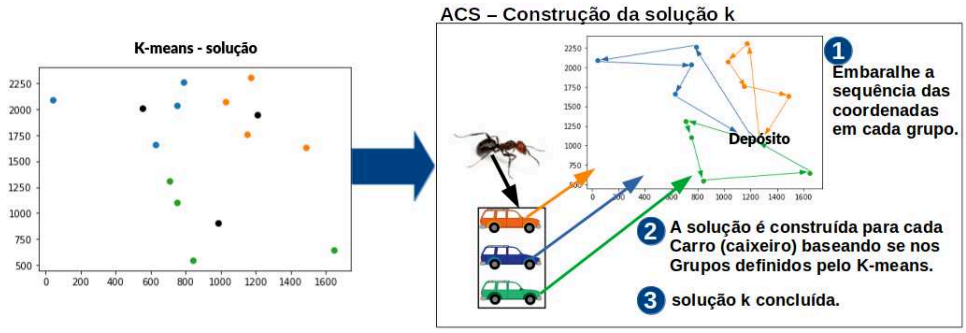


Figura 22 – Esquema de geração de soluções para o ACS com o pré-processamento realizado pelo K-means. Os pontos pretos na solução K-means representam os centróides encontrados pelo algoritmo.

Na Figura 22, o processo de representação do k-means-ACS funciona da seguinte maneira:

- ❏ Na etapa 1, são embaralhados as sequências de coordenadas para cada grupo do *cluster* para a solução k, e criada a lista de vizinhos para cada caixeiro.
- ❏ Na etapa 2, a solução é construída para cada caixeiro segundo a lista de vizinhos e a RTE definida no ACS (Equação 17), sendo que a cada nova cidade adicionada a rota de um caixeiro, ocorre a atualização local da trilha (tabela) de feromônio.
- ❏ Por fim, na etapa 3, a solução k é concluída e submetida a avaliação para atualização da melhor solução local. Este processo se repete até que uma população com T_p soluções (formigas) esteja completa. Ao final de cada iteração ou geração do algoritmo ocorre a atualização global da trilha de feromônio.

4.3 Sistema de Escalonamento de Rotas

Nesta dissertação foi realizado uma adaptação do sistema de coordenação de robôs móveis de (ALVES, 2017) para o problema de escalonamento de múltiplos veículos com modelagem mTSP. Além da adaptação foram inseridos os algoritmos PSO, ACS, K-means-GA e K-means-ACS de modo a poder realizar uma comparação com o GA mono-objetivo apresentado no trabalho anterior. Nesta seção será detalhado o sistema e adaptação para a tarefa de coordenação de uma frota de veículos.

Os experimentos apresentados nesta dissertação foram executados no sistema, tendo como entradas os mapas do ambiente de simulação do CaRINA II. Para isso, foram realizadas as seguintes alterações:

- ❏ Modificações na interface gráfica, com o aumento da tela de execução e botões.
- ❏ Inserção de 2 mapas virtuais do ambiente de simulação do CaRINA II.
- ❏ Alteração da implementação do A*, possibilitando a execução nos *pixels* das vias dos mapas virtuais do ambiente CaRINA II.
- ❏ Inserção das implementações dos algoritmos PSO, ACS, K-means-GA e K-means-ACS.
- ❏ Adaptação para impressão dos escalonamentos realizados pelos algoritmos adicionados na interface gráfica.

O processo de escalonamento funciona conforme ilustrado na Figura 23. Primeiro, o usuário procura o arquivo de imagem do mapa clicando no botão *pesquisar*. Em seguida, o painel de execução aparece com o mapa desejado. Ao selecionar a opção depósito, o usuário pode definir a localização do depósito clicando no mapa. Para definir um local a ser visitado, o usuário pode selecionar a opção *local* e configurá-lo clicando também no mapa. Ele pode repetir este passo o quanto for necessário para definir todos os locais desejados. Ao clicar na opção *excluir*, o sistema remove todos os locais do mapa, inclusive o depósito.

Depois disso, é necessário criar a matriz de caminho usada para o escalonamento de rotas. Quando o usuário clica na opção *encontrar caminhos*, o sistema começa a calcular caminhos para todos os locais. O sistema plota cada caminho no mapa assim que ele é calculado pelo algoritmo A*. Uma mensagem aparece para o usuário quando o processo é concluído. A Figura 23 apresenta as etapas necessárias para o processo de escalonamento no sistema de escalonamento adaptado neste trabalho de dissertação.

Para escalonar as rotas usando os algoritmos desenvolvidos, o usuário deve selecionar um dos algoritmos e preencher o número de caixeiros (veículos) a serem usados e clicar em *executar*. O usuário pode acompanhar o progresso no painel de execução, pois o

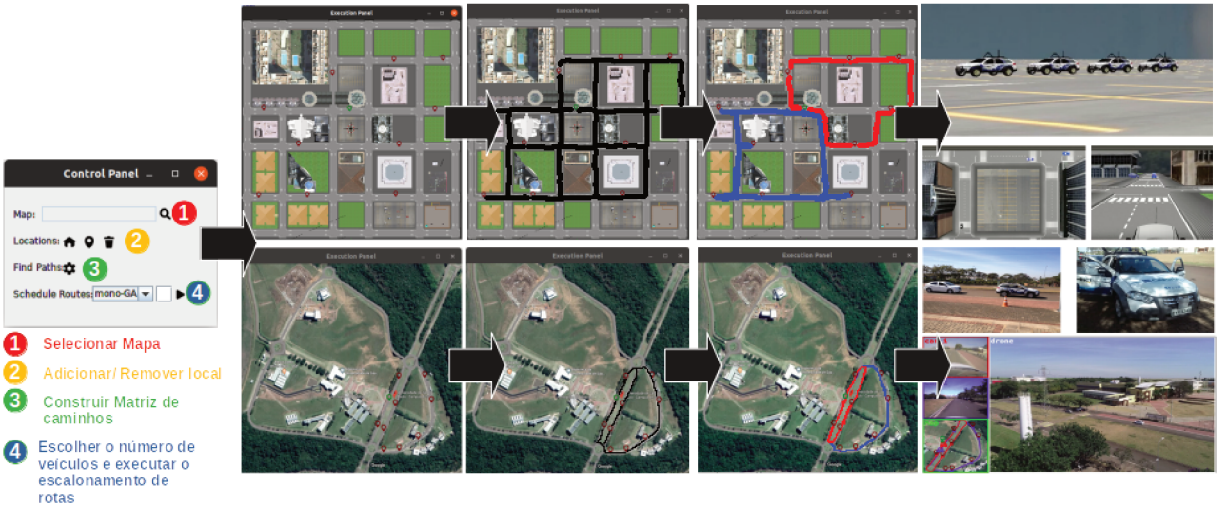


Figura 23 – Escalonamento de rotas. Este processo mostra o fluxo do sistema de escalonamento de rotas: 1) Selecione um roteiro; 2) Defina o depósito e os locais desejados no mapa; 3) Construir gráfico completo e matriz de caminho; 4) Programar rotas para cada veículo usando um algoritmo do sistema; 5) Enviar rotas para os carros.

sistema apresenta a melhor solução encontrada após cada iteração dos algoritmos. Cores diferentes são usadas para representar a rota de cada veículo. Uma mensagem aparece para o usuário quando o escalonamento chega ao fim.

Por fim, as listas de coordenadas a serem visitadas podem ser enviadas para cada veículo. Em nossos experimentos, definimos essas coordenadas para o modelo virtual de carros autônomos em um ambinete simulado, e também em um estudo de caso em um ambiente real. Assim possibilitando a navegação dos veículos com o escalonamento realizado pelo sistema.

Ferramentas de simulação podem ser uma opção para o desenvolvimento de sistemas de veículos cooperativos. Simulações físicas realistas de veículos, sensores e ambientes complexos podem ser criados para reduzir o tempo e os custos dos testes de campo. Nosso sistema de simulação foi implementado usando, como principais ferramentas, *Modular Open Robots Simulation Engine* (MORSE) e o Sistema Operacional de Robôs (ROS).

4.4 Considerações finais

Este capítulo apresentou a descrição de uma abordagem híbrida como uma solução para o mTSP, com aplicação em um sistema centralizado para escalonamento de rotas para veículos autônomos. Inicialmente, o *K-means* é usado como um pré-processamento para gerar rotas que distribuem os locais de visita entre os veículos. Posteriormente, essas rotas são definidas como população inicial para os algoritmos GA e ACS. A escolha em não realização do pré-processamento para o PSO é referente a experimentos anteriores

que demonstraram que o PSO não apresentou uma minimização eficiente, ficando preso em mínimos locais. Isso pode ser associado a dinâmica do algoritmo PSO, que foi fundamentado para problemas contínuos. Na adaptação para problemas discretos, o algoritmo é inferior se comparado com o ACS, que foi fundamentado para problemas discretos e o GA que se adequá em ambos tipos de problemas.

Na implementação do K-means-GA é possível que os pontos de visita mudem de grupo, dado a dinâmica de cruzamento e mutação do indivíduo. Já no ACS, os pontos permanecem agrupados sendo a rota de cada caixeiro definida com com base nos grupos e na RTE. No próximo capítulo são apresentados os experimentos e discussão dos resultados.

Capítulo 5

Experimentos e Análise dos Resultados

Este capítulo apresenta os experimentos realizados e suas configurações. Serão também apresentados os resultados obtidos pelos algoritmos desenvolvidos na pesquisa desenvolvida. Por fim, uma análise dos resultados de cada algoritmo é realizada.

5.1 Método para a Avaliação

Nesta dissertação foram conduzidos quatro experimentos visando o escalonamento de rotas para m veículos com os algoritmos bio-inspirados e com as versões híbridas com *K-means*. Para isso foi realizado uma comparação dos algoritmos no sistema de escalonamento com a configuração de três mapas, onde foram selecionados n pontos de visita. Para condução e validação das hipóteses, foram definidos algumas métodos de avaliação, instâncias com configurações de mapas e parâmetros dos algoritmos.

5.1.1 Métodos Utilizados para Validação de Hipóteses

Para validar os resultados, em cada experimento foi utilizado o teste estatístico *Kruskal-Wallis* para verificar a hípotes H_0 e um pós teste de *Dunn-Bonferroni*, o qual realiza um teste entre cada par de grupos de modo a estimar se há ou não diferença significativa entre esse grupos, considerando um nível de significância $\alpha = 0,05$. Os testes utilizados estão definidos na seção 2.7 e as hípoteses nulas criadas nas subseções dos experimentos 5.3 e 5.4.

5.1.2 Instâncias mTSP

Como a avaliação dos algoritmos é realizada com instâncias mTSP definidas no sistema de escalonamento de rotas (seção 4.3), não foram utilizadas instâncias de *benchmarks* presentes na literatura como as da TSP-lib. Todas as instâncias foram computadas pelo A*, o qual tem o papel de gerar a matriz de caminhos, sendo essa matriz a entrada para os algoritmos.

5.1.3 Parâmetros dos Algoritmos

Para realizar uma comparação com o GA (ALVES; LOPES, 2015), os parâmetros do GA são definifos com os valores que provaram bom funcionamento nos experimentos comparativos de (ALVES; LOPES, 2015). A Tabela 6 apresenta os parâmetros utilizados para o GA e K-means-GA.

Tamanho da população	160
Número de gerações	200
Taxa de cruzamento	60%
Taxa de mutação	30%
Tour	3

Tabela 6 – Parâmetros utilizados pelos algorimos GA e K-means-GA.

Para o PSO foram testados diversas combinações de paramêtros assim como os apresentados em (CLERC, 2010). Contudo, o algoritmo apresentou melhores resultados com a combinação apresentada na Tabela 7.

Tamanho da população	160
Número de gerações	200
Cognitivo e Social	2,0
Peso de inércia	$w_{max} = 0.9$ e $w_{min} = 0.4$

Tabela 7 – Parâmetros utilizados pelo algoritmo PSO.

Os parâmetros utilizados para o ACS foram baseados nos mesmos do trabalho (DORIGO; STUTZLE, 2004), sendo os melhores obtidos experimentalmente na aplicação do ACS em um grande conjunto de instâncias TSP e também sendo o mesmos parâmetros utilizados por (VALLIVAARA, 2008). A Tabela 8 apresenta os parâmetros definidos para o ACS e K-means-ACS.

Tamanho da população	160
Número de iterações	200
Influência do Feromônio	1
Informação Heurística	2
Taxa de evaporação	0,1
Parâmetro de escolha da RTE	0,9
Valor do feromônio inicial	10^{-4}
Feromônio excretado	1

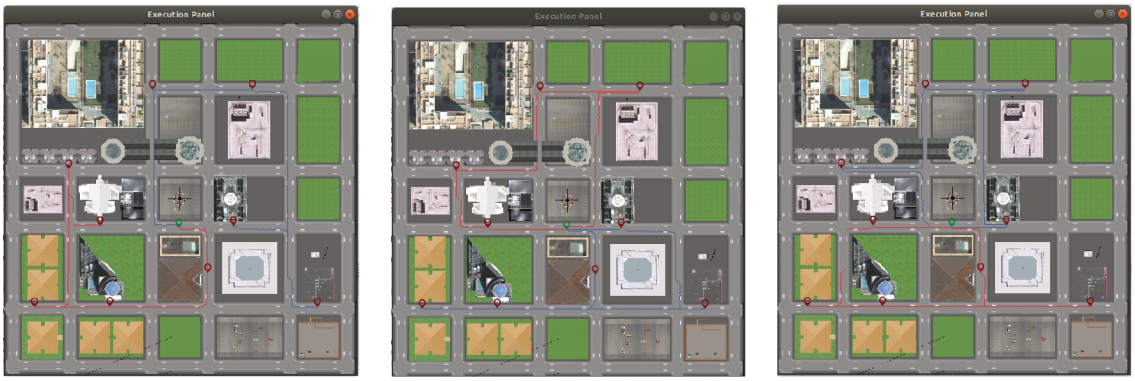
Tabela 8 – Parâmetros utilizados pelos algoritmos ACS e K-means-ACS.

Os experimentos foram conduzidos com o sistema de escalonamento de rotas descrito na seção 4.3, com a definição de pontos de visitas nos mapas. Além disso foi realizada uma simulação utilizando ferramentas *Modular Open Robots Simulation Engine*(MORSE) e *Robot operating System* (ROS) (GÓMEZ et al., 2014), e um experimento real no Campus 2 da Univerdiade de São Paulo, usando um veículo autonômo e um de direção manual.

Durante os experimentos, o sistema foi executado na seguinte configuração: sistema operacional Ubuntu 18.10, processador Intel Core i7 setima geração (2,70GHZ x 4), 15.5 *Gigabytes* de memória RAM e disco rígido de 2 *Terabytes*.

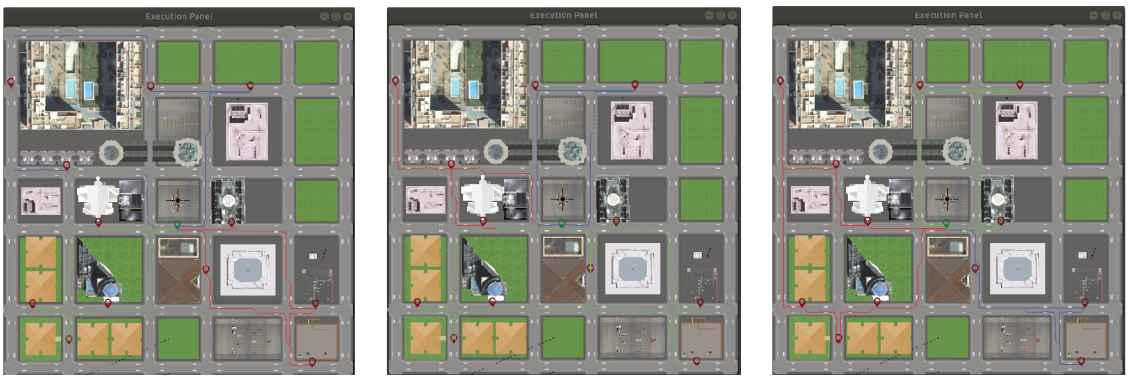
5.2 Experimento 1 - Comparação: PSO, ACS e GA: Cenários com Poucos Carros e Locais de Visita

Neste experimento dois cenários foram definidos no mapa *city*. Os cenários são apresentados nas Figuras 24 e 25. Em cada cenário foram executados os algoritmos GA, PSO e ACS.



GA PSO ACS

Figura 24 – Ambiente de teste usando o mapa da nossa cidade virtual para o cenário 1: 2 carros, 1 depósito e 9 locais. O depósito está marcado em verde e os locais em vermelho. Rotas de carros são pintadas com cores diferentes (vermelho e azul).



GA PSO ACS

Figura 25 – Ambiente de teste no mapa da nossa cidade virtual para o cenário 2: 3 carros, 1 depósito e 12 locais. Depósito está marcado em verde e locais em vermelho. Rotas de carros são pintadas com cores diferentes (vermelho, azul e verde).

As Figuras 24 e 25 apresentam resultados em uma execução de cada algoritmo. Para cada algoritmo foram realizadas dez execuções em ambos cenários. Os parâmetros usados foram os definidos na subseção 5.1.3.

As Tabelas 9, 10 e 11 apresentam os resultados dos escalonamentos dos cenários apresentados nas Figuras 24 e 25 para cada algoritmo.

A Tabela 12 compara a média do número de iterações requeridas para a convergência do GA, PSO e ACS. A Tabela 13 apresenta a otimização da distância total e o balancea-

GA					
Cenário	Carro 1	Carro 2	Carro 3	Δ	σ
2 carros e 9 locais	1266	1027	...	2293	169
3 carros e 12 locais	1413	952	823	3188	310,18

Tabela 9 – Escalonamento com GA: comprimento da rota de cada carro e objetivos. As distâncias são dadas em pixels e podem ser convertidas em metros usando a escala do mapa real.

PSO					
Cenário	Carro 1	Carro 2	Carro 3	Δ	σ
2 carros e 9 locais	1303	1057	...	2360	173,95
3 carros e 12 locais	783	1000	1674	3457	464,62

Tabela 10 – Escalonamento com PSO: comprimento da rota de cada carro e objetivos. As distâncias são dadas em pixels e podem ser convertidas em metros usando a escala do mapa real.

ACS					
Cenário	Carro 1	Carro 2	Carro 3	Δ	σ
2 carros e 9 locais	1131	1203	...	2334	50,91
3 carros e 12 locais	952	1373	869	3194	270,23

Tabela 11 – Escalonamento com ACS: comprimento da rota de cada carro e objetivos. As distâncias são dadas em pixels e podem ser convertidas usando a escala do mapa.

mento das rotas simultaneamente para cada algoritmo em ambos cenários. Os resultados apresentados na Tabela 13 são os melhores obtidos nas dez execuções.

Média das Iterações			
Cenário	GA	PSO	ACS
2 carros e 9 locais	34,6	7,7	19,3
3 carros e 12 locais	63,6	109,7	75

Tabela 12 – Número médio de iterações para atingir a convergência GA, PSO e ACS.

Distância total e balanceamento das rotas						
Cenário	GA		PSO		ACS	
	Δ	σ	Δ	σ	Δ	σ
2 carros e locais	2334	50,91	2334	50,91	2334	50,91
3 carros e 12 locais	3108	303,84	3188	310,18	3194	270,23

Tabela 13 – Minimização de ambos os objetivos, distância total e desvio padrão das rotas.

Com os resultados apresentados obtivemos uma ideia de como conduzir os demais experimentos. Com o número de execuções obtido nesse experimento não é assegurada a relevância estatística, sendo necessário no mínimo 30 execuções de cada algoritmo. Além disso, apenas os dois cenários não são suficientes para comparação entre os algoritmos, sendo necessário a configuração de novos cenários com mais locais de visita e mais carros.

O vídeo deste experimento está disponível online em: <https://youtu.be/WpvASE1zyhw>

5.3 Experimento 2 - Comparação: K-means-GA, K-means-ACS com GA, PSO e ACS: Cenários com Múltiplos Carros e Locais de Visita

Para este experimento em simulação, treze cenários foram definidos com diferentes quantidades de carros e locais de entrega. Cada cenário foi gerado com a seleção de locais de visita em um mapa do sistema, onde cada instância é computada através da matriz de caminhos detalhada na subseção 4.1.2. O mapa utilizado no experimento é apresentado na Figura 26.

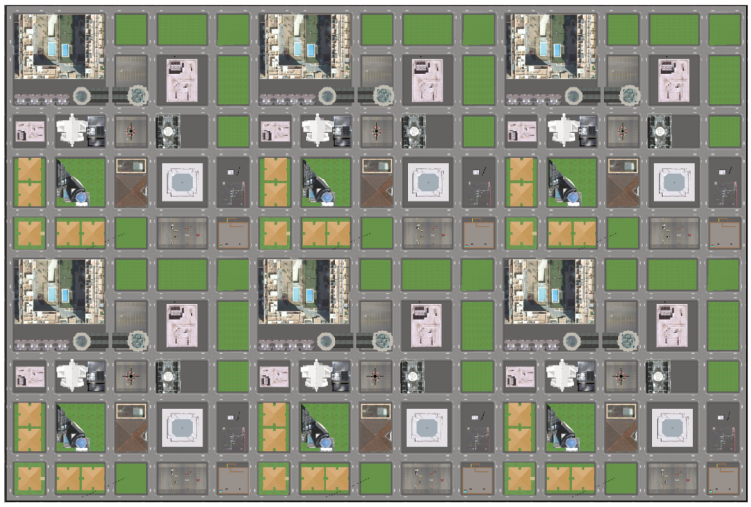


Figura 26 – Ambiente de teste no mapa 2 da cidade virtual.

O mapa da Figura 26 é uma extensão do primeiro mapa apresentado nos experimentos da subseção 5.2. O primeiro mapa tinha uma dimensão menor e por isso notou-se a necessidade de aumentá-lo para adição de mais locais de visita, possibilitando os testes deste experimento. Os cenários foram definidos com a escolha do local de partida e retorno dos veículos (depósito) e os locais de visita dos veículos. Nas Figuras 29, 30, 31 e 32 são mostrados os treze cenários definidos no sistema de escalonamento de rotas.

As configurações dos cenários definidos no mapa da cidade virtual 2, de acordo com as Figuras 29, 30, 31 e 32, são detalhadas na Tabela 14.

Para cada cenário definido, foram realizados escalonamentos com os algoritmos GA, PSO, ACS, K-means-GA e K-means-ACS, sendo efetuadas 50 execuções para cada algoritmo. Desse modo, para cada cenário, foi executado 250 escalonamentos, somando um total final de 3250 execuções. O resultado das execuções é apresentado na Tabela 15.

A função *fitness*, a qual a Tabela 15 apresenta a minimização, é a função descrita na Equação 25. Para cada algoritmo foram selecionados o melhor e o pior valor de aptidão dentre as 50 execuções. Além disso, a média dos valores de *fitness* e o desvio padrão também foram calculados.

Cenário	Configuração
(a)	2 carros e 10 locais de visita
(b)	3 carros e 15 locais de visita
(c)	4 carros e 20 locais de visita
(d)	5 carros e 25 locais de visita
(e)	6 carros e 30 locais de visita
(f)	7 carros e 35 locais de visita
(g)	8 carros e 40 locais de visita
(h)	9 carros e 45 locais de visita
(i)	10 carros e 50 locais de visita
(j)	11 carros e 55 locais de visita
(k)	12 carros e 60 locais de visita
(l)	13 carros e 80 locais de visita
(m)	14 carros e 100 locais de visita

Tabela 14 – Configuração dos cenários apresentados nas Figuras 29, 30, 31 e 32.

Para estimar a significância dos resultados em cada cenário, foi executado o método *Kruskal-Wallis* com nível de significância ($\alpha = 0,05$). Assim, a seguinte hipótese nula H_0 foi criada: Os algoritmos possuem o mesmo nível de desempenho em termos de *fitness* no cenário avaliado.

5.3.1 Avaliação Cenário (a)

Para o cenário (a), o *valor-p* = 0,000 encontrado pelo teste *Kruskal-Wallis* foi menor que o nível de significância ($\alpha = 0,01$). Desse modo, há evidências fortes para sugerir uma diferença entre pelo menos um par de grupos (representados pelos algoritmos). Assim, H_0 foi rejeitada e um teste de comparações múltiplas de *Dunn-Bonferroni* foi realizado para medir a diferença estátistica entre os dez pares de grupos possíveis.

Com base nos resultados apresentados para o cenário aqui avaliado, na Tabela 15 é possível observar que o algoritmo ACS obteve um melhor desempenho em cinquenta execuções em relação ao processo de minimização da função de aptidão (Equação 25). O teste estatístico de múltiplas comparações apresentado na Tabela 20 evidência que não há diferença estatisticamente significativa entre os algoritmos GA e ACS. Desse modo compreendemos que tanto o GA quanto o ACS são adequadas para este cenário.

5.3.2 Avaliação Cenário (b)

Para este cenário, o *valor-p* = 0,000 encontrado pelo *Kruskal-Wallis* foi menor que o nível de significância. Assim, há evidências para sugerir uma diferença entre pelo menos um par de grupos (representados pelos algoritmos). Dessa forma, nesse cenário H_0 também foi rejeitada e novamente o teste de *Dunn-Bonferroni* foi realizado para medir a diferença estatística entre os pares de grupos.

Para este cenário também é apresentado com os resultados da Tabela 15 que algoritmo ACS foi melhor que os demais algoritmos em cinquenta execuções. Contudo, a Tablea 21 apresenta o resultado do teste estatístico que evidência que as medianas de *fitness* entre

Cenário	Algoritmo	Melhor *fitness*	Pior *fitness*	Media	D Padrão	Mediana
2 carros 10 locaís	GA	**2925,83**	3111,59	2929,55	26,27	**2925,83**
	PSO	**2925,83**	3592,61	3192,66	162,16	3158,51
	ACS	**2925,83**	**2925,83**	**2925,83**	**0,00**	**2925,83**
	k-means-GA	**2925,83**	3111,59	3044,72	90,07	3111,59
	k-means-ACS	**2925,83**	3111,59	3041,00	91,08	3111,59
3 carros 15 locaís	GA	**3953,32**	4196,01	3963,95	36,64	**3953,32**
	PSO	3988,48	5707,88	4785,62	304,20	4796,82
	ACS	**3953,32**	**4009,45**	**3961,76**	**9,96**	3959,72
	k-means-GA	3973,14	4698,04	4275,84	235,61	4299,28
	k-means-ACS	3973,14	4617,12	4316,25	204,85	4299,28
4 carros 20 locaís	GA	**6026,60**	6688,24	**6229,37**	175,77	**6184,85**
	PSO	7294,01	8883,62	8355,64	374,71	8391,16
	ACS	6060,96	**6416,10**	6279,99	**69,51**	6278,33
	k-means-GA	6173,94	7553,31	6716,81	423,21	6573,15
	k-means-ACS	6183,23	7966,73	6657,75	441,65	6547,17
5 carros 25 locaís	GA	**7429,69**	9169,72	8202,52	369,86	8192,42
	PSO	11015,94	12972,52	11930,34	458,84	11895,37
	ACS	7686,07	**8329,09**	**8080,90**	**150,62**	**8111,94**
	k-means-GA	8080,11	10075,86	8904,84	432,27	8921,25
	k-means-ACS	7868,69	10100,67	8996,26	468,98	9026,37
6 carros 30 locaís	GA	**9240,22**	11996,49	10331,17	698,06	10188,18
	PSO	13405,41	16910,23	15892,88	608,51	15984,94
	ACS	9429,44	**10478,92**	**9998,96**	**197,50**	**10007,89**
	k-means-GA	9990,74	12075,83	10785,83	477,90	10720,47
	k-means-ACS	9781,14	12165,11	10897,04	574,64	10860,17
7 carros 35 locaís	GA	11055,28	14809,37	12495,03	786,49	12464,26
	PSO	17796,26	21131,67	19514,50	744,73	19582,60
	ACS	**10963,57**	**11894,61**	**11517,36**	**206,97**	**11497,83**
	k-means-GA	11192,60	13451,09	12141,79	564,55	12162,69
	k-means-ACS	11230,23	13840,17	12166,42	595,73	12154,61
8 carros 40 locaís	GA	**12785,44**	18154,30	15367,24	1072,89	15269,10
	PSO	21570,04	25383,06	23807,85	896,76	23960,53
	ACS	13306,51	**14443,05**	**13875,16**	**224,71**	**13881,96**
	k-means-GA	12986,58	15600,32	14352,58	579,36	14294,25
	k-means-ACS	12947,84	16273,53	14511,97	732,04	14555,92
9 carros 45 locaís	GA	15813,65	20286,78	17448,53	1049,42	17412,72
	PSO	25262,51	29072,77	27936,12	767,99	28117,16
	ACS	14889,47	**16087,27**	**15637,93**	**287,32**	**15681,40**
	k-means-GA	14988,96	17603,22	16187,69	562,10	16180,08
	k-means-ACS	**14254,77**	17894,15	16065,20	738,21	15989,96
10 carros 50 locaís	GA	17936,78	25424,75	20742,51	1503,45	20633,73
	PSO	29927,16	33807,63	32231,72	857,00	32319,88
	ACS	16604,97	**17720,00**	**17253,56**	**275,44**	**17331,46**
	k-means-GA	**16152,93**	20685,47	18624,81	1121,24	18622,24
	k-means-ACS	16452,19	21524,24	19126,22	1226,43	19086,01
11 carros 55 locaís	GA	20290,11	28252,24	23203,83	1474,05	22998,57
	PSO	34233,02	38070,09	36503,01	894,03	36569,74
	ACS	18873,22	**20466,42**	20006,13	**302,22**	20011,68
	k-means-GA	18276,86	21534,84	**19756,62**	769,92	**19623,91**
	k-means-ACS	**17933,28**	21927,44	19793,24	862,78	19704,29
12 carros 60 locaís	GA	22838,16	31307,42	26986,15	2019,37	26686,58
	PSO	39593,40	43236,54	41626,36	900,06	41621,23
	ACS	21583,88	**22998,77**	22220,57	**285,68**	22282,83
	k-means-GA	**20472,43**	24199,00	**21945,19**	811,96	**21972,27**
	k-means-ACS	20917,43	24134,55	22333,77	837,99	22363,78
13 carros 80 locaís	GA	31761,18	42083,95	36765,01	2181,92	36665,42
	PSO	54283,33	59134,66	56922,85	1050,89	56941,85
	ACS	24937,75	**27115,26**	26391,50	**408,79**	26463,98
	k-means-GA	23877,73	27618,35	25829,58	879,96	25777,02
	k-means-ACS	**23576,45**	28803,86	**25827,42**	1087,65	**25637,38**
14 carros 100 locaís	GA	41294,79	54287,20	45962,00	2869,84	45233,80
	PSO	65912,16	73511,49	69968,29	1672,09	70227,21
	ACS	27879,07	**30011,62**	29459,72	**428,44**	29601,19
	k-means-GA	**26805,59**	30409,54	28484,69	871,44	**28402,42**
	k-means-ACS	26962,16	30272,42	**28403,41**	838,27	28407,37

Tabela 15 – Minimização de *fitness*: Os resultados apresentam o melhor, pior, a média, desvio padrão e a mediana do *fitness* nas 50 execuções para cada algoritmo em cada cenário.

GA e ACS não apresentam diferença estatística com *valor-p* = ,337. Assim, tanto GA quanto ACS são adequados para o cenário (b).

5.3.3 Avaliação Cenários (c), (d) e (e)

Nos cenários (c), (d) e (e) foram realizadas as seguintes análises estatísticas, cujos resultados mostram que H_0 foi rejeitada com um *valor-p* = $0,000$. Desse modo, há evidências para sugerir uma diferença estatística significativa entre pelo menos um par de grupos. Após a aplicação do teste *Kruskal-Wallis* em cada cenário, o teste de *Dunn-Bonferroni* foi realizado para medir a diferença estatística entre os pares de grupos.

Com os resultados apresentados na Tabela 15 é possível compreender que o GA obteve melhor desempenho em cinquenta execuções para os três cenários, seguido do ACS. Contudo, o resultado do teste estatístico apresnetado nas (Tabelas 22, 23 e 24) evidência que não há diferença estatística significativa entre a distribuição de *fitness* dos algortimos GA e ACS.

5.3.4 Avaliação Cenário (f)

No cenário avaliado, H_0 foi rejeitada com *valor-p* = 0,000. Com isso, há evidências para sugerir uma diferença significativa entre pelo menos um par de grupos. A Tabela 25 apresenta o resultado do teste de comparações múltiplas entre os dez possíveis pares de grupos.

Com base nos resultados apresentados na Tabela 15 é possível observar que o ACS obteve um melhor desempenho em relação a minimização da função *fitness* (Equação 25). O teste estatístico apresentado na Tabela 25 mostra que existe diferença estatística significativa do ACS para os demais algoritmos, comprovando que para a configuração deste cenário o ACS superou os demais algoritmos. Além disso, o teste mostra que não existe diferença estatística significativa entre a distribuição de *fitness* dos algoritmos GA, K-means-GA e K-means-ACS.

5.3.5 Avaliação Cenário (g)

Neste cenário, H_0 foi rejeitada com *valor-p* = 0,000. Desse modo, há evidências para sugerir uma diferença significativa entre pelo menos um par de grupos. O resultado do teste de comparações múltiplas entre os dez possíveis pares grupos é apresentado na Tabela 26.

Avaliando o resultado apresentado na Tabela 26 é possível observar que o GA conseguiu encontrar um melhor valor de *fitness* se comparado aos demais algoritmos nas cinquenta execuções. Porém, novamente é mostrado que o GA não consegue manter um boa qualidade nas execuções apresentando uma média é desvio padrão maiores que os

outros algoritmos, com exceção do PSO. O resultado do teste estatístico de múltiplas comparações entre os dos pares de grupos mostra que apenas o K-means-GA e K-means-ACS não apresentaram diferença estatística significativa. Dessa forma, o GA superou os demais algoritmos para este cenário.

5.3.6 Avaliação Cenário (h)

Para este cenário, o teste *Kruskal-Wallis* rejeitou H_0 com *valor-p* = 0,000. Assim, há evidências para sugerir uma diferença significativa entre pelo menos um par de grupos. O resultado do teste de comparações entre os dez possíveis pares de grupos é apresentado na Tabela 27.

Com base nos resultados apresentados na Tabela 15 é possível observar que o K-means-ACS encontrou um melhor valor de *fitness*, sendo seguido pelo ACS e K-means-GA. O resultado do teste estatístico apresentado na Tablea 27 mostra que não há diferença significativa entre os pares ACS e K-means-ACS, K-means-ACS e K-means-GA. Desse modo, existem evidencias para concluir que o ACS, K-means-GA e K-means-ACS são adequados para este cenário.

5.3.7 Avaliação Cenários (i) e (m)

Nos cenários (i) e (m), H_0 foi rejeitada com *valor-p* = 0,000. Sendo assim, há evidências para sugerir uma diferença significativa entre pelo menos um par de grupos. Os resultados dos testes de comparações entre os dez possíveis pares de grupos em cada cenário são apresentados nas Tabelas 28 e 32.

De acordo com os resultados apresentados na Tabela 15, para estes cenários o K-means-GA conseguiu encontrar um melhor *fitness* entre os demais algoritmos em cinquenta execuções. Contudo, o resultado do teste de estatístico *Dunn-Bonferroni* apresentado nas Tabelas 28 e 32 mostra que não há diferença estatística significativa entre os grupos K-means-GA e K-means-ACS. Desse modo, há evidências para afirmar que os algoritmos K-means-GA e K-means-ACS superaram os demais algorimos nestes cenários.

5.3.8 Avaliação Cenários (j) e (l)

Nos cenários (j) e (l), o teste *Kruskal-Wallis* rejeitou H_0 com um nível de significância *valor-p* = 0,000. Com isso, há fortes evidências para sugerir uma diferença entre pelo menos um par de grupos. Um teste de comparações múltiplas foi aplicado de maneira a medir a diferença estatística entre os pares de grupos de cada cenário. As Tabelas 29 e 31 apresentam os resultados do pós teste aplicado para avaliar se há ou não significância estatística entre entre os grupos representados pelos algoritmos.

De acordo com os resultados apresentados na Tabela 15 para o cenários aqui avaliados, o K-means-ACS conseguiu encontrar um melhor *fitness* em cinquenta execuções, seguido

pelo K-means-GA e ACS. Contudo, o teste de estatśitico (Tabelas 29 e 31) mostram que não existe diferença significativa entre as distribuições de *fitness* dos algoritmos K-means-ACS, K-means-GA e ACS. Assim, com base nos resultados apresentados, há indícios para afirmar que K-means-ACS, K-means-GA e ACS são os mais adequados para os cenários avaliados.

5.3.9 Avaliação Cenário (k)

Na avaliação do cenário (k), o teste estatístico rejeitou H_0 com um nível de significância *valor-p* = 0,000. Dessa forma, há evidências para sugerir uma diferença significativa entre pelo menos um par de grupos. Um pós teste foi aplicado posteriormente para avaliar a diferença entre os pares de grupos. O resultado do teste é apresentado na Tabela 30.

De acordo com os resultados apresentados na Tabela 15 é possível observar que o K-means-GA apresentou uma minimização da função *fitness* mais efetiva em cinquenta execuções, sendo seguido pelo K-means-ACS. Contudo o teste *Dunn-Bonferroni* (Tabela 30) mostra que não há diferença significativa entre K-means-GA, K-means-ACS e ACS. Desse modo, há fortes indícios para afirmar que para o cenário avaliado os algoritmos K-means-GA, K-means-ACS e ACS são os mais adequados.

5.3.10 Análise: Balanceamento de Rotas X Distância Total

A função *fitness* usada pelos algoritmos (Equação 25) combina dois objetivos, sendo distância total Δ e balancemaneto das rotas σ. O propósito principal da função usada é minimizar a distância total viajada pela frota de veículos enquanto reduz a diferença entre as distâncias de cada veículo. A Tabela 16 apresenta a minimização de ambos objetivos decompostos.

Na Tabela 16 é possível observar que com o aumento de veículos e locais de visita, os algoritmos K-means-GA e K-means-ACS produzem soluções mais balanceadas. Isso, devido ao pré-processamento realizado com *K-means* na geração da população dos algoritmos GA e ACS. Por outro lado, podemos também observar que em alguns casos o GA e o ACS obtiveram uma distância total menor enquanto o desvio padrão é mais elevado, isso é atribuído ao fato da construção randômica dos indivíduos. Com isso, para um mapa em que os locais de visita estão próximos pode acontecer que alguns carros fiquem sem locais para visitar, ocasionando uma distância total menor e desvio padrão um pouco mais elevado.

5.3.11 Análise da Convergência dos Algoritmos

Para cada execução, um algoritmo realiza 200 iterações ou gerações. Contudo, o processo de minimização pode ocorrer antes que o número de iterações atinja o valor

Cenário	Algoritmo	Δ	σ	$F = \Delta + \sigma$
2 carros 10 locaís	GA	2853,00	72,83	**2925,83**
	PSO	2853,00	72,83	**2925,83**
	ACS	2853,00	72,83	**2925,83**
	K-means-GA	2853,00	72,83	**2925,83**
	K-means-ACS	2853,00	72,83	**2925,83**
3 carros 15 locaís	GA	3564,00	389,32	**3953,32**
	PSO	3564,00	424,48	3988,48
	ACS	3564,00	389,32	**3953,32**
	K-means-GA	3554,00	419,14	3973,14
	K-means-ACS	3554,00	419,14	3973,14
4 carros 20 locaís	GA	5846,00	180,60	**6026,60**
	PSO	6863,00	431,01	7294,01
	ACS	5268,00	792,96	6060,96
	K-means-GA	6034,00	139,94	6173,94
	K-means-ACS	6044,00	139,23	6183,23
5 carros 25 locaís	GA	6400,00	1029,69	**7429,69**
	PSO	10677,00	338,94	11015,94
	ACS	6546,00	1140,07	7686,07
	K-means-GA	7837,00	243,11	8080,11
	K-means-ACS	7149,00	719,69	7868,69
6 carros 30 locaís	GA	8342,00	898,22	**9240,22**
	PSO	13217,00	188,41	13405,41
	ACS	8539,00	890,44	9429,44
	K-means-GA	9337,00	653,74	9990,74
	K-means-ACS	9224,00	557,14	9781,14
7 carros 35 locaís	GA	10451,00	604,28	11055,28
	PSO	17034,00	762,26	17796,26
	ACS	10016,00	947,57	**10963,57**
	K-means-GA	10635,00	557,60	11192,60
	K-means-ACS	10730,00	500,23	11230,23
8 carros 40 locaís	GA	11917,00	868,44	**12785,44**
	PSO	20806,00	764,04	21570,04
	ACS	12553,00	753,51	13306,51
	K-means-GA	12260,00	726,58	12986,58
	K-means-ACS	12229,00	718,84	12947,84
9 carros 45 locaís	GA	14814,00	999,65	15813,65
	PSO	23456,00	1806,51	25262,51
	ACS	13916,00	973,47	14889,47
	K-means-GA	14582,00	406,96	14988,96
	K-means-ACS	13625,00	629,77	**14254,77**
10 carros 50 locaís	GA	17458,00	478,78	17936,78
	PSO	28966,00	961,16	29927,16
	ACS	15462,00	1142,97	16604,97
	K-means-GA	15552,00	600,93	**16152,93**
	K-means-ACS	16062,00	390,19	16452,19
11 carros 55 locaís	GA	19397,00	893,11	20290,11
	PSO	33061,00	1172,02	34233,02
	ACS	18021,00	852,22	18873,22
	K-means-GA	17731,00	545,86	18276,86
	K-means-ACS	17383,00	550,28	**17933,28**
12 carros 60 locaís	GA	22213,00	625,16	22838,16
	PSO	38783,00	810,40	39593,40
	ACS	20721,00	862,88	21583,88
	K-means-GA	19818,00	654,43	**20472,43**
	K-means-ACS	20405,00	512,43	20917,43
13 carros 80 locaís	GA	30804,00	957,18	31761,18
	PSO	52901,00	1382,33	54283,33
	ACS	24099,00	838,75	24937,75
	K-means-GA	23356,00	521,73	23877,73
	K-means-ACS	22963,00	613,45	**23576,45**
14 carros 100 locaís	GA	40489,00	805,79	41294,79
	PSO	64368,00	1544,16	65912,16
	ACS	27418,00	461,07	27879,07
	K-means-GA	26198,00	607,59	**26805,59**
	K-means-ACS	26462,00	500,16	26962,16

Tabela 16 – Minimização de ambos os objetivos, distância total e desvio padrão das rotas.

definido para cada algoritmo. Desse modo as Figuras 33, 34 e 35 apresentam gráficos *boxplot* com as variações do número de iterações gastos por cada algoritmo até atingir a

convergência em cada execução.

Cada gráfico boxplot tem cinco informações importantes, como valor mínimo, primeiro quartil, mediana, terceiro quartil e valor máximo. Além disso, a representação de *outliers*, quando a observação é maior ou menor que 1,5 vezes a distância interquartílica.

Com os gráficos apresentados é possível compreender a variabilidade de cada observação e notar que na maioria dos cenários o K-means-GA apresenta uma variação menor que suas versões comuns. Nos cenários (a), (b), (c) observa-se que os valores dos algoritmos se diferem em valor máximo, terceiro quartil e mediana, onde os algoritmos K-means-GA e K-means-ACS apresentam melhores variações em relação ao número de iterações necessárias a convergência. Para os demais cenários é possível observar que o K-means-GA consegue alcançar a convergência com menos iterações em relação ao GA, já o K-means-ACS não apresenta uma grande diferença para o ACS.

De modo a estimar a significância dos resultados em cada cenário, foi executado o método *Kruskal-Wallis* com nível de significância ($\alpha = 0,05$). Com isso, uma hipótese nula H_0 foi criada: Os algoritmos apresentam a mesma taxa de convergência no cenário avaliado.

Analisando os resultados dos gráficos *boxplots* e dos testes estatísticos referentes a convergência dos algoritmos no experimento 2, é possível observar que com exceção do cenário (c), os demais cenários apresentam que não existe diferença entre a distribuição do K-means-ACS e ACS. Porem, entre K-means-GA e GA existe diferença significativa. Desse modo, podemos concluir que o pré-processamento com K-means proporcionou uma aceleração na convergência do GA. Para o cenário (c), a tabela 35 mostra que existe diferenças entre as médias dos pares K-means-ACS e ACS e também entre as médias do par K-means-GA e GA. Desse modo, podemos concluir que para esse cenário o pré-processamento com K-means proporcionou uma melhora significativa na aceleração da convergência dos algoritmos GA e ACS.

5.3.12 Considerações Finais do Experimento 2

Com a análise do experimento apresentado nessa seção, temos indícios para concluir que com o aumento de locais de visita, os algoritmos que usam *K-means* como pré-processamento conseguem melhores resultados na minimização. Onde, para os cenários de (a) a (e) os algoritmos GA e ACS são melhores que os demais, apresentando resultados competitivos entre si. No cenário (f) o ACS supera os demais e no cenário (g) o GA. A partir do cenário (h) os algoritmos K-means-GA e K-means-ACS superam os demais apresentando resultados competitivos com o ACS. Desse modo, a utilização do *K-means* como pré-processamento teve um melhor desempenho aplicado ao GA, melhorando a qualidade das soluções produzidas pelo algoritmo. Como o ACS já apresenta boas soluções em quase todos os cenários, a realização do pré-processamento não apresentou mudanças significativas na geração de boas soluções no processo de minimização da função *fitness*.

Isso pode ser relacionado ao fato que o ACS é um algoritmo que foi fundamentado para problemas de busca em grafos, obtendo um bom desempenho em problemas como TSP. O que pode ser observado nos cenários deste experimento, onde o ACS consegue obter melhores soluções conforme ocorre o aumento de carros e locais de visita.

Em relação a convergência dos algoritmos, as análises apresentadas no experimento apresentam evidências para concluir que o pré-processamento realizado com *K-means* para o GA proporcionou uma aceleração na convergência do algoritmo. O K-means-GA obteve uma menor distribuição com os gráficos *box-plots* e o teste estatístico evidencia a diferença entre as médias dos algoritmos. Já para o ACS, o teste estatístico não encontrou diferença significativa entre as médias do K-means-ACS e ACS, em todos cenários. Desse modo, podemos concluir que não houve uma aceleração significativa na convergência do ACS com a aplicação do pré-processamento com *K-means*.

Nesse experimento foi apresentado além dos algoritmos GA, PSO e ACS, os algoritmos híbridos K-means-GA e K-means-ACS. A escolha em não realizar o pré-processamento para o PSO é referente a experimentos anteriores que comprovaram que o PSO não apresenta bons resultados na minimização da função *fitness* apresentada, ficando preso em mínimos locais. Tal experimento variava o número de carros e locais de acordo com número de carros e locais de visita dos cenários (a) ao (g). Ressalta-se que esse experimento não foi adicionado a dissertação porque seria redudante, sendo que os resultados são praticamente os mesmos que os dos cenários apresentados nesta seção. Dessa forma, a escolha para realização do pré-processamento com *K-means*, se deu com a motivação de melhorar os resultados dos algoritmos GA e ACS.

5.4 Experimento 3 - Comparação: K-means-GA, K-means-ACS com GA, PSO e ACS: Cenários com Poucos Carros e Muitos Locais de Visita

Em um ambiente real talvez não seja possível o aumento do número de veículos conforme são aumentados os locais a serem visitados. Esse questionamento é discutido com os resultado deste experimento que realiza comparações entre os algoritmos em cenários com poucos carros e muitos locais de visita (entrega).

Para este experimento quatro cenários foram definidos. As configurações dos quatro cenários são apresentadas nos mapas da Figura 36. Cada cenário foi gerado com a seleção de locais de visita no mapa (Figura 26) e a definição do número de carros no sistema de escalonamento como mostra o processo apresentado na Figura 23.

A configuração dos cenários é expressa da seguinte forma: (a2) configura 2 carros e 55 locais de visita, (b2) configura 3 carros e 60 locais de visita, (c2) configura 4 carros e 80 locais de visita e (d2) configura 5 carros e 100 locais de visita.

Para cada cenário, foram realizados escalonamentos com os algoritmos GA, PSO, ACS, K-means-GA e K-means-ACS, sendo efetuadas 50 execuções para cada algoritmo. O resultado das execuções é apresentado na Tabela 17.

Cenário	Algoritmo	Melhor *fitness*	Pior *fitness*	Media	D Padrão	Mediana
2 carros 55 locaís	GA	15938,97	21845,05	18699,92	1327,26	18778,20
	PSO	26539,09	33528,56	30455,14	1516,05	30587,26
	ACS	10318,02	11217,94	10959,97	194,37	11010,79
	k-means-GA	12078,01	16166,07	13956,71	879,64	13796,23
	k-means-ACS	**10218,58**	**10975,50**	**10685,20**	**147,71**	**10690,16**
3 carros 60 locaís	GA	18182,50	24715,23	21745,82	1534,45	21901,69
	PSO	30953,77	39248,46	35638,92	2027,61	35625,43
	ACS	**12642,00**	**13271,53**	**12959,48**	**156,72**	**12972,50**
	K-means-GA	141647,26	17810,59	15774,39	934,76	15725,36
	K-means-ACS	12833,57	15061,08	14002,52	710,60	14131,46
4 carros 80 locaís	GA	27123,92	36461,20	32306,29	2153,88	32400,45
	PSO	47098,09	55427,03	51873,42	2048,35	51986,46
	ACS	**14954,33**	**15696,10**	**15385,07**	**150,88**	**15395,67**
	K-means-GA	17350,80	22047,46	19519,36	960,58	19425,27
	K-means-ACS	15515,64	17694,11	16685,29	430,81	16721,95
5 carros 100 locaís	GA	36005,68	45177,60	40331,23	2107,05	40310,40
	PSO	59746,80	70556,83	65594,76	2261,29	65578,87
	ACS	17433,84	**18258,37**	**17948,94**	**205,09**	**17983,70**
	K-means-GA	20785,49	25807,12	22586,24	1059,04	22502,20
	K-means-ACS	**17261,12**	19690,91	18974,62	469,41	18999,73

Tabela 17 – Minimização de *fitness*: Os resultados apresentam o melhor, pior, a média, desvio padrão e a mediana do *fitness* nas 50 execuções para cada algoritmo em cada cenário.

De modo a estimar a significância dos resultados em cada cenário, foi executado o método *Kruskal-Wallis* com nível de significância ($\alpha = 0,05$). Assim, a seguinte hipótese nula H_0 foi criada: Os algoritmos possuem o mesmo nível de desempenho em termos de *fitness* no cenário avaliado.

5.4.1 Avaliação Cenário (a2)

No cenário (a2) o teste *Kruskal-Wallis* rejeitou H_0 com um nível de significância *valor-p* = 0,000. Dessa forma, há evidências para sugerir uma diferença significativa entre pelo menos um par de grupos. A Tabela 46 apresenta o resultado do teste de *Dunn-Bonferroni* que avalia se há ou não diferença significativa entre os grupos.

De acordo com os resultados apresentados na Tabela 17 é possível observar que o algoritmo K-means-ACS apresentou melhores resultados na minimização do função *fitness*. Contudo, os resultados do teste estatístico apresentado na Tabela 46 mostram que não há diferença significativa entre K-means-ACS e ACS. Desse modo, os Algoritmos K-means-ACS e ACS foram os melhores para a configuração desse cenário.

5.4.2 Avaliação Cenários (b2) e (c2)

Nos cenários (b2) e (c2) o teste *Kruskal-Wallis* rejeitou H_0 com um nível de significância *valor-p* = 0,000. Dessa forma, há evidências para sugerir uma diferença significativa

entre pelo menos um par de grupos. As Tabelas 47 e 48 apresentam o resultado do teste de *Dunn-Bonferroni*.

Com base nos resultados apresentados na Tabela 17 é possível observar que para estes cenários o ACS apresentou melhores resultados, e como comprovação disso, as Tabelas 47 e 48 apresentam o resultado do teste estatístico que mostra que a distribuição de *fitness* do ACS apresenta diferença significativa para os demais algoritmos. Desse modo, para a configuração desses cenários o ACS superou os demais algoritmos na minimização do *fitness*.

5.4.3 Avaliação Cenário (d2)

No cenário (d2) o teste *Kruskal-Wallis* rejeitou H_0 com um nível de significância *valor-p* = 0,000. Dessa forma, há evidências para sugerir uma diferença significativa entre pelo menos um par de grupos. A Tabela 49 apresenta o resultado do teste de *Dunn-Bonferroni*.

Com base nos resultados apresentados na Tabela 17, é possível observar que o K-means-ACS apresentou melhores resultados quanto a minimização da função *fitness*. Com os resultados do teste estatístico apresentado na Tabela 23 é comprovado a superioridade do K-means-ACS sobre os demais algoritmos, onde, é apresentado que existe diferença com nível de significância entre K-means-ACS e os demais algoritmos.

5.4.4 Análise: Convergência dos Algoritmos

A Figura 37 apresenta a distribuição do número de iterações demandadas por cada algoritmo até atingir a convergência em cada execução.

Visando estimar a significância dos resultados em cada cenário, foi executado o teste *Kruskal-Wallis* com nível de significância ($\alpha = 0,05$). Assim, a seguinte hipótese nula H_0 foi criada: Os algoritmos apresentam a mesma taxa de convergência no cenário avaliado.

Analisando os resultados dos gráficos *boxplots* e dos testes estatísticos referentes a convergência dos algoritmos no experimento 3, é possível observar que para os quatro cenários, o teste *Kruskal-Wallis* rejeitou H_0 com um nível de significância *valor-p* = 0,000. Assim, existem evidências significativas para sugerir diferença entre pelo menos um par de grupos de cada cenário. As Tabelas 50, 51, 52 e 53 apresentam os resultados do teste estatístico que busca avaliar a difernça entre os possíveis pares de grupos.

De acordo com a Figura 37, nos quatro cenários deste experimento é possível observar que existe pouca variação referente ao número de iterações requeridas até a convergência dos algoritmos com e sem pre-processamento com *K-means*. O resultado do teste estatístico comprova tal suposição, onde é apresentado que não há diferença significativa entre as médias dos algoritmos ACS e K-menas-ACS, e GA e K-means-GA. Desse modo, para os quatro cenários, podemos concluir que não houve melhora significativa relacionada a convergência dos algoritmos com aplicação do pré-processamento com *K-means*.

5.4.5 Considerações Finais do Experimento 3

Com a análise do experimento referente a minimização da função *fitness* temos fortes indícios para concluir que o ACS apresenta bons resultados com tais configurações. O K-means-ACS apresentou uma melhor minimização nos cenários (a2) e (d2) porém o teste estatístico evidencia que não diferença estatística significativa entre as médias do K-means-ACS e ACS nestes cenários. O ACS e K-means-ACS obtiveram desempenho superior aos demais algoritmos, porem é importante ressaltar que o pré-processamento aplicado ao GA aumentou bastante o desempenho do algoritmo quanto a minimização do *fitness*. O teste estatístico também comprova que as diferenças das médias do K-means-GA e GA apresentam diferença se comparadas há um nível de significância 0,05. Desse modo, temos evidências para concluir que com a configuração apresentada nos cenários desse experimento, o pré-processamento com *K-means* melhorou consideravelmente o desempenho do GA na minimização do *fitness*. Contudo, para o ACS não houve melhoria significativa. Novamente, o bom desempenho do ACS pode ser relacionado ao fato que o algoritmo foi fundamentado para problemas de busca em grafos, obtendo um bom desempenho em problemas como TSP. O que também pode ser observado nos cenários deste experimento.

Em relação a análise da convergência dos algoritmos nos quatro cenários definidos, temos evidências para concluir que não houve melhoria significativa, ou seja, não houve aceleração da convergência dos algoritmos GA e ACS com a aplicação do pré-processamento com *K-means*. Sendo que, as variações apresentadas na Figura 37 para os algoritmos com pré-processamento não apresentam diferença significativa para suas versões sem pré-processamento.

5.5 Experimento 4 - Estudo de Caso - Cénario: Campus 2 - USP

Além dos experimentos simulados no sistema (seção 4.3), esta dissertação também apresenta um estudo de caso em cenário real. Tal estudo foi conduzido no Campus 2 da Universidade de São Paulo (USP). O Mapa do Campus foi capturado como uma imagem do *Google Maps* (Figura 27) e as vias pintadas com a cor que o sistema reconhece para a realização do planejamento de caminhos. Como experimentos reais são mais difíceis de conduzir, optamos por definir apenas um cenário com 2 carros e 9 locais de visita. Os veículos utilizados no experimento foram o CaRINA II e um Chevrolet/Prisma (Figura 28) , sendo o ultimo de direção manual.

Na Figura 27 é descrito o processo realizado para o escalonamento de rotas para dois veículos no sistema (seção 4.3). Em (a) é apresentado a seleção do depósito e dos nove locais de visita; (b) realização do planejamento de caminhos com A*; (c) escalonamento

(a) (b) (c)

Figura 27 – Processo de escalonamento de rotas no Campus 2 USP. (a) definição dos locais. (b) Planejamento de caminhos. (c) Escalonamento de rotas para 2 carros.

com um dos algoritmos.

O experimento foi dividido em duas etapas, na primeira etapa foram realizados 50 escalonamentos para cada algoritmo no sistema, de modo a verificar qual algortimo apresentava melhores resultados para este cenário. Os resultados dos escalonamentos no sistema são apresentados na Tabela 18, com o melhor, pior, média e o desvio padrão nas 50 execuções para cada algoritmo.

Cenário	Algoritmo	Melhor *fitness*	Pior *fitness*	Media	D Padrão
2 carros 9 locais	GA	**939,73**	955,84	940,70	3,86
	PSO	**939,73**	1092,00	988,70	48,05
	ACS	**939,73**	**939,73**	**939,73**	**0,00**
	k-means-GA	1075,89	1109,96	1081,34	12,62
	k-means-ACS	1075,89	1109,96	1086,79	16,05

Tabela 18 – Minimização de *fitness* - Campus 2.

Na segunda etapa, foi realizado um escalonamento com um dos algoritmos que apresentou melhor minimização na Tabela 18. Com o algoritmo escolhido, foi possível realizar a reprodução no cenário real. A Tabela 19 apresenta o resultado do escalonamento realizado no sistema para o estudo de caso.

GA				
Cenário	Carro 1	Carro 2	Δ	σ
2 carros e 9 locaís	487,00	370,00	857,00	82,73

Tabela 19 – Escalonamento Campus 2: comprimento da rota de cada carro e objetivos. As distâncias são dadas em pixels e podem ser convertidas em metros usando a escala do mapa real.

Após a realização do escalonamento foi conduzido o estudo real no campus 2. Ressalta-se que o sistema (seção 4.3) não fornece uma interface de comunicação com o sistema do CaRINA II e a definição dos pontos no ambiente real foi realizada por observação

no mapa virtual e marcação dos pontos de latitude e longitude no sistema de GPS[1] do CaRINA II. Após isso, usamos o CaRINA II para mapear os locais de visita para o Chevrolet/Prisma, com base nos pontos definidos no GPS. No escalonamento apresentado na Figura 27 imagem (c), o CaRINA II realizou as visitas na rota em azul e os pontos da rota em vermelho foram visitados pelo Chevrolet/Prisma. O estudo real foi realizado com a aplicação do algortimo GA no escalonamento das rotas.

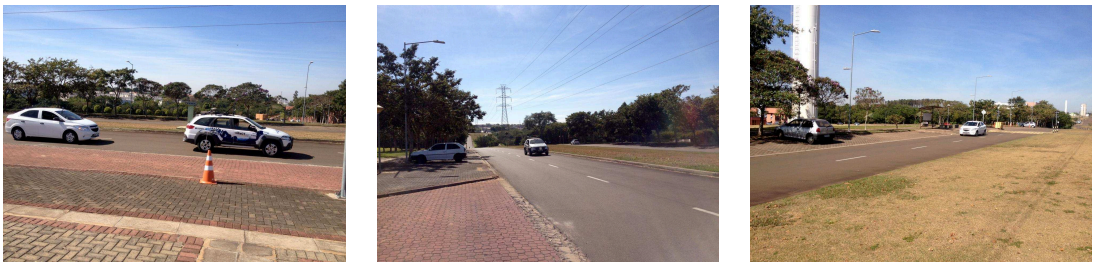


Figura 28 – Estudo de Caso USP: CaRINA 2 e Chevrolet/Prisma.

A condução do estudo de caso foi realizada em conjunto com a equipe do Laboratório de Robótica Móvel (LRM) Instituto de Ciências Matemáticas e de Computação (ICMC) da USP (Figura 38). Para a gravação do estudo, foi utilizado um Veículo Aéreo Não Tripulado (VANT) modelo *Phantom 4*.

Ressalta-se que este estudo de caso teve como objetivo demonstrar a aplicação do sistema de escalonamento de rotas em um experimento real com um veículo autonômo e um veículo de direção manual.

O vídeo deste experimento está disponível online em: <https://youtu.be/Sokr-TxYHfU>

[1] Sistema de Posicionamento Global, consiste numa tecnologia de localização por satélite.

Capítulo 6

Conclusão

Este trabalho apresenta o problema de escalonar rotas para múltiplos carros e locais de visita. O problema pode ser visto como uma instância do mTSP, onde carros são caixeiros e cidades são locais a serem visitados. O Trabalho discute o uso do *K-means* como um método de pré-processamento para geração de bons indivíduos para a população inicial atribuída à algoritmos bio-inspirados, neste caso GA e ACS. De modo a alcançar os objetivos desta pesquisa, os algoritmos foram implementadas e adicionadas ao sistema centralizado de escalonamento. Ressalta-se que o sistema foi adaptado, tornando-se um sistema de escalonamento de rotas para veículos. Com isso, o obejtivo geral desta pesquisa "Propor uma abordagem mTSP para coordenação de uma frota de veículos autônomos de entrega em um sistema centralizado"foi alcançado. Observou-se que ao aplicar o algoritmo *K-means* como método de pré-processamento para geração de bons indivíduos para populações iniciais dos algoritmos houve uma melhoria significativa com aplicação para GA mono-objetivo. A demonstração da coordenação dos veículos pode ser vista nos experimentos simulado e real, conduzidos.

Para os objetivos específicos, temos indícios significativos para concluir que foram alcançados, sendo os principais realizar uma comparação com o GA proposto em (ALVES; LOPES, 2015) e os algoritmos PSO e ACS com as versões híbridas com *K-means*. Para isso, experimentos foram conduzidos no sistema centralizado, onde comparamos as abordagens GA, PSO e ACS. Além disso, indentificamos que para o aumento constante de carros e locais de visita, a aplicação do *K-means* como um pré-processamento na geração de bons indivíduos pode proporcionar melhores resultados levando a uma minimização mais rápida e mais efetiva. Ressalta-se que houve uma melhora significativa da aplicação do *K-means* como pré-processamento para o GA, essa melhora pode ser observada nos experimentos 5.3 e 5.4.

6.1 Principais Contribuições

Para destacar as contribuições desta pesquisa, foram definidas três hipóteses na seção 1.3. A seguir são discutidas e validadas cada hipótese:

- Para responder esta pergunta, foi utilizada uma função *fitness* apresentada no trabalho (ALVES; LOPES, 2015). Tal função, realiza a combinação de dois objetivos simultâneos. Desse modo, os algoritmos conduzem um processo que visa minimizar a soma desses dois objetivos, ocasionando rotas mais equilibradas em relação a distância total viajada e o desvio padrão dessa distância.

- No experimento 5.3 as análises apresentam evidencias para concluir que o pré-processamento realizado com *K-means* para o GA proporcionou uma aceleração na convergência do algoritmo. Entretanto, para o ACS, o teste estatístico não encontrou diferença significativa entre as médias do K-means-ACS e ACS, em todos cenários.

- Para validar essa hípotese, o experimento 5.3 pode ser usado. Tal experimento, apresenta evidências para concluir que o algoritmo ACS é o melhor se comparado ao GA e PSO quando é realizado um aumento de locais de entrega e veículos.

6.2 Trabalhos Futuros

Apesar do bom desempenho apresentado pelos algoritmos com pré-processamento (K-means-ACS e K-means-GA), alguns pontos de melhoria são indicados. Um dos principais pontos levantados é que não foi possível realizar a configuração de cenários com mais locais de visita devido ao tempo que o A* leva para realizar o planejamento. Sendo que para alguns cenários o planejamento levou de três a cinco dias para ser finalizado. Levando isto em consideração, e que umas das limitações dos trabalhos relacionados frente ao planejamento e escalonamento em sistemas de execução em tempo real pode ser vista quanto a aplicação em ambientes reais, onde é exigido um tempo de resposta quase que instantâneo. Destacamos possíveis melhorias nesta pesquisa:

- Pesquisa e desenvolvimento de um método de planejamento que possua um menor tempo de execução comparado com métodos convencionais. Como medida para solucionar esse probelma tem-se como objetivo o estudo e aplicação de uma nova abordagem de distruibuição do algoritmo A*;

- Trabalhar com algoritmos bio-inspirados com mais objetivos. O problema de minimização da distância total e balanceamento de rotas é visto como um problema de 2 objetivos a serem minimizados. Trazendo o problema para uma aplicação em

tempo real novos objetivos devem ser adicionados. Desse modo, tratar esse tipo de problema com classes como *multi-objective* é visto como um tópico futuro;

- ❏ Integração do sistema centralizado com sistemas de veículos autônomos, possibilitando a definição de pontos de visita em um mapa de *GPS* e re-escalonamento;
- ❏ Desenvolvimento de um sistema centralizado para escalonamento e re-escalonamento online com foco em veículos autônomos.

6.3 Contribuições em Produção Bibliográfica

A pesquisa realizada durante esta dissertação produziu a submissão de um artigo científico no periódico *Anais da Academia Brasileira de Ciências (AABC) (Qualis B1)*. O artigo submetido conta com os resultados apresentados no experimento 5.3 e 5.5.

Referências

ABDEL-KADER, R. F. Hybrid discrete pso with ga operators for efficient qos-multicast routing. **Ain Shams Engineering Journal**, Elsevier, v. 2, n. 1, p. 21–31, 2011. Disponível em: <https://doi.org/10.1016/j.asej.2011.05.002>.

ADAMI, C. **Introduction to Artificial Life**. [S.l.]: Springer-Verlag/ Telos, 1998.

AL-TURJMAN, F. Hybrid approach for mobile couriers election in smart-cities. In: IEEE. **2016 IEEE 41st Conference on Local Computer Networks (LCN)**. 2016. p. 507–510. Disponível em: <https://doi.org/10.1109/LCN.2016.79>.

ALI, A. I.; KENNINGTON, J. L. The asymmetric m-travelling salesmen problem: A duality based branch-and-bound algorithm. **Discrete Applied Mathematics**, ELSEVIER, v. 13, p. 259–276, 1986. Disponível em: <https://doi.org/10.1016/0166-218X(86)90087-9>.

ALVES, R. M. F. **Coordenação, localização e navegação para robôs de serviço em ambientes internos**. Tese (Doutorado) — Universidade Federal de Uberlândia, 2017. Disponível em: <http://dx.doi.org/10.14393/ufu.te.2017.21>.

ALVES, R. M. F.; LOPES, C. R. Using genetic algorithms to minimize the distance and balance the routes for the multiple traveling salesman problem. In: **Evolutionary Computation (CEC)**. IEEE, 2015. Disponível em: <https://doi.org/10.1109/CEC.2015.7257285>.

ARYA, V.; GOYAL, A.; JAISWAL, V. An optimal solution to multiple travelling salesperson problem using modified genetic algorithm. **International Journal of Application or Innovation in Engineering & Management (IJAIEM)**, v. 3, p. 425–430, 2014. Disponível em: <https://ijaiem.org/volume3issue1/IJAIEM-2014-01-31-100.pdf>.

BÄCK, T.; FOGEL, D. B.; MICHALEWICZ, Z. **Handbook of evolutionary computation**. [S.l.]: CRC Press, 1997.

BACK, T.; FOGEL, D. B.; MICHALEWICZ, Z. **Evolutionary Computation 1, Basic Algorithms and Operators**. 1. ed. [S.l.]: Taylor e Francis Group, 2000. ISBN 9780750306645.

BAGLOEE, S. A. et al. Autonomous vehicles: challenges, opportunities, and future implications for transportation policies. **Journal of modern transportation**, Springer, v. 24, n. 4, p. 284–303, 2016. Disponível em: <https://doi.org/10.1007/s40534-016-0117-3>.

BARBETTA, P. A.; REIS, M. M.; BORNIA, A. C. **Estatística: para cursos de engenharia e informática**. [S.l.]: Atlas São Paulo, 2004. v. 3.

BARBOSA, C. E. M. **Algoritmos bio-inspirados para solução de problemas de otimização**. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2017. Disponível em: <https://repositorio.ufpe.br/handle/123456789/25229>.

BARBOSA, D.; KASHIWABARA, A. et al. Aplicaçao da otimizaçao por colônia de formigas ao problema de múltiplos caixeiros viajantes no atendimento de ordens de serviço nas empresas de distribuiçao de energia elétrica. In: SBC. **Anais Principais do XI Simpósio Brasileiro de Sistemas de Informação**. 2015. p. 23–30. Disponível em: <https://doi.org/10.5753/sbsi.2015.5797>.

BAYKASOĞLU, A.; ÖZBEL, B. K. Multiple traveling salesman game for cost allocation: a case problem for school bus services. In: **LM-SCM 2016 XIV. INTERNATIONAL LOGISTICS AND SUPPLY CHAIN CONGRESS**. [s.n.], 2016. p. 64. Disponível em: <https://doi.org/10.1155/2014/696945>.

BEKTAS, T. The multiple traveling salesman problem: an overview of formulations and solution procedures. **Omega**, ELSEVIER, v. 34, p. 209–219, 2006. Disponível em: <https://doi.org/10.1016/j.omega.2004.10.004>.

BELLMORE, M.; BELLMORE, M. Transformation of multisalesman problem to the standard traveling salesman problem. **Journal of the ACM (JACM)**, ACM, v. 21, p. 500–504, 1974. Disponível em: <https://doi.org/10.1145/321832.321847>.

BOLANOS, R. et al. A population-based algorithm for the multi travelling salesman problem. **International Journal of Industrial Engineering Computations**, v. 7, n. 2, p. 245–256, 2016. Disponível em: <https://doi.org/10.5267/j.ijiec.2015.10.005>.

BONABEAU, E.; DORIGO, M.; THERAULAZ, G. **Swarm Intelligence From Natural ti Artificial Systems**. 1. ed. [S.l.]: Oxford University Press, 1999.

BRAEKERS, K.; RAMAEKERS, K.; NIEUWENHUYSE, I. V. The vehicle routing problem: State of the art classification and review. **Computers & Industrial Engineering**, Elsevier, v. 99, p. 300–313, 2016. Disponível em: <https://doi.org/10.1016/j.cie.2015.12.007>.

CASTRO, L. N. de et al. Computação natural: Uma breve visão geral. In: **Workshop em Nanotecnologia e Computação Inspirada na Biologia**. Rio de Janeiro: [s.n.], 2004.

CHAKRABORTY, M.; CHAKRABORTY, U. K. Branching process analysis of linear ranking and binary tournament selection in genetic algorithms. **Journal of computing and information technology**, Fakultet elektrotehnike i računarstva Sveučilišta u Zagrebu, v. 7, n. 2, p. 107–113, 1999. Disponível em: <http://cit.fer.hr/index.php/CIT/article/view/2907/1771>.

CHEN, A.-l.; YANG, G.-k.; WU, Z.-m. Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. **Journal of Zhejiang University-Science A**, Springer, v. 7, n. 4, p. 607–614, 2006. Disponível em: <https://doi.org/10.1631/jzus.2006.A0607>.

CHEN, W.-N. et al. A novel set-based particle swarm optimization method for discrete optimization problems. **IEEE Transactions on evolutionary computation**, IEEE, v. 14, n. 2, p. 278–300, 2009. Disponível em: <https://doi.org/10.1109/TEVC.2009.2030331>.

CHEN, X. et al. Ant colony optimization based memetic algorithm to solve bi-objective multiple traveling salesmen problem for multi-robot systems. **IEEE Access**, IEEE, v. 6, p. 21745–21757, 2018. Disponível em: <https://doi.org/10.1109/ACCESS.2018.2828499>.

CHENGMING, Q. An ant colony algorithm with stochastic local search for the vrp. In: IEEE. **3rd International Conference on Innovative Computing Information and Control, Los Alamitos, CA, USA**. 2008. p. 464–468. Disponível em: <https://doi.org/10.1109/ICICIC.2008.130>.

CHRISTOFIDES, N. **Worst-case analysis of a new heuristic for the travelling salesman problem**. [S.l.], 1976. Disponível em: <https://apps.dtic.mil/dtic/tr/fulltext/u2/a025602.pdf>.

CLERC, M. Discrete particle swarm optimization, illustrated by the traveling salesman problem. In: **New optimization techniques in engineering**. Springer, 2004. p. 219–239. Disponível em: <https://doi.org/10.1007/978-3-540-39930-8_8>.

_____. **Particle swarm optimization**. [S.l.]: John Wiley & Sons, 2010. v. 93.

DASGUPTA, D. **Artificial Immune Systems and Their Applications**. [S.l.]: Springer, 1999.

DAVENDRA, D. **Traveling Salesman Problem, Theory and Applications**. Ed, 2010. ISBN 978-953-307-426-9. Disponível em: <https://doi.org/10.5772/547>.

DAVENPORT, J. H. A: 20piano movers". **ACM SIGSAM Bulletin**, ACM New York, NY, USA, v. 20, n. 1-2, p. 15–17, 1986. Disponível em: <https://doi.org/10.1145/12917.12919>.

DAVIS, L. **Handbook of genetic algorithms**. [S.l.]: CumInCAD, 1991.

DEEP, K.; THAKUR, M. A new crossover operator for real coded genetic algorithms. **Applied mathematics and computation**, Elsevier, v. 188, n. 1, p. 895–911, 2007. Disponível em: <https://doi.org/10.1016/j.amc.2006.10.047>.

DESALE, S. et al. Heuristic and meta-heuristic algorithms and their relevance to the real world: a survey. **Int. J. Comput. Eng. Res. Trends**, Citeseer, v. 351, n. 5, p. 2349–7084, 2015. Disponível em: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.741.3773&rep=rep1&type=pdf>.

DHEIN, G.; NETO, A. F. K.; ARAÚJO, O. C. B. de. The multiple traveling salesman problem with backup coverage. **Electronic Notes in Discrete Mathematics**, Elsevier, v. 66, p. 135–142, 2018. Disponível em: <https://doi.org/10.1016/j.endm.2018.03.018>.

DINO. **Carros autônomos já se tornam realidade e "robotáxis" começam a operar**. 2019. Disponível em: <https://exame.abril.com.br/negocios/dino/carros-autonomos-ja-se-tornam-realidade-e-robotaxis-comecam-a-operar/>.

DORIGO, M. Optimization, learning and natural algorithms. **PhD Thesis, Politecnico di Milano**, 1992.

DORIGO, M.; CARO, G. D. Ant colony optimization: a new meta-heuristic. In: IEEE. **Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)**. 1999. v. 2, p. 1470–1477. Disponível em: <https://doi.org/10.1109/CEC.1999.782657>.

DORIGO, M.; GAMBARDELLA, L. M. Ant colony system: a cooperative learning approach to the traveling salesman problem. **IEEE Transactions on evolutionary computation**, IEEE, v. 1, n. 1, p. 53–66, 1997. Disponível em: <https://doi.org/10.1109/4235.585892>.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: optimization by a colony of cooperating agents. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, IEEE, v. 26, n. 1, p. 29–41, 1996. Disponível em: <https://doi.org/10.1109/3477.484436>.

DORIGO, M.; STUTZLE, T. **Ant Colony Optimization. Bradford Company, MA**. [S.l.: s.n.], 2004.

DRÉO, J. et al. **Metaheuristics for hard optimization: methods and case studies**. [S.l.]: Springer Science & Business Media, 2006.

DUCHOŇ, F. et al. Path planning with modified a star algorithm for a mobile robot. **Procedia Engineering**, Elsevier, v. 96, p. 59–69, 2014. Disponível em: <https://doi.org/10.1016/j.proeng.2014.12.098>.

DUNN, O. J. Multiple comparisons using rank sums. **Technometrics**, Taylor & Francis Group, v. 6, n. 3, p. 241–252, 1964. Disponível em: <https://doi.org/10.1080/00401706.1964.10490181>.

ELLEGOOD, W. A. et al. School bus routing problem: Contemporary trends and research directions. **Omega**, Elsevier, v. 95, p. 102056, 2020. Disponível em: <https://doi.org/10.1016/j.omega.2019.03.014>.

ELLIOTT, D.; KEEN, W.; MIAO, L. Recent advances in connected and automated vehicles. **Journal of Traffic and Transportation Engineering (English Edition)**, Elsevier, v. 6, n. 2, p. 109–131, 2019. Disponível em: <https://doi.org/10.1016/j.jtte.2018.09.005>.

FARIA, E. **Teste de Significância**. 2017. Disponível em: <http://www.facom.ufu.br/~elaine/disc/MFCD/TesteEstatistico.pdf>.

FERNANDES, L. C. et al. Carina intelligent robotic car: Architectural design and applications. **Journal of Systems Architecture - Embedded Systems Design**, v. 60, p. 372–392, 2014. Disponível em: <https://doi.org/10.1016/j.sysarc.2013.12.003>.

_____. Intelligent robotic car for autonomous navigation: Platform and system architecture. In: **Second Brazilian Conference on Critical Embedded Systems**. IEEE, 2012. Disponível em: <https://doi.org/10.1109/CBSEC.2012.26>.

FERNÁNDEZ, C. et al. Autonomous navigation and obstacle avoidance of a micro-bus. **International Journal of Advanced Robotic Systems**, SAGE Publications Sage UK: London, England, v. 10, n. 4, p. 212, 2013. Disponível em: <https://doi.org/10.5772/56125>.

FOGEL, D. B. A parallel processing approach to a multiple traveling salesman problem using evolutionary programming. In: **Proceedings on the Fourth Annual Parallel Processing Symposium**. [S.l.]: Fullterton, CA, 1990. p. 318–326.

FUSSY, P.; OLIVEIRA, L. de. **O futuro é o carro sem motorista?** 2015. Disponível em: <http://g1.globo.com/carros/noticia/2015/11/o-futuro-e-o-carro-sem-motorista.html>.

GAMBARDELLA, L. M.; TAILLARD, É.; AGAZZI, G. Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In: CITESEER. **New ideas in optimization**. [S.l.], 1999.

GAVISH, B.; SRIKANTH, K. An optimal solution method for large-scale multiple traveling salesmen problems. **Journal of the Operational Research Society**, INFORMS, v. 24, p. 698–717, 1986. Disponível em: <https://doi.org/10.1287/opre.34.5.698>.

GOKSAL, F. P.; KARAOGLAN, I.; ALTIPARMAK, F. A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery. **Computers & Industrial Engineering**, Elsevier, v. 65, n. 1, p. 39–53, 2013. Disponível em: <https://doi.org/10.1016/j.cie.2012.01.005>.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization and Machine Learning**. [S.l.]: Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989. ISBN 0201157675.

GOLDBERG, D. E.; LINGLE, R. et al. Alleles, loci, and the traveling salesman problem. In: LAWRENCE ERLBAUM, HILLSDALE, NJ. **Proceedings of an international conference on genetic algorithms and their applications**. [S.l.], 1985. v. 154, p. 154–159.

GÓMEZ, A. E. et al. Simulation platform for cooperative vehicle systems. In: IEEE. **17th International IEEE Conference on Intelligent Transportation Systems (ITSC)**. 2014. p. 1347–1352. Disponível em: <https://doi.org/10.1109/ITSC.2014.6957874>.

GROMICHO, J.; PAIXAO, J.; BRONCO, J. Exact solution of multiple traveling salesman problems. In: **Combinatorial Optimization**. Springer, 1992. p. 291–292. Disponível em: <https://doi.org/10.1007/978-3-642-77489-8_27>.

GULCU, S. D.; ORNEK, H. K. Solution of multiple travelling salesman problem using particle swarm optimization based algorithms. **International Journal of Intelligent Systems and Applications in Engineering**, IJISAE, v. 7, p. 72–82, 2019. Disponível em: <https://doi.org/10.18201//ijisae.2019252784>.

GUNAWAN, S.; SUSYANTO, N.; BAHAR, S. Vehicle routing problem with pick-up and deliveries using genetic algorithm in express delivery services. In: AIP PUBLISHING LLC. **AIP Conference Proceedings**. 2019. v. 2192, n. 1, p. 060009. Disponível em: <https://doi.org/10.1063/1.5139155>.

GUNAWAN, S. A. et al. Smoothed a-star algorithm for nonholonomic mobile robot path planning. In: IEEE. **2019 International Conference on Information and Communications Technology (ICOIACT)**. 2019. p. 654–658. Disponível em: <https://doi.org/10.1109/ICOIACT46704.2019.8938467>.

GUOXING, Y. Transformation of multidepot multisalesmen problem to the standard travelling salesman problem. **European Journal of Operational Research**, ELSEVIER, v. 81, p. 557–560, 1995. Disponível em: <https://doi.org/10.1016/0377-2217(94)00011-Z>.

GUTIN, G.; YEO, A.; ZVEROVICH, A. Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the tsp. **Discrete Applied Mathematics**, Elsevier, v. 117, n. 1-3, p. 81–86, 2002. Disponível em: <https://doi.org/10.1016/S0166-218X(01)00195-0>.

HASHIMOTO, K. **Técnicas de otimização combinatória multiobjetivo aplicadas na estimação do desempenho elétrico de redes de distribuição.** Tese (Doutorado) — Universidade de São Paulo, 2004. Disponível em: <https://doi.org/10.11606/T.3.2004.tde-19112004-165342>.

HATA, A. Y.; WOLF, D. F. Feature detection for vehicle localization in urban environments using a multilayer lidar. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, v. 17, n. 2, p. 420–429, 2015. Disponível em: <https://doi.org/10.1109/TITS.2015.2477817>.

HEIBERGER, R. M.; NEUWIRTH, E. One-way anova. In: **R through excel**. [S.l.]: Springer, 2009. p. 165–191.

HOLLAND, J. H. et al. **Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence**. [S.l.]: MIT press, 1992.

HONG, S.; PADBERG, M. W. A note on the symmetric multiple traveling salesman problem with fixed charges. **Operations Research**, INFORMS, v. 25, p. 871–874, 1977. Disponível em: <https://doi.org/10.1287/opre.25.5.871>.

HU, Y.; YANG, S. X. A knowledge based genetic algorithm for path planning of a mobile robot. In: IEEE. **IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004**. 2004. v. 5, p. 4350–4355. Disponível em: <https://doi.org/10.1109/ROBOT.2004.1302402>.

HU, Z.-L. et al. Optimize grouping and path of pylon inspection in power system. **IEEE Access**, IEEE, v. 8, p. 108885–108895, 2020. Disponível em: <https://doi.org/10.1109/ACCESS.2020.3001435>.

IBMCORP. **IBM SPSS Statistics for Windows, Version 20.0**. 2011. Armonk, NY: IBM Corp.

JANIKOW, C. Z.; MICHALEWICZ, Z. An experimental comparison of binary and floating point representations in genetic algorithms. In: **ICGA**. [S.l.: s.n.], 1991. v. 1991, p. 31–36.

JONKER, R.; VOLGENANT, T. An improved transformation of the symmetric multiple traveling salesman problem. **Operations Research**, INFORMS, v. 36, p. 163–167, 1988. Disponível em: <https://doi.org/10.1287/opre.36.1.163>.

JORDEHI, A. R.; JASNI, J. Parameter selection in particle swarm optimisation: a survey. **Journal of Experimental & Theoretical Artificial Intelligence**, Taylor & Francis, v. 25, n. 4, p. 527–542, 2013. Disponível em: <https://doi.org/10.1080/0952813X.2013.782348>.

JÚNIOR, O. S. da S. **Algoritmos para os Problemas de Roteirização Estática e Dinâmica de Veículos com Janelas de Tempo**. Tese (Doutorado) — PUC-Rio, 2013. Disponível em: <https://doi.org/10.17771/PUCRio.acad.21994>.

JUNJIE, P.; DINGWEI, W. An ant colony optimization algorithm for multiple travelling salesman problem. In: IEEE. **First International Conference on Innovative Computing, Information and Control-Volume I (ICICIC'06)**. 2006. v. 1, p. 210–213. Disponível em: <https://doi.org/10.1109/ICICIC.2006.40>.

KARA, I.; BEKTAS, T. Integer linear programming formulations of multiple salesman problems and its variations. **European Journal of Operational Research**, ELSEVIER, v. 174, p. 1449–1458, 2006. Disponível em: <https://doi.org/10.1016/j.ejor.2005.03.008>.

KARAKATIč, S.; PODGORELEC, V. A survey of genetic algorithms for solving multi depot vehicle routing problem. **Applied Soft Computing**, Elsevier, v. 27, p. 519–532, 2015. Disponível em: <https://doi.org/10.1016/j.asoc.2014.11.005>.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: IEEE. **Proceedings of ICNN'95-International Conference on Neural Networks**. 1995. v. 4, p. 1942–1948. Disponível em: <https://doi.org/10.1109/ICNN.1995.488968>.

KENNEDY, J.; EBERHART, R. C. A discrete binary version of the particle swarm algorithm. In: IEEE. **1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation**. 1997. v. 5, p. 4104–4108. Disponível em: <https://doi.org/10.1109/ICSMC.1997.637339>.

KENNEDY, J.; EBERHART, R. C.; SHI, Y. **Swarm Intelligence**. [S.l.]: Morgan Kaufmann Publishers, 2001.

KESELMAN, H. J.; ROGAN, J. C. The tukey multiple comparison test: 1953–1976. **Psychological Bulletin**, American Psychological Association, v. 84, n. 5, p. 1050–1056, 1977. Disponível em: <https://doi.org/10.1037/0033-2909.84.5.1050>.

KIRáLY, A.; ABONYI, J. A novel approach to solve multiple traveling salesmen problem by genetic algorithm. In: **Computational Intelligence in Engineering**. Springer, 2010. p. 141–151. Disponível em: <https://doi.org/10.1007/978-3-642-15220-7_12>.

KRUSKAL, W. H.; WALLIS, W. A. Use of ranks in one-criterion variance analysis. **Journal of the American statistical Association**, Taylor & Francis Group, v. 47, n. 260, p. 583–621, 1952. Disponível em: <https://doi.org/10.1080/01621459.1952.10483441>.

KULKARNI, R. V.; BHAVE, P. R. Integer programming formulations of vehicle routing problems. **European Journal of Operational Research**, ELSEVIER, v. 20, p. 58–67, 1985. Disponível em: <https://doi.org/10.1016/0377-2217(85)90284-X>.

LAPORTE, G.; NOBERT, Y. A cutting planes algorithm for the m-salesmen problem. **Journal of the Operational Research Society**, JSTOR, v. 31, p. 1017–1023, 1980. Disponível em: <https://doi.org/10.2307/2581282>.

LAPORTE, G.; NOBERT, Y.; TAILLEFER, S. Solving a family of multi-depot vehicle routing and location-routing problems. **Transportation Science**, INFORMS, v. 22, p. 161–172, 1988. Disponível em: <https://doi.org/10.1287/trsc.22.3.161>.

LARRANAGA, P. et al. Genetic algorithms for the travelling salesman problem: A review of representations and operators. **Artificial Intelligence Review**, Springer, v. 13, n. 2, p. 129–170, 1999. Disponível em: <https://doi.org/10.1023/A:1006529012972>.

LATAH, M. Solving multiple tsp problem by k-means and crossover based modified aco algorithm. **International Journal of Engineering Research & Technology - IJERT**, v. 5, n. 2, p. 430–434, 2016. Disponível em: <https://www.ijert.org/research>.

LAUMOND, J.-P. et al. **Robot motion planning and control**. [S.l.]: Springer, 1998. v. 229.

LEVENE, H. Robust tests for equality of variances. In: **Contributions to probability and statistics: Essays in honor of Harold Hotelling**. [S.l.]: Stanford University Press, 1960. p. 278–292.

LI, X.; TIAN, P. An ant colony system for the open vehicle routing problem. In: SPRINGER. **International Workshop on Ant Colony Optimization and Swarm Intelligence**. 2006. p. 356–363. Disponível em: <https://doi.org/10.1007/11839088_33>.

LI, X.; TIAN, P.; LEUNG, S. C. An ant colony optimization metaheuristic hybridized with tabu search for open vehicle routing problems. **Journal of the Operational Research Society**, Taylor & Francis, v. 60, n. 7, p. 1012–1025, 2009. Disponível em: <https://doi.org/10.1057/palgrave.jors.2602644>.

LIAO, C.-J.; TSENG, C.-T.; LUARN, P. A discrete version of particle swarm optimization for flowshop scheduling problems. **Computers & Operations Research**, Elsevier, v. 34, n. 10, p. 3099–3111, 2007. Disponível em: <https://doi.org/10.1016/j.cor.2005.11.017>.

LIAO, X.-L.; CHIEN, C.-H.; TING, C.-K. A genetic algorithm for the minimum latency pickup and delivery problem. In: **Evolutionary Computation (CEC)**. IEEE, 2014. Disponível em: <https://doi.org/10.1109/CEC.2014.6900627>.

LIN, S.; KERNIGHAN, B. W. An effective heuristic algorithm for the traveling-salesman problem. **Operations research**, INFORMS, v. 21, n. 2, p. 498–516, 1973. Disponível em: <https://doi.org/10.1287/opre.21.2.498>.

LIU, Q. et al. Global path planning for autonomous vehicles in off-road environment via an a-star algorithm. **International Journal of Vehicle Autonomous Systems**, Inderscience Publishers (IEL), v. 13, n. 4, p. 330–339, 2017. Disponível em: <https://doi.org/10.1504/IJVAS.2017.087148>.

LRM. **Projeto CaRINA 2**. 2011. Disponível em: <http://lrm.icmc.usp.br/web/index.php?n=Port.ProjCarina2Info>.

MA, Y. et al. Coordinated optimization algorithm combining ga with cluster for multi-uavs to multi-tasks task assignment and path planning. In: IEEE. **2019 IEEE 15th International Conference on Control and Automation (ICCA)**. 2019. p. 1026–1031. Disponível em: <https://doi.org/10.1109/ICCA.2019.8899987>.

MACQUEEN, J. et al. Some methods for classification and analysis of multivariate observations. In: OAKLAND, CA, USA. **Proceedings of the fifth Berkeley symposium on mathematical statistics and probability**. [S.l.], 1967. v. 1, n. 14, p. 281–297.

MANDELBROT, B. B. **The Fractal Geometry of Nature**. [S.l.]: WH freeman, 1983. v. 173.

MARINAKIS, Y.; MARINAKI, M. A hybrid genetic–particle swarm optimization algorithm for the vehicle routing problem. **Expert Systems with Applications**, Elsevier, v. 37, n. 2, p. 1446–1455, 2010. Disponível em: <https://doi.org/10.1016/j.eswa.2009.06.085>.

MARINAKIS, Y.; MIGDALAS, A.; PARDALOS, P. M. Expanding neighborhood grasp for the traveling salesman problem. **Computational Optimization and Applications**, Springer, v. 32, n. 3, p. 231–257, 2005. Disponível em: <https://doi.org/10.1007/s10589-005-4798-5>.

_____. Multiple phase neighborhood search—grasp based on lagrangean relaxation, random backtracking lin–kernighan and path relinking for the tsp. **Journal of combinatorial optimization**, Springer, v. 17, n. 2, p. 134–156, 2009. Disponível em: <https://doi.org/10.1007/s10878-007-9104-2>.

MASUTTI, T. A.; CASTRO, L. N. de. Vroptbees: A bee-inspired framework for solving vehicle routing problems. **International Journal of Natural Computing Research (IJNCR)**, IGI Global, v. 7, n. 1, p. 32–56, 2018. Disponível em: <https://doi.org/10.4018/IJNCR.2018010103>.

MICHALEWICZ, Z.; SCHOENAUER, M. Evolutionary algorithms for constrained parameter optimization problems. **Evolutionary computation**, IEEE, v. 4, n. 1, p. 1–32, 1996. Disponível em: <https://doi.org/10.1162/evco.1996.4.1.1>.

MINITAB, B. da. **Como escolher entre um teste não paramétrico e um teste paramétrico**. 2019. Disponível em: <https://blog.minitab.com/pt/como-escolher-entre-um-teste-nao-parametrico-e-um-teste-parametrico>.

MITCHELL, T. M. **Machine Learning**. [S.l.]: McGraw-Hill Science/ Engineering, 1997. ISBN 0070428077.

MODARES, A.; SOMHOM, S.; ENKAWA, T. A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems. **International Transactions in Operational Research**, ELSEVIER, v. 6, p. 591–606, 1999. Disponível em: <https://doi.org/10.1016/S0969-6016(99)00015-5>.

MONTEMANNI, R. et al. Ant colony system for a dynamic vehicle routing problem. **Journal of Combinatorial Optimization**, Springer, v. 10, n. 4, p. 327–343, 2005. Disponível em: <https://doi.org/10.1007/s10878-005-4922-6>.

NERY, S. W. L. **Análise de operadores de cruzamento genético aplicados ao problema do Caixeiro Viajante**. 2017. Monografia (Bacharelado em Ciências da Computação), UFG (Universidade Federal de Goiás), Catalão, Brasil. Disponível em: <https://files.cercomp.ufg.br/weby/up/498/o/SamuelWanbergLourencoNery2016.pdf>.

NIELSEN, M. A.; CHUANG, I. L. Quantum computation and quantum information. **American Journal of Physics**, v. 70, p. 558, 4 2002. Disponível em: <https://doi.org/10.1119/1.1463744>.

OKULEWICZ, M.; MAŃDZIUK, J. Application of particle swarm optimization algorithm to dynamic vehicle routing problem. In: SPRINGER. **International Conference on Artificial Intelligence and Soft Computing**. 2013. p. 547–558. Disponível em: <https://doi.org/10.1007/978-3-642-38610-7_50>.

OLIVEIRA, M. C. S. de; SILVA, T. L.; ALOISE, D. J. Otimização por nuvem de partículas: diferença entre aplicações a problemas contínuos e discretos. In: **Proc. of the Brazilian Sym. on Operations Research**. [S.l.: s.n.], 2004.

OLIVER, I.; SMITH, D.; HOLLAND, J. R. Study of permutation crossover operators on the traveling salesman problem. In: HILLSDALE, NJ: L. ERLHAUM ASSOCIATES, 1987. **Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA**. [S.l.], 1987.

PANG, W. et al. Fuzzy discrete particle swarm optimization for solving traveling salesman problem. In: IEEE. **The Fourth International Conference on Computer and Information Technology, 2004. CIT'04.** 2004. p. 796–800. Disponível em: <https://doi.org/10.1109/CIT.2004.1357292>.

PAPPA, G. L. **Algoritmos Bio-Inspirados Conceitos e Aplicações em Aprendizado de Máquina**. 2005. Disponível em: <https://homepages.dcc.ufmg.br/~glpappa/cverao/CursoVerao-Parte1.pdf>.

PARDALOS, P. M.; DU, D.-Z.; GRAHAM, R. L. **Handbook of combinatorial optimization**. Springer, 2013. ISBN 978-1-4419-7997-1. Disponível em: <https://doi.org/10.1007/978-1-4419-7997-1>.

PENDLETON, S. D. et al. Perception, planning, control, and coordination for autonomous vehicles. **Machines**, Multidisciplinary Digital Publishing Institute, v. 5, n. 1, p. 6, 2017. Disponível em: <https://doi.org/10.3390/machines5010006>.

PINOTTI, C. de A. S. **Desafios na Aplicação de Particle Swarm Optimization em um Problema de Planejamento de Produção de uma Olaria**. Dissertação (Dissertação de Mestrado) — Universidade Federal do Paraná, Março 2017.

POLI, R. et al. **A field guide to genetic programming**. [S.l.]: Lulu. com, 2008.

POTVIN, J.-Y.; LAPALME, G.; ROUSSEAU, J.-M. A generalized k-opt exchange procedure for the mtsp. **INFOR: Information Systems and Operational Research**, Taylor & Francis, v. 27, p. 474–481, 1989. Disponível em: <https://doi.org/10.1080/03155986.1989.11732113>.

PăUN, G.; ROZENBERG, G.; SALOMAA, A. **DNA Computing New Computing Paradigms**. 1. ed. [S.l.]: Springer, 1998. ISBN 9783662035634.

QUEIROZ, M. M. de. **Métodos Heurísticos Aplicados ao Problema de Programação da Frota de Navios PLVs**. Dissertação (Mestrado) — Universidade de São Paulo, 2011. Disponível em: <https://doi.org/10.11606/D.3.2011.tde-15032012-123216>.

RAO, M. R. A note on the multiple traveling salesmen problem. **Operations Research**, INFORMS, v. 28, p. 628–632, 1980. Disponível em: <https://doi.org/10.1287/opre.28.3.628>.

REIMANN, M.; DOERNER, K.; HARTL, R. F. D-ants: Savings based ants divide and conquer the vehicle routing problem. **Computers & Operations Research**, Elsevier, v. 31, n. 4, p. 563–591, 2004. Disponível em: <https://doi.org/10.1016/S0305-0548(03)00014-5>.

REIMANN, M.; STUMMER, M.; DOERNER, K. A savings based ant system for the vehicle routing problem. In: MORGAN KAUFMANN PUBLISHERS INC. **Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation**. [S.l.], 2002. p. 1317–1326.

RIDEL, D. et al. Obstacle avoidance using stereo-based generic obstacle tracking. In: IEEE. **2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)**. 2017. p. 1–6. Disponível em: <https://doi.org/10.1109/SBR-LARS-R.2017.8215284>.

RUSSELL, R. A. An effective heuristic for the m-tour traveling salesman problem with some side conditions. **Journal of the Operational Research Society**, INFORMS, v. 25, p. 517–524, 1977. Disponível em: <https://doi.org/10.1287/opre.25.3.517>.

RUSSELL, S.; NORVIG, P. **Inteligência Artificial - 3ª Ed. 2013**. [S.l.]: Elsevier, 2013. ISBN 9788535237016.

RYAN, J. L. et al. Reactive tabu search in unmanned aerial reconnaissance simulations. In: **Proceedings of the 30th conference on Winter simulation**. IEEE, 1998. Disponível em: <https://doi.org/10.1109/WSC.1998.745084>.

SADIQ, S. The traveling salesman problem: Optimizing delivery routes using genetic algorithms. In: **SAS Global Forum 2012**. [s.n.], 2012. Disponível em: <http://support.sas.com/resources/papers/proceedings12/161-2012.pdf>.

SAKURAI, Y. et al. Backtrack and restart genetic algorithm to optimize delivery schedule. In: **Sixth International Conference on Signal-Image Technology and Internet Based Systems**. IEEE, 2010. Disponível em: <https://doi.org/10.1109/SITIS.2010.24>.

_____. Inner random restart genetic algorithm to optimize delivery schedule. In: **International Conference on Systems, Man and Cybernetics**. IEEE, 2010. Disponível em: <https://doi.org/10.1109/ICSMC.2010.5642248>.

SATHYA, N.; MUTHUKUMARAVEL, A. Two phase hybrid ai-heuristics for multiple travelling salesman problem. **International Journal of Applied Engineering Research**, v. 12, n. 22, p. 12659–12664, 2017. Disponível em: <https://www.ripublication.com/ijaer17/ijaerv12n22_124.pdf>.

SEDIGHPOURA, M.; YOUSEFIKHOSHBAKHT, M.; DARANI, N. M. An effective genetic algorithm for solving the multiple traveling salesman problem. **Journal of Optimization in Industrial Engineering**, v. 8, p. 73–79, 2011. Disponível em: <http://www.qjie.ir/article_89.html>.

SERAPIÃO, A. B. d. S. Fundamentos de otimização por inteligência de enxames: uma visão geral. **Sba: Controle & Automação Sociedade Brasileira de Automatica**, SciELO Brasil, v. 20, n. 3, p. 271–304, 2009.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). **Biometrika**, JSTOR, v. 52, n. 3/4, p. 591–611, 1965. Disponível em: <https://doi.org/10.2307/2333709>.

SIEGEL, S.; JR, N. J. C. **Estatística não-paramétrica para ciências do comportamento**. [S.l.]: Artmed Editora, 1975.

SILVA, S. F. d. **Seleção de características por meio de algoritmos genéticos para aprimoramento de rankings e de modelos de classificação**. Tese (Doutorado) — Universidade de São Paulo, 2011. Disponível em: <https://doi.org/10.11606/T.55.2011.tde-19072011-151501>.

SIMON, H. **Neural networks: a comprehensive foundation**. [S.l.]: Prentice hall, 1999.

SONG, C.-H.; LEE, K.; LEE, W. D. Extended simulated annealing for augmented tsp and multi-salesmen tsp. In: **Proceedings of the International Joint Conference on Neural Networks**. IEEE, 2003. p. 2340–2343. Disponível em: <https://doi.org/10.1109/IJCNN.2003.1223777>.

SOUZA, J. R. d. **Navegação autônoma para robôs móveis usando aprendizado supervisionado.** Tese (Doutorado) — Universidade de São Paulo, 2014. Disponível em: <https://doi.org/10.11606/T.55.2014.tde-10062014-094624>.

STEFANELLO, F. **Hibridização de Métodos Exatos e Heurísticos para Resolução de Problemas de Otimização Combinatória**. Dissertação (Mestrado) — Universidade Federal de Santa Maria, 2011. Disponível em: <http://repositorio.ufsm.br/handle/1/5378>.

SYSWERDA, G. Schedule optimization using genetic algorithms. In: **Handbook of genetic algorithms, L. Davis**. [S.l.: s.n.], 1991. p. 332–349.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. **Introduction to data mining**. [S.l.]: Pearson Education India, 2016.

TANG, L.; LIU, J.; YANG, A. R. Z. A multiple traveling salesman problem model for hot rolling scheduling in shanghai baoshan iron & steel complex. **European Journal of Operational Research**, ELSEVIER, v. 124, p. 267–282, 2000. Disponível em: <https://doi.org/10.1016/S0377-2217(99)00380-X>.

TIMMIS, J.; NEAL, M.; THORNILEY, J. An adaptive neuro-endocrine system for robotic systems. In: **2009 IEEE Workshop on Robotic Intelligence in Informationally Structured Space**. [s.n.], 2009. p. 129–136. Disponível em: <https://doi.org/10.1109/RIISS.2009.4937917>.

TOOR, A.; WARREN, T. **Domino's and Ford will test self-driving pizza delivery cars**. 2017. Disponível em: <https://www.theverge.com/2017/8/29/16213544/dominos-ford-pizza-self-driving-car>.

TORKI, A.; SOMHON, S.; ENKAWA, T. A competitive neural network algorithm for solving vehicle routing problem. **Computers & Industrial Engineering**, ELSEVIER, v. 33, p. 473–476, 1997. Disponível em: <https://doi.org/10.1016/S0360-8352(97)00171-X>.

VAKHUTINSKY, A. I.; GOLDEN, B. L. Solving vehicle routing problems using elastic nets. In: **International Conference on Neural Networks**. IEEE, 1994. p. 4535–4540. Disponível em: <https://doi.org/10.1109/ICNN.1994.375004>.

VALLIVAARA, I. A team ant colony optimization algorithm for the multiple travelling salesmen problem with minmax objective. In: ACTA PRESS. **Proceedings of the 27th IASTED International Conference on Modelling, Identification and Control**. [S.l.], 2008. p. 387–392.

VIALI, L. **Testes de Hipóteses Não Paramétricos**. 2008. Disponível em: <http://www.mat.ufrgs.br/~viali/estatistica/mat2282/material/apostilas/Testes_Nao_Parametricos.pdf>.

WADE, M. R. **Vale do Silício está ganhando corrida para construir carros sem motorista**. 2018. Disponível em: <https://exame.abril.com.br/tecnologia/vale-do-silicio-esta-ganhando-corrida-para-construir-carros-sem-motorista/>.

WEI, C.; QIANG, S.; GAO, X. Z. Artificial endocrine system and applications. In: **2006 Chinese Control Conference**. [s.n.], 2006. p. 1433–1437. ISSN 2161-2927. Disponível em: <https://doi.org/10.1109/CHICC.2006.280709>.

WOLF, D. F. et al. Robótica móvel inteligente: da simulação às aplicações no mundo real. In: **XXVIII Jornadas de Atualização em Informática**. [s.n.], 2009. Disponível em: <http://osorio.wait4.org/publications/2009/CL_JAI2009_Completo.pdf>.

WU, J.; TAN, Y. A particle swarm optimization algorithm for grain logistics vehicle routing problem. In: IEEE. **2009 ISECS International Colloquium on Computing, Communication, Control, and Management**. 2009. v. 3, p. 364–367. Disponível em: <https://doi.org/10.1109/CCCM.2009.5267915>.

XU, X. et al. Two phase heuristic algorithm for the multiple-travelling salesman problem. **Soft Computing**, Springer, v. 22, n. 19, p. 6567–6581, 2018. Disponível em: <https://doi.org/10.1007/s00500-017-2705-5>.

YANG, C.; SZETO, K. Y. Solving the traveling salesman problem with a multi-agent system. In: IEEE. **2019 IEEE Congress on Evolutionary Computation (CEC)**. 2019. p. 158–165. Disponível em: <https://doi.org/10.1109/CEC.2019.8789895>.

YANG, X.-S. **Nature-Inspired Metaheuristic Algorithms**. 2. ed. [S.l.]: Luniver Press, 2010. ISBN 1905986289.

YAO, J. et al. Path planning for virtual human motion using improved a* star algorithm. In: IEEE. **2010 Seventh international conference on information technology: new generations**. 2010. p. 1154–1158. Disponível em: <https://doi.org/10.1109/ITNG.2010.53>.

YU, W.; LU, L. A route planning strategy for the automatic garment cutter based on genetic algorithm. In: **Evolutionary Computation (CEC)**. IEEE, 2014. Disponível em: <https://doi.org/10.1109/CEC.2014.6900425>.

YU, Z. et al. An implementation of evolutionary computation for path planning of cooperative mobile robots. In: **Proceedings of the 4th World Congress on Intelligent Control and Automation**. IEEE, 2002. Disponível em: <https://doi.org/10.1109/WCICA.2002.1021392>.

ZHANG, T.; GRUVER, W. A.; SMITH, M. H. Team scheduling by genetic search. In: **Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials. IPMM'99**. IEEE, 1999. Disponível em: <https://doi.org/10.1109/IPMM.1999.791495>.

ZHANG, Z.; ZHAO, Z. A multiple mobile robots path planning algorithm based on a-star and dijkstra algorithm. **International Journal of Smart Home**, v. 8, n. 3, p. 75–86, 2014. Disponível em: <http://dx.doi.org/10.14257/ijsh.2014.8.3.07>.

ZHOU, H.; SONG, M.; PEDRYCZ, W. A comparative study of improved ga and pso in solving multiple traveling salesmen problem. **Applied Soft Computing**, Elsevier, v. 64, p. 564–580, 2018. Disponível em: <https://doi.org/10.1016/j.asoc.2017.12.031>.

—

Apêndices

APÊNDICE A

Imagens e Tabelas do Experimento 2

A.1 Configurações dos Cenários do Experimento

As Figuras a seguir apresentam as configurações definidas para os treze cenários do experimento 2.

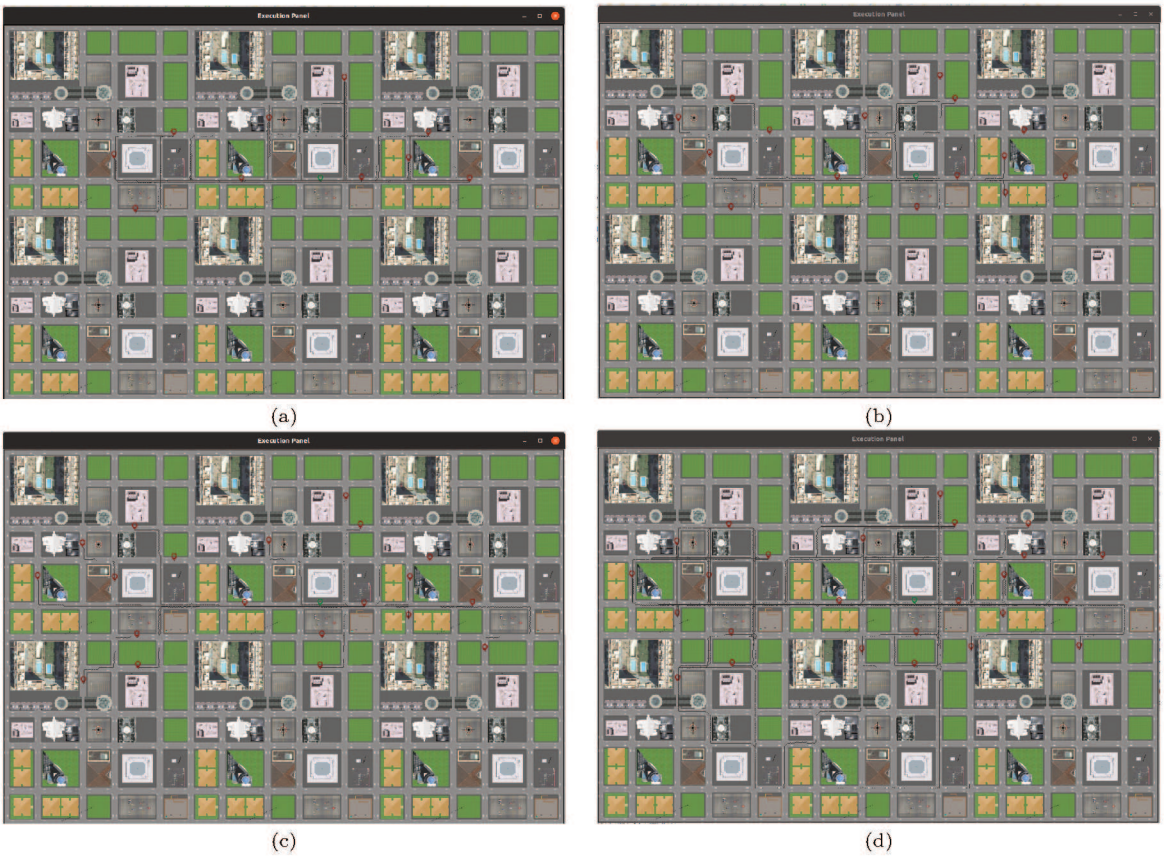


Figura 29 – Configuração dos cenários (a) ao (d), definidos para o escalonamento de rotas no mapa da cidade virtual 2.

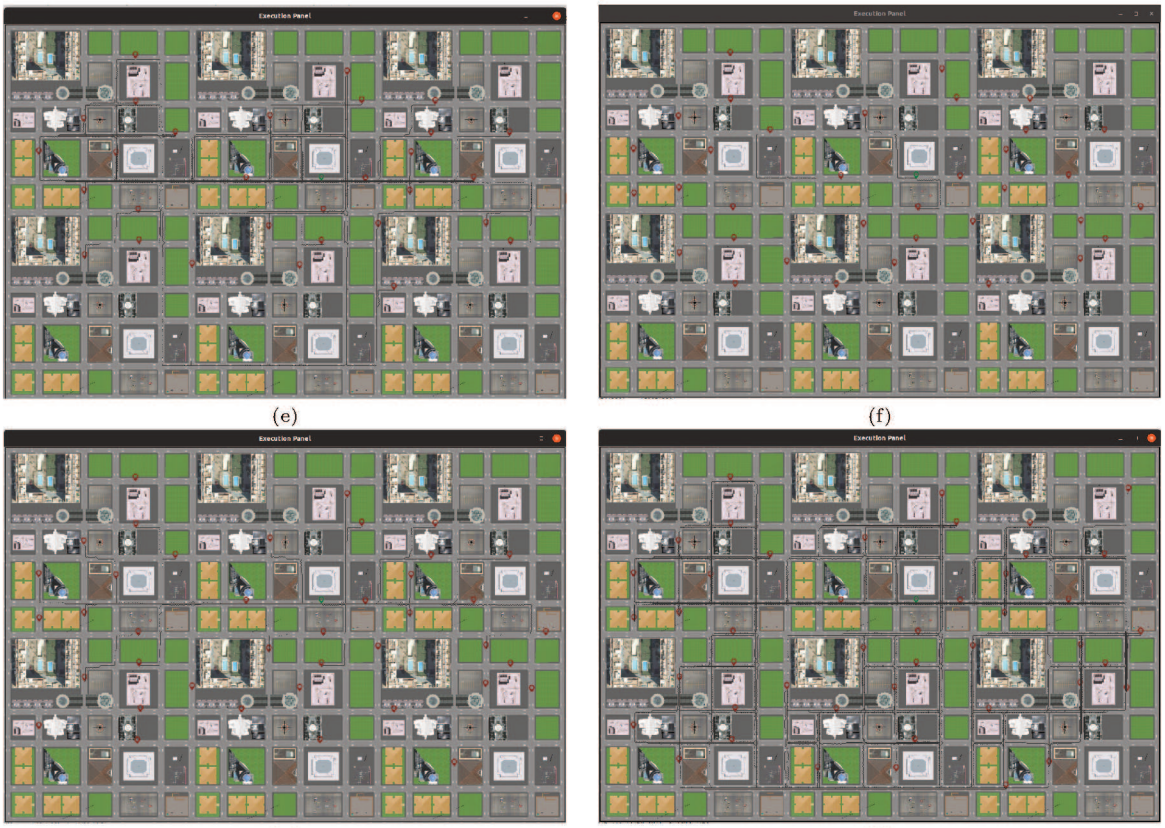
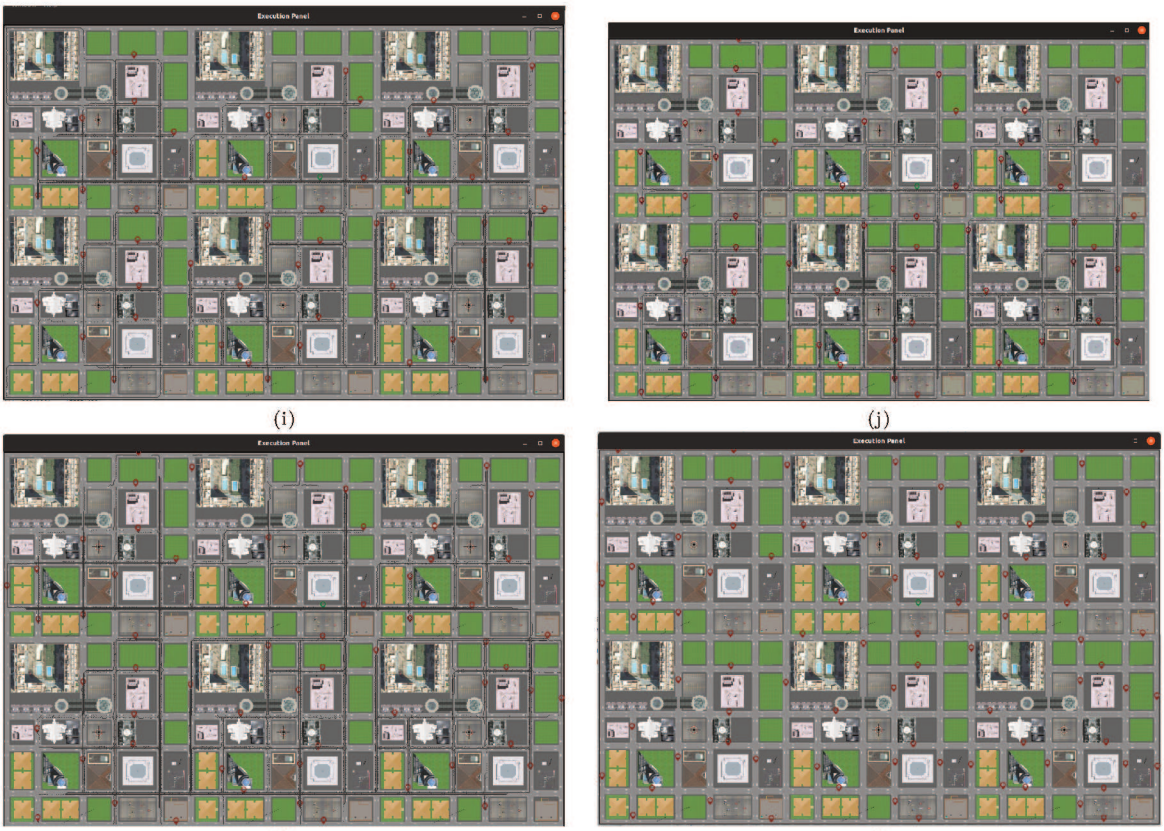


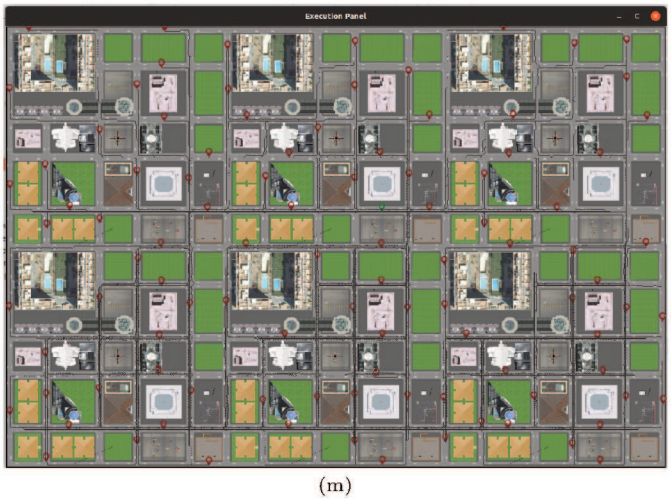
Figura 30 – Configuração dos cenários (e) ao (h), definidos para o escalonamento de rotas no mapa da cidade virtual 2.



(i) (j)

(k) (l)

Figura 31 – Configuração dos cenários (i) ao (l), definidos para o escalonamento de rotas no mapa da cidade virtual 2.



(m)

Figura 32 – Configuração do cenário (m) definido para o escalonamento de rotas no mapa da cidade virtual 2.

A.2 Tabelas com os Resultados Estatísticos da Minimização de *Fitness*

As Tabelas a seguir apresentam os resultados do Teste *Dunn-Bonferroni* aplicado como pós teste ao teste *Kruskal-Wallis*.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
ACS - GA	2,290	,176	1,000
ACS - K-means-ACS	-70,990	-5,455	**,000**
ACS - K-means-GA	-73,280	-5,631	**,000**
ACS - PSO	130,940	10,061	**,000**
GA - K-means-ACS	-68,700	-5,279	**,000**
GA - K-means-GA	-70,990	-5,455	**,000**
GA - PSO	-128,650	-9,885	**,000**
K-means-ACS - K-means-GA	2,290	1,76	1,000
K-means-ACS - PSO	59,950	4,607	**,000**
K-means-GA - PSO	57,660	4,431	**,000**

Tabela 20 – Cenário (a): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (a). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
GA - ACS	-29,880	-2,079	,377
GA - K-means-GA	-113,820	-7,918	**,000**
GA - K-means-ACS	-119,780	-8,333	**,000**
GA - PSO	-178,370	-12,408	**,000**
ACS - K-means-GA	-83,940	-5,839	**,000**
ACS - K-means-ACS	-89,900	-6,254	**,000**
ACS - PSO	148,490	10,330	**0,000**
K-means-GA - K-means-ACS	-5,960	-,415	1,000
K-means-GA - PSO	64,550	4,490	**,000**
K-means-ACS - PSO	58,590	4,076	**,000**

Tabela 21 – Cenário (b): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (b). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
GA - ACS	-12,560	-,869	1,000
GA - K-means-ACS	-76,130	-5,264	**,000**
GA - K-means-GA	-78,090	-5,400	**,000**
GA - PSO	-166,120	-11,487	**,000**
ACS - K-means-ACS	-63,570	-4,396	**,000**
ACS - K-means-GA	-65,530	-4,531	**,000**
ACS - PSO	153,560	10,619	**,000**
K-means-ACS - K-means-GA	1,960	,136	1,000
K-means-ACS - PSO	89,990	6,223	**,000**
K-means-GA - PSO	88,030	6,087	**,000**

Tabela 22 – Cenário (c): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (c). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
ACS - GA	15,420	1,066	1,000
ACS - K-means-GA	-90,940	-6,288	**,000**
ACS - K-means-ACS	-98,200	-6,790	**,000**
ACS - PSO	176,140	12,179	**,000**
GA - K-means-GA	-75,520	-5,222	**,000**
GA - K-means-ACS	-82,780	-5,724	**,000**
GA - PSO	-160,720	-11,113	**,000**
K-means-GA - K-means-ACS	-7,260	-,502	1,000
K-means-GA - PSO	85,200	5,891	**,000**
K-means-ACS - PSO	77,940	5,389	**,000**

Tabela 23 – Cenário (d): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (d). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
ACS - GA	34,300	2,372	,177
ACS - K-means-GA	-82,850	-5,729	**,000**
ACS - K-means-ACS	-89,330	-6,177	**,000**
ACS - PSO	176,620	12,212	**,000**
GA - K-means-GA	-48,550	-3,357	**,008**
GA - K-means-ACS	-55,030	-3,805	**,001**
GA - PSO	-142,320	-9,841	**,000**
K-means-GA - K-means-ACS	-6,480	-,448	1,000
K-means-GA - PSO	93,770	6,484	**,000**
K-means-ACS - PSO	87,290	6,036	**,000**

Tabela 24 – Cenário (e): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (e). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
ACS - K-means-GA	-64,220	-4,440	**,000**
ACS - K-means-ACS	-65,580	-4,534	**,000**
ACS - GA	87,720	6,065	**,000**
ACS - PSO	179,380	12,403	**,000**
K-means-GA - K-means-ACS	-1,360	-,094	1,000
K-means-GA - GA	23,500	1,625	1,000
K-means-GA - PSO	115,160	7,963	**,000**
K-means-ACS - GA	22,140	1,531	1,000
K-means-ACS - PSO	113,800	7,869	**,000**
GA - PSO	-91,660	-6,338	**,000**

Tabela 25 – Cenário (f): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (f). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
ACS - K-means-GA	-46,420	-3,210	**,013**
ACS - K-means-ACS	-55,740	-3,854	**,001**
ACS - GA	99,600	6,887	**,000**
ACS - PSO	175,440	12,131	**,000**
K-means-GA - K-means-ACS	-9,320	-,644	1,000
K-means-GA - GA	53,180	3,677	**,002**
K-means-GA - PSO	129,020	8,921	**,000**
K-means-ACS - GA	43,860	3,033	**,024**
K-means-ACS - PSO	119,700	8,277	**,000**
GA - PSO	-75,840	-5,244	**,000**

Tabela 26 – Cenário (g): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (g). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
ACS - K-means-ACS	-37,840	-2,616	,089
ACS - K-means-GA	-52,400	-3,623	**,003**
ACS - GA	108,880	7,528	**,000**
ACS - PSO	174,780	12,085	**,000**
K-means-ACS - K-means-GA	14,560	,1,007	1,000
K-means-ACS - GA	71,040	4,912	**,000**
K-means-ACS - PSO	136,940	9,469	**,000**
K-means-GA -GA	56,480	3,905	**,001**
K-means-GA - PSO	122,380	8,462	**,000**
GA - PSO	-65,900	-4,557	**,000**

Tabela 27 – Cenário (h): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (h). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
ACS - K-means-GA	-58,660	-4,056	**,000**
ACS = K-means-ACS	-78,020	-5,395	**,000**
ACS - GA	123,560	8,543	**,000**
ACS - PSO	190,060	13,141	**,000**
K-means-GA - K-means-ACS	-19,360	-1,339	1,000
K-means-GA - GA	64,900	4,487	**,000**
K-means-GA - PSO	131,400	9,086	**,000**
K-means-ACS - GA	45,540	3,149	**,016**
K-means-ACS - PSO	112,040	7,747	**,000**
GA - PSO	-66,500	-4,598	**,000**

Tabela 28 – Cenário (i): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (i). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-GA - K-means-ACS	-3,140	-,217	1,000
K-means-GA - ACS	20,700	1,431	1,000
K-means-GA - GA	106,480	7,362	**,000**
K-means-GA - PSO	157,580	10,896	**,000**
K-means-ACS - ACS	17,560	1,214	1,000
K-means-ACS - GA	103,340	7,145	**,000**
K-means-ACS - PSO	154,440	10,679	**,000**
ACS - GA	85,780	5,931	**,000**
ACS - PSO	136,880	9,464	**,000**
GA - PSO	-51,100	-3,533	**,004**

Tabela 29 – Cenário (j): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (j). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-GA - ACS	16,960	1,173	1,000
K-means-GA - K-means-ACS	-21,180	-1,464	1,000
K-means-GA - GA	111,540	7,712	**,000**
K-means-GA - PSO	162,420	11,230	**,000**
ACS - K-means-ACS	-4,220	-,292	1,000
ACS - GA	94,580	6,540	**,000**
ACS - PSO	145,460	10,058	**,000**
K-means-ACS - GA	90,360	6,248	**,000**
K-means-ACS -PSO	141,240	9,766	**,000**
GA - PSO	-50,880	-3,518	**,004**

Tabela 30 – Cenário (k): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (k). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-GA - K-means-ACS	-,200	-,014	1,000
K-means-GA -ACS	29,740	2,056	,397
K-means-GA - GA	109,980	7,604	**,000**
K-means-GA - PSO	159,980	11,062	**,000**
K-means-ACS - ACS	29,540	2,043	,411
K-means-ACS - GA	109,780	7,591	**,000**
K-means-ACS - PSO	159,780	11,048	**,000**
ACS - GA	80,240	5,548	**,000**
ACS - PSO	103,240	9,005	**,000**
GA - PSO	-50,000	-3,457	**,005**

Tabela 31 – Cenário (l): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (l). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-ACS - K-means-GA	3,800	,263	1,000
K-means-ACS - ACS	54,520	3,770	**,002**
K-means-ACS - GA	119,440	8,259	**,000**
K-means-ACS - PSO	169,440	11,716	**,000**
K-means-GA - ACS	50,720	3,507	**,005**
K-means-GA -GA	115,640	7,996	**,000**
K-means-GA - PSO	165,640	11,453	**,000**
ACS - GA	64,920	4,489	**,000**
ACS - PSO	114,920	7,946	**,000**
GA - PSO	-50,000	-3,457	**,005**

Tabela 32 – Cenário (m): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (m). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

A.3 Gráficos de Análise da Convergência dos Algoritmos

Os gráficos boxplot a seguir apresentam a distribuição do número de iterações que cada algoritmo gastou até atingir a convergência.

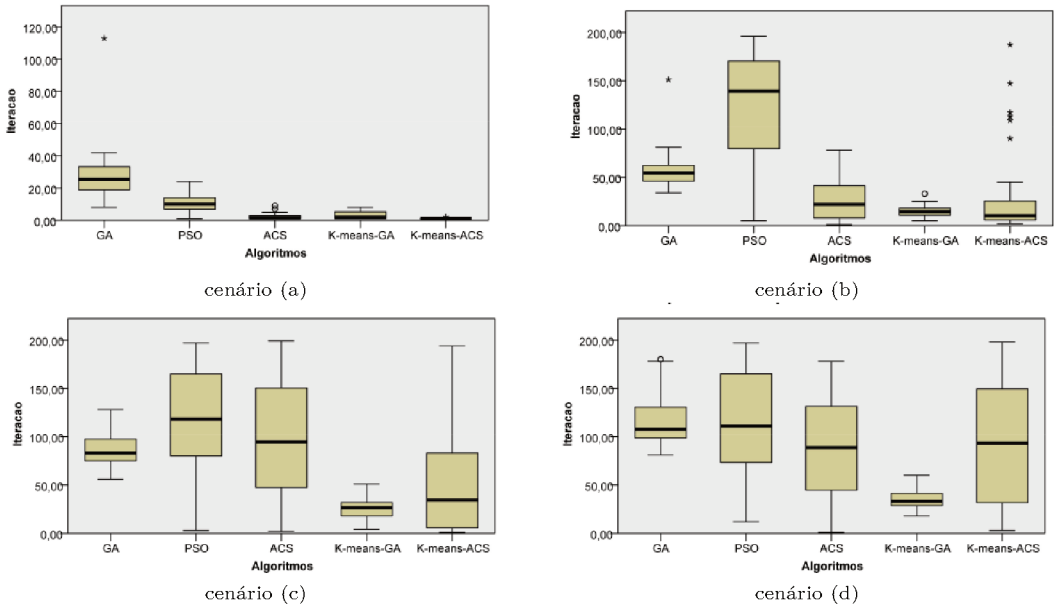
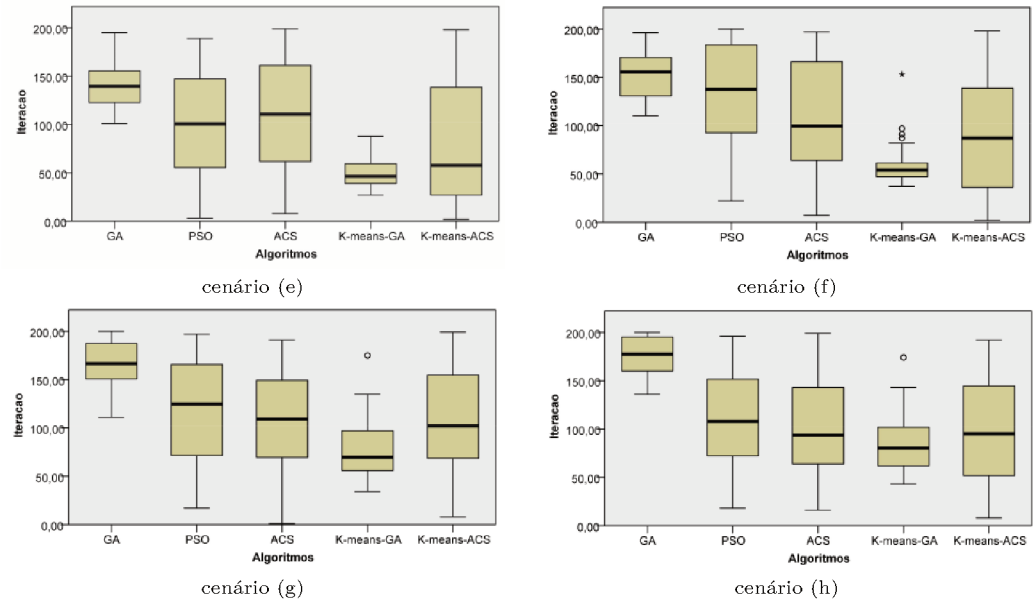


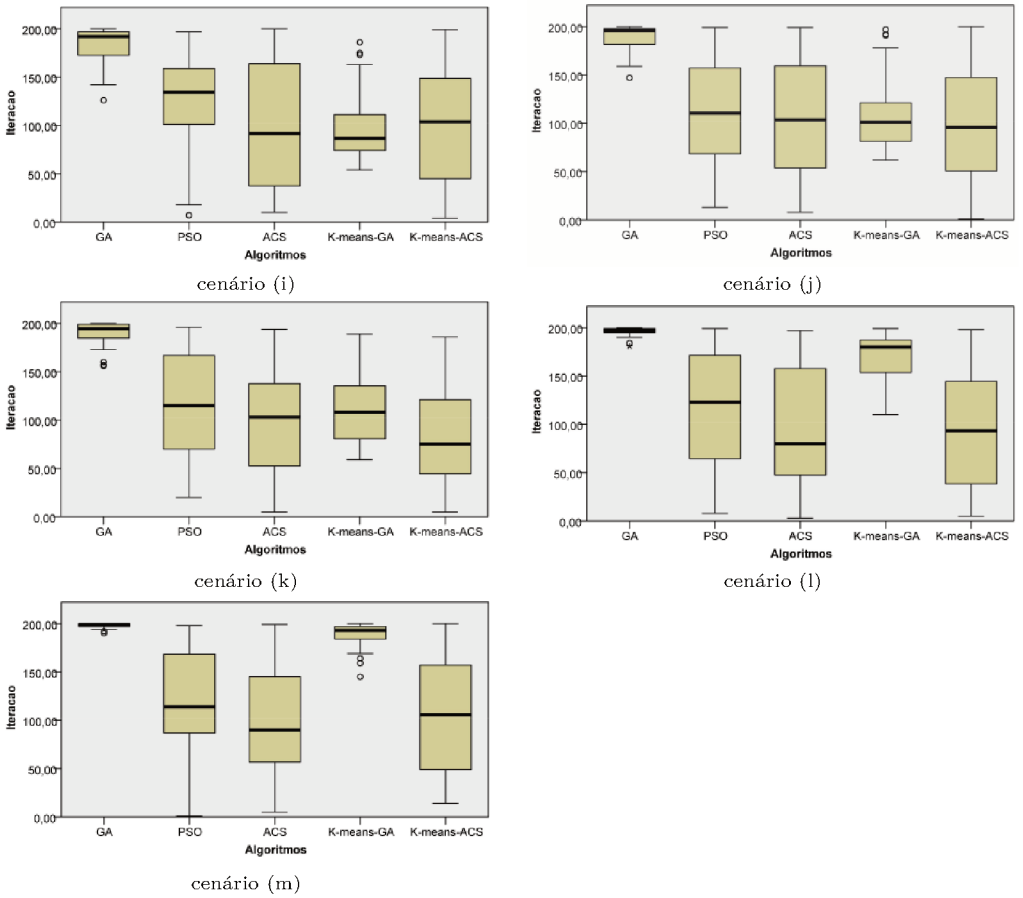
Figura 33 – Convergência no experimento 2. Cenários (a) ao (d). Eixo Y de cada gráfico boxplot mostra o número de iterações requeridos para chegar a convergência. Eixo X apresenta os algoritmos.



cenário (e) cenário (f)

cenário (g) cenário (h)

Figura 34 – Convergência no experimento 2. Cenários (e) ao (h). Eixo Y de cada gráfico boxplot mostra o número de iterações requeridos para chegar a convergência. Eixo X apresenta os algoritmos.



cenário (i) cenário (j)

cenário (k) cenário (l)

cenário (m)

Figura 35 – Convergência no experimento 2. Cenários (i) ao (m). Eixo Y de cada gráfico boxplot mostra o número de iterações requeridos para chegar a convergência. Eixo X apresenta os algoritmos.

A.4 Tabelas com os Resultados Estatísticos da Convergência dos Algoritmos

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-ACS - ACS	33,510	2,375	,175
K-means-ACS - K-means-GA	49,880	3,535	**,004**
K-means-ACS - PSO	120,920	8,570	**,000**
K-means-ACS - GA	172,290	12,211	**,000**
ACS - K-means-GA	-16,370	-1,160	1,000
ACS - PSO	87,410	6,195	**,000**
ACS - GA	138,780	9,836	**,000**
K-means-GA - PSO	71,040	5,035	**,000**
K-means-GA - GA	122,410	8,676	**,000**
PSO - GA	51,370	3,641	**,003**

Tabela 33 – Cenário (a): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (a). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-GA - K-means-ACS	-1,830	-,127	1,000
K-means-GA - ACS	22,290	1,542	1,000
K-means-GA - GA	95,150	6,580	**,000**
K-means-GA - PSO	129,880	8,982	**,000**
K-means-ACS - ACS	20,460	1,415	1,000
K-means-ACS - GA	93,320	6,454	**,000**
K-means-ACS - PSO	128,050	8,856	**,000**
ACS - GA	72,860	5,039	**,000**
ACS - PSO	107,590	7,441	**,000**
GA - PSO	-34,730	-2,402	,163

Tabela 34 – Cenário (b): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (b). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-GA - K-means-ACS	-34,130	-2,360	,183
K-means-GA - ACS	93,900	6,493	**,000**
K-means-GA - GA	97,490	6,741	**,000**
K-means-GA - PSO	124,430	8,604	**,000**
K-means-ACS - ACS	59,770	4,133	**,000**
K-means-ACS - GA	63,360	4,381	**,000**
K-means-ACS - PSO	90,300	6,244	**,000**
ACS - GA	3,590	,248	1,000
ACS - PSO	30,530	2,111	,348
GA - PSO	-26,940	-1,863	,625

Tabela 35 – Cenário (c): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (c). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-GA - ACS	72,430	5,008	**,000**
K-means-GA - K-means-ACS	-79,570	-5,502	**,000**
K-means-GA - PSO	109,760	7,590	**,000**
K-means-GA - GA	114,090	7,889	**,000**
ACS - K-means-ACS	-7,140	-,494	1,000
ACS - PSO	37,330	2,581	**,040**
ACS - GA	41,660	2,881	**,040**
K-means-ACS - PSO	30,190	2,088	,368
K-means-ACS - GA	34,520	2,387	,170
PSO - GA	4,330	,299	1,000

Tabela 36 – Cenário (d): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (d). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-GA - K-means-ACS	-35,920	-2,484	,130
K-means-GA - PSO	59,560	4,118	**,000**
K-means-GA - ACS	72,950	5,044	**,000**
K-means-GA - GA	113,670	7,860	**,000**
K-means-ACS - PSO	23,640	1,635	1,000
K-means-ACS - ACS	37,030	2,561	,105
K-means-ACS - GA	77,750	5,376	**,000**
PSO - ACS	-13,390	-,926	1,000
PSO - GA	54,110	3,742	**,002**
ACS - GA	40,720	2,816	**,049**

Tabela 37 – Cenário (e): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (e). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-GA - K-means-ACS	-39,340	-2,720	,065
K-means-GA - ACS	65,420	4,524	**,000**
K-means-GA - PSO	93,700	6,479	**,000**
K-means-GA - GA	119,840	8,287	**,000**
K-means-ACS - ACS	26,080	1,803	,713
K-means-ACS - PSO	54,360	3,759	**,002**
K-means-ACS - GA	80,500	5,566	**,000**
ACS - PSO	28,280	1,955	,505
ACS - GA	54,420	3,763	**,002**
PSO - GA	26,140	1,808	,707

Tabela 38 – Cenário (f): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (f). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-GA - K-means-ACS	-40,540	-2,803	,051
K-means-GA - ACS	43,640	3,018	**,025**
K-means-GA - PSO	55,670	3,849	**,001**
K-means-GA - GA	122,100	8,443	**,000**
K-means-ACS - ACS	3,100	,214	1,000
K-means-ACS - PSO	15,130	1,046	1,000
K-means-ACS - GA	81,560	5,640	**,000**
ACS - PSO	12,030	,832	1,000
ACS - GA	78,460	5,425	**,000**
PSO - GA	66,430	4,593	**,000**

Tabela 39 – Cenário (g): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (g). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-GA - K-means-ACS	-18,640	-1,289	1,000
K-means-GA - ACS	26,810	1,854	,638
K-means-GA - PSO	36,270	2,508	,121
K-means-GA - GA	123,480	8,538	**,000**
K-means-ACS - ACS	8,170	,565	1,000
K-means-ACS - PSO	17,630	1,219	1,000
K-means-ACS - GA	104,840	7,249	**,000**
ACS - PSO	9,460	,654	1,000
ACS - GA	96,670	6,684	**,000**
PSO - GA	87,210	6,030	**,000**

Tabela 40 – Cenário (h): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (h). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-GA - K-means-ACS	-3,580	-248	1,000
K-means-GA - ACS	4,790	,331	1,000
K-means-GA - PSO	36,650	2,534	,113
K-means-GA - GA	114,230	7,899	**,000**
K-means-ACS - ACS	1,210	,084	1,000
K-means-ACS - PSO	33,070	2,287	,222
K-means-ACS - GA	110,650	7,651	**,000**
ACS - PSO	31,860	2,203	,276
ACS - GA	109,440	7,568	**,000**
PSO - GA	77,580	5,365	**,000**

Tabela 41 – Cenário (i): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (i). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-ACS - ACS	6,440	,445	1,000
K-means-ACS - K-means-GA	6,980	,483	1,000
K-means-ACS - PSO	11,540	,798	1,000
K-means-ACS - GA	116,690	8,069	**,000**
ACS - K-means-GA	-,540	-,037	1,000
ACS - PSO	5,100	,353	1,000
ACS - GA	110,250	7,624	**,000**
K-means-GA - PSO	4,560	,315	1,000
K-means-GA - GA	109,710	7,586	**,000**
PSO - GA	105,150	7,271	**,000**

Tabela 42 – Cenário (j): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (j). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-ACS - ACS	19,230	1,330	1,000
K-means-ACS - K-means-GA	35,990	2,489	,128
K-means-ACS - PSO	37,630	2,602	,093
K-means-ACS - GA	139,750	9,664	**,000**
ACS - K-means-GA	-16,760	-1,159	1,000
ACS - PSO	18,400	1,272	1,000
ACS - GA	120,520	8,334	**,000**
K-means-GA - PSO	1,640	,113	1,000
K-means-GA - GA	103,760	7,175	**,000**
PSO - GA	102,120	7,062	**,000**

Tabela 43 – Cenário (k): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (k). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-ACS - ACS	2,500	,173	1,000
K-means-ACS - PSO	20,670	1,430	1,000
K-means-ACS - K-means-GA	73,840	5,107	**,000**
K-means-ACS - GA	138,690	9,592	**,000**
ACS - PSO	18,170	1,257	1,000
ACS - K-means-GA	-71,340	-4,934	**,000**
ACS - GA	136,190	9,419	**,000**
PSO - K-means-GA	-53,170	-3,677	**,002**
PSO - GA	118,020	8,162	**,000**
K-means-GA - GA	64,850	4,485	**,000**

Tabela 44 – Cenário (l): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (l). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
ACS - K-means-ACS	-10,320	-,714	1,000
ACS - PSO	16,810	1,163	1,000
ACS - K-means-GA	-100,830	-6,976	**,000**
ACS - GA	142,490	9,858	**,000**
K-means-ACS - PSO	6,490	,499	1,000
K-means-ACS - K-means-GA	90,510	6,262	**,000**
K-means-ACS - GA	132,170	9,144	**,000**
PSO - K-means-GA	-84,020	-5,813	**,000**
PSO - GA	125,680	8,695	**,000**
K-means-GA - GA	41,660	2,882	**,039**

Tabela 45 – Cenário (m): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (m). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

APÊNDICE B

Imagens e Tabelas do Experimento 3

B.1 Configurações dos Cenários do Experimento

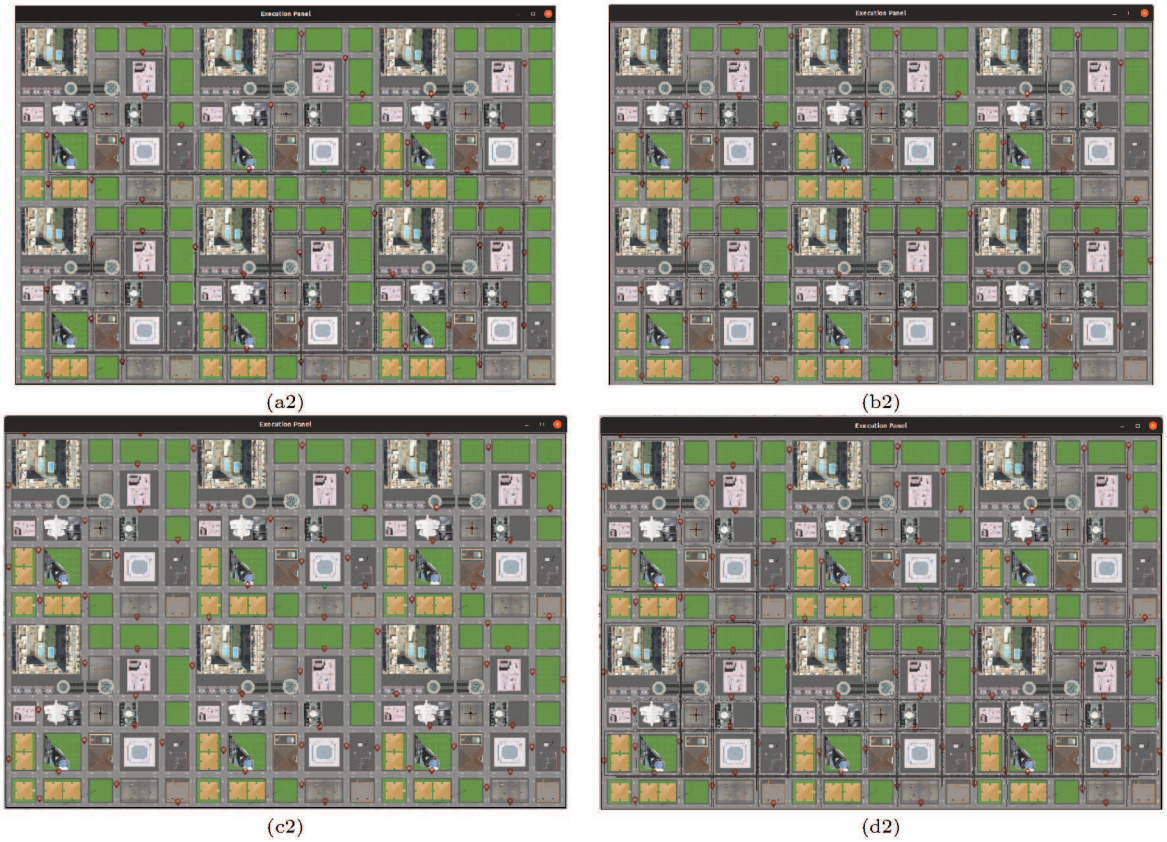


Figura 36 – Configuração dos quatro cenários definidos para o escalonamento de rotas no mapa da cidade virtual 2.

B.2 Tabelas com os Resultados Estatísticos da Minimização de *Fitness*

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-ACS - ACS	36,360	2,514	,119
K-means-ACS - K-means-GA	93,200	6,444	**,000**
K-means-ACS - GA	143,160	9,899	**,000**
K-means-ACS - PSO	193,180	13,357	**,000**
ACS - K-means-GA	-58,840	-3,930	**,001**
ACS - GA	106,800	7,385	**,000**
ACS - PSO	156,820	10,843	**,000**
K-means-GA - GA	49,960	3,454	**,006**
K-means-GA - PSO	99,980	6,913	**,000**
GA - PSO	-50,020	-3,459	**,005**

Tabela 46 – Cenário (a2): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (a2). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
ACS - K-means-ACS	-44,580	-3,082	**,021**
ACS - K-means-GA	-93,120	-6,439	**,000**
ACS - GA	145,900	10,088	**,000**
ACS - PSO	195,900	13,545	**,000**
K-means-ACS - K-means-GA	48,540	3,356	**,008**
K-means-ACS - GA	101,320	7,006	**,000**
K-means-ACS - PSO	151,320	10,463	**,000**
K-means-GA - GA	52,780	3,649	**,003**
K-means-GA - PSO	102,780	7,107	**,000**
GA - PSO	-50,000	-3,457	**,005**

Tabela 47 – Cenário (b2): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (b2). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
ACS - K-means-ACS	-49,680	-3,435	**,006**
ACS - K-means-GA	-99,720	-6,895	**,000**
ACS - GA	149,800	10,358	**,000**
ACS - PSO	199,800	13,815	**,000**
K-means-ACS - K-means-GA	50,040	3,460	**,005**
K-means-ACS - GA	100,120	6,923	**,000**
K-means-ACS - PSO	150,120	10,380	**,000**
K-means-GA - GA	50,080	3,463	**,005**
K-means-GA - PSO	100,080	6,920	**,000**
GA - PSO	-50,000	-3,457	**,005**

Tabela 48 – Cenário (c2): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (c2). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
ACS - K-means-ACS	-45,720	-3,161	**,016**
ACS - K-means-GA	-97,860	-6,766	**,000**
ACS - GA	147,860	10,224	**,000**
ACS - PSO	197,860	13,681	**,000**
K-means-ACS - K-means-GA	52,140	3,605	**,003**
K-means-ACS - GA	102,140	7,062	**,000**
K-means-ACS - PSO	152,140	10,520	**,000**
K-means-GA - GA	50,000	3,457	**,005**
K-means-GA - PSO	100,000	6,914	**,000**
GA - PSO	-50,000	-3,457	**,005**

Tabela 49 – Cenário (d2): Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (d2). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

B.3 Gráficos de Análise da Convergência dos Algoritmos

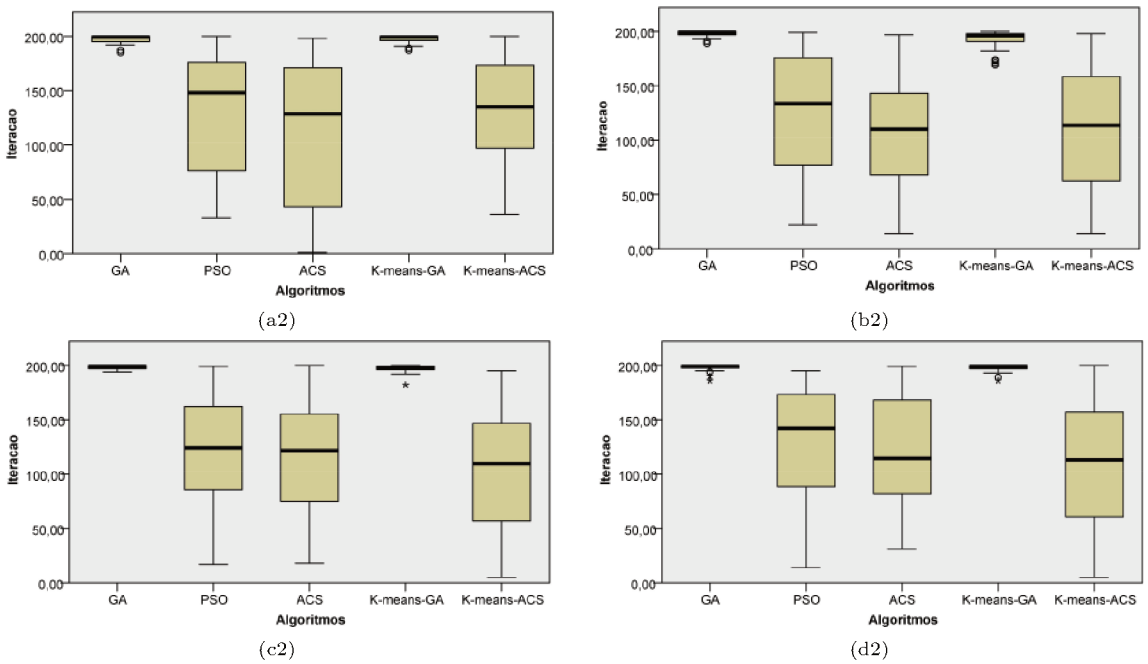


Figura 37 – Convergência no experimento 3. Eixo Y de cada gráfico boxplot mostra o número de iterações requeridas para chegar a convergência. Eixo X apresenta os algoritmos.

B.4 Tabelas com os Resultados Estatísticos da Convergência dos Algoritmos

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
ACS - K-means-ACS	-12,960	-,898	1,000
ACS - PSO	14,390	,998	1,000
ACS - GA	126,240	8,752	**,000**
ACS - K-means-GA	-130,210	-9,027	**,000**
K-means-ACS - PSO	1,430	,099	1,000
K-means-ACS - GA	113,280	7,853	**,000**
K-means-ACS - K-means-GA	117,250	8,128	**,000**
PSO - GA	111,850	7,754	**,000**
PSO - K-means-GA	-115,820	-8,029	**,000**
GA - K-means-GA	-3,970	-,275	1,000

Tabela 50 – Cenário (a2): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (a2). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
ACS - K-means-ACS	-6,690	-,463	1,000
ACS - PSO	19,800	1,370	1,000
ACS - K-means-GA	-108,760	-7,526	**,000**
ACS - GA	139,700	9,667	**,000**
K-means-ACS - PSO	13,110	,907	1,000
K-means-ACS - K-means-GA	102,070	7,063	**,000**
K-means-ACS - GA	133,010	9,204	**,000**
PSO - K-means-GA	-88,960	-6,156	**,000**
PSO - GA	119,900	8,297	**,000**
K-means-GA - GA	30,940	2,141	,323

Tabela 51 – Cenário (b2): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (b2). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-ACS - ACS	9,290	,643	1,000
K-means-ACS - PSO	16,650	1,153	1,000
K-means-ACS - K-means-GA	123,280	8,538	**,000**
K-means-ACS - GA	136,030	9,421	**,000**
ACS - PSO	7,360	,510	1,000
ACS - K-means-GA	-113,990	-7,895	**,000**
ACS - GA	126,740	8,778	**,000**
PSO - K-means-GA	-106,630	-7,385	**,000**
PSO - GA	119,380	8,268	**,000**
K-means-GA - GA	12,750	,883	1,000

Tabela 52 – Cenário (c2): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (c2). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

Grupo1 - Grupo2	Estatística de Teste	Desvio Padrão da Estatística de Teste	*valor-p*
K-means-ACS - ACS	5,870	,407	1,000
K-means-ACS - PSO	9,000	,624	1,000
K-means-ACS - K-means-GA	123,020	8,525	**,000**
K-means-ACS - GA	126,760	8,784	**,000**
ACS - PSO	3,130	,217	1,000
ACS - K-means-GA	-117,150	-8,118	**,000**
ACS - GA	120,890	8,377	**,000**
PSO - K-means-GA	-114,020	-7,901	**,000**
PSO - GA	117,760	8,161	**,000**
K-means-GA - GA	3,740	,259	1,000

Tabela 53 – Cenário (d2): Análise de convergência. Teste de *Dunn-Bonferroni* aplicado aos resultados do cenário (d2). Os valores em negrito indicam que há diferença significativa considerando um nível de significância de $0,05$.

B.5 Equipe do LRM e CaRINA II

Os estudo de caso realizado no campus da USP teve auxilio da equipe do LRM (Figura 38).

Aluno e Orientador - UFU Equipe LRM - ICMC/USP

Figura 38 – Fotos com o CaRINA II no LRM ICMC/USP.