Frederico de Castro Neto

Implementação da Heurística de Lin-Kernighan e sua Aplicação no Sequenciamento de Pontos de Rebitagem

Bauru - SP

Maio de 2023

Frederico de Castro Neto

Implementação da Heurística de Lin-Kernighan e sua Aplicação no Sequenciamento de Pontos de Rebitagem

Dissertação de Mestrado

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Engenharia de Bauru

Programa de Pós-Graduação em Engenharia de Produção

Orientadora: Prof^a. Dr^a. Edilaine Martins Soler

Coorientadora: Prof^a. Dr^a. Adriana Cristina Cherri Nicola

Bauru - SP

Maio de 2023

C355i

Castro Neto, Frederico de

Implementação da heurística de Lin-Kernighan e sua aplicação no sequenciamento de pontos de rebitagem / Frederico de Castro Neto. -- Bauru, 2023

113 f. : il., tabs.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Faculdade de Engenharia, Bauru

Orientadora: Edilaine Martins Soler

Coorientadora: Adriana Cristina Cherri Nicola

1. Modelos matemáticos. 2. Otimização matemática. 3. Métodos heurísticos. 4. Heurística de Lin-Kernighan. 5. Problema do caixeiro viajante. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Engenharia, Bauru. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Bauru

ATA DA DEFESA PÚBLICA DA DISSERTAÇÃO DE MESTRADO DE FREDERICO DE CASTRO NETO, DISCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO, DA FACULDADE DE ENGENHARIA - CÂMPUS DE BAURU.

Aos 29 dias do mês de junho do ano de 2023, às 14:00 horas, por meio de Videoconferência, realizou-se a defesa de DISSERTAÇÃO DE MESTRADO de FREDERICO DE CASTRO NETO, intitulada **IMPLEMENTAÇÃO DA HEURÍSTICA DE LIN-KERNIGHAN E SUA APLICAÇÃO NO SEQUENCIAMENTO DE PONTOS DE REBITAGEM**. A Comissão Examinadora foi constituida pelos seguintes membros: Profa. Dra. EDILAINE MARTINS SOLER (Orientador(a) - Participação Virtual) do(a) Departamento de Matematica / Faculdade de Ciencias de Bauru - UNESP, Prof. Dr. SILVIO ALEXANDRE DE ARAUJO (Participação Virtual) do(a) Departamento de Matemática / Universidade Estadual Paulista Julio de Mesquita Filho - UNESP - Câmpus de São José do Rio Preto, Prof. Dr. PEDRO AUGUSTO MUNARI JUNIOR (Participação Virtual) do(a) Departamento de Engenharia de Produção / Universidade Federal de São Carlos. Após a exposição pelo mestrando e arguição pelos membros da Comissão Examinadora que participaram do ato, de forma presencial e/ou virtual, o discente recebeu o conceito final:_ APROVADO _ . Nada mais havendo, foi lavrada a presente ata, que após lida e aprovada, foi assinada pelo(a) Presidente(a) da Comissão Examinadora.

Edilaine M. Soler

Profa. Dra. EDILAINE MARTINS SOLER

Dedicado aos pesquisadores do Problema do Caixeiro Viajante,
que em meio a um infinito de possibilidades,
nunca desistiram de sonhar.

AGRADECIMENTOS

Agradeço aos meus pais, Maria Luiza Rodrigues Pereira de Castro e Nilton Jesus de Castro, por me escolherem como objeto de amor, carinho e educação ao longo de todos os anos de minha vida. Sem o apoio de vocês e de nossa família eu não seria nada.

À minha esposa, Aline Borgatto França de Castro, pela compreensão e paciência durante um período em que não estive totalmente presente, suportando minha escolha pela dedicação acadêmica.

À minha orientadora, professora Edilaine Martins Soler, pela introdução ao mundo da Pesquisa Operacional e otimização e por ter apoiado e direcionado este trabalho a partir do seu conhecimento e experiência profissional.

À minha coorientadora, professora Adriana Cristina Cherri Nicola, e ao professor Silvio Alexandre de Araujo pelo suporte e colaboração prestados para o desenvolvimento e conclusão deste trabalho.

À EMBRAER, Empresa Brasileira de Aeronáutica, por acreditar no meu potencial, me apoiar em projetos desafiadores e patrocinar este estudo, em especial a todos os amigos do setor de Engenharia de Produção e Montagem que fizeram parte do meu crescimento profissional e aos supervisores Rodrigo Oliveira Leme e Marco Aurélio Moreira pela oportunidade de fazer parte de uma das maiores empresas do setor aeronáutico do mundo.

À Faculdade de Engenharia de Bauru, universidade composta por competentes professores e profissionais que me apoiaram durante a graduação em Engenharia Mecânica e que continuam me apoiando durante essa nova fase de estudos e aprimoramento científico.

"O Problema do Caixeiro Viajante
não é um problema, é um vício."
(Christos H. Papadimitriou, 1991)

RESUMO

O meio industrial moderno e seus processos de manufatura levam fabricantes a um ambiente competitivo, no qual a busca pela melhoria contínua e a excelência produtiva são peças fundamentais para o sucesso de seus negócios. No setor aeronáutico, o processo de manufatura conhecido como rebitagem ocupa um lugar de destaque, justificado por sua representatividade nos roteiros de produção e sua importância para a qualidade estrutural e segurança de voo do avião. Assim sendo, otimizar o ciclo produtivo agregado a esse processo se torna objetivo primário de fabricantes, a fim de reduzir o custo final de seus produtos. Neste contexto, este trabalho propõe o desenvolvimento de uma aplicação computacional para otimizar o sequenciamento de rebites instalados por máquinas de rebitagem automática de modo a automatizar a tarefa de sequenciamento e reduzir o tempo de trabalho destes equipamentos. Para isso, o problema é modelado como o Problema do Caixeiro Viajante e resolvido através da heurística de Lin-Kernighan, a qual foi implementada computacionalmente. Testes numéricos utilizando instâncias acadêmicas e industriais foram realizados a fim de validar o modelo de otimização proposto e comprovar a eficiência da heurística implementada. Os resultados numéricos obtidos comprovaram a eficiência do sistema computacional desenvolvido e o potencial do mesmo para o sequenciamento de rebites.

Palavras-chave: Problema do Caixeiro Viajante, Rebitagem Automatizada, Heurística de Lin-Kernighan.

ABSTRACT

Modern industrial environment and its manufacturing processes lead manufacturers to a competitive scenario, in which the demand for continuous improvement and productive excellence are primal issues for the success of their businesses. In aerospace industry, the manufacturing process known as fastening occupies a prominent role, justified by its relevance in production routes and its importance for the structural quality and flight safety of the aircraft. Therefore, optimizing the manufaturing cycle of this process becomes the main objective of manufacturers, in order to reduce the final cost of their products. In this context, this work proposes the development of an application to optimize the sequencing of fasteners installed by automatic riveting machines in order to automate the sequencing task and reduce the equipment working hours. The problem is modelled as the Travelling Salesman Problem and solved using Lin-Kernighan Heuristic, which was implemented. Numerical tests using academic and industrial instances were performed aiming to validate the proposed optimization model and prove the efficiency of implemented heuristic. Numerical results obtained confirmed the efficiency of implemented computational system and its potential for rivet sequencing.

Keywords: Travelling Salesman Problem, Automatic Fastening, Lin-Kernighan Heuristic.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AN	*Aeronautical Standard*
CAM	*Computer-Aided Manufacturing*
CDP	Corpo de Prova
CLK	*Chained Lin-Kernighan*
CN	Controle Numérico
CNC	Controle Numérico Computadorizado
CRFP	*Carbon Fiber Reinforced Polymers*
CTSP	*Colored Travelling Salesman Problem*
DRL	*Deep Reinforcement Learning*
GA	*Genetic Algorithm*
GLN	*Graph Learning Network*
GRASP	*Greedy Randomized Adaptive Search Procedure*
IA	Inteligência Artificial
IP	*Integer Programming*
LKH	*Lin-Kernighan-Helsgaun Heuristic*
MS	*Military Standard*
MCTS	*Monte Carlo Tree Search*
NAS	*National Aeronautical Standard*
NLP	*Natural Language Processing*
PCB	*Printed Circuit Board*
PCV	Problema do Caixeiro Viajante
RL	*Reinforcement Learning*
TSPCN	*Traveling Salesman Problem with Circular Neighborhood*
VBA	*Visual Basic for Applications*

SUMÁRIO

1 INTRODUÇÃO

A partir da Revolução Industrial e o nascimento da indústria, a humanidade passou por uma transformação socioeconômica na qual se destaca, no campo dos negócios, a competição produtiva, uma vez que foi responsável pela evolução científica e tecnológica ao longo do século passado até os dias atuais. Como parte dessa evolução, o ramo da automação industrial ganhou destaque no meio fabril, a partir da busca pela eficiência produtiva e redução de custos de fabricação. Com o advento de máquinas e robôs automatizados, fabricantes puderam multiplicar seus índices produtivos em grande escala e, ao mesmo tempo, incentivar o desenvolvimento desse segmento. Segundo Terzic et al. (2008) o mercado mundial de automação em 2004 teve um volume total de 121,8 bilhões de euros com um crescimento anual até o final da década de aproximadamente 2,8%. Esse mesmo mercado, foi cotado em 196,6 bilhões de dólares em 2021, com prospecção de alcançar 412,8 bilhões de dólares até 2030 (Precedence Research, 2021).

Em paralelo ao desenvolvimento industrial, um outro movimento nascia em meados da década de 40: a Pesquisa Operacional. Esse campo de estudo utiliza modelos matemáticos e estatísticos com objetivo de construir sistemas de auxílio para tomadas de decisão, possuindo uma conexão direta com o mundo industrial, a qual colaborou com o desenvolvimento de ferramentas geradoras de resultados significativos na cadeia produtiva mundial (XING et al., 2013).

No setor aeronáutico o cenário competitivo entre fabricantes é extremamente agressivo, uma vez que a fabricação de aeronaves é um processo de manufatura complexo cujo resultado envolve vidas humanas. Mesmo em um segmento onde a fabricação é realizada em menor escala, diversas soluções de automação fabril foram propostas e desenvolvidas ao longo dos anos, das quais se destacam soluções de automação para rebitagem, uma vez que este é o principal processo de união de componentes estruturais aeronáuticos na atualidade. Porém, por se tratar de um equipamento de nicho aeronáutico, poucos estudos foram desenvolvidos com a proposta de otimizar o seu uso, fato contraditório dada sua relevância tecnológica e financeira.

Cheng et al. (2012) realizou um estudo de modelagem de gabaritos para rebitadoras automáticas com objetivo de otimizar a rebitagem e diminuir a variação posicional dos respectivos rebites instalados. Wei et al. (2013) propôs um algoritmo de normalização angular a fim de acelerar o processo fabril e garantir o correto ângulo de furação do equipamento. Merluzzi e Bahr (2017) desenvolveram um trocador automático das ferramentas de conformação dos rebites, reduzindo assim o tempo de *setup* manual agregado ao processo de rebitagem automatizada.

A proposta deste trabalho é modelar e desenvolver um método de otimização a ser aplicado no sequenciamento de pontos de rebitagem, a partir da adaptação do Problema do Caixeiro Viajante e sua respectiva resolução, de modo a aumentar a performance do equipamento robotizado. Do ponto de vista computacional, este trabalho também tem como objetivo realizar a implementação da heurística de Lin-Kernighan, uma das principais heurísticas aplicadas para a resolução do Problema do Caixeiro Viajante, conforme primeira publicação de Kernighan e Lin em 1973, documentá-la e distribuí-la para a comunidade científica, incentivando assim o estudo deste método e de suas ramificações surgidas ao longo dos anos. Tal método heurístico será utilizado para resolução das instâncias estudadas neste trabalho, a fim de validar sua eficiência.

A implementação computacional foi feita na linguagem de programação Python e os testes numéricos iniciais foram realizados em instâncias acadêmicas do Problema do Caixeiro Viajante, buscando assim a comprovação da eficiência do algoritmo implementado. Posteriormente, pequenas adaptações foram realizadas para execução dos testes com instâncias industriais, nos quais os resultados foram comparados com o sequenciamento em operação no equipamento de rebitagem.

Este trabalho é divido em oito capítulos. O Capítulo 1 apresenta a introdução e a base do trabalho. O Capítulo 2 aborda o histórico e a modelagem matemática do Problema do Caixeiro Viajante e os motivos pelos quais este trabalho propõe sua implementação. O Capítulo 3 introduz os conceitos básicos do processo de rebitagem e as motivações para aplicação do modelo de otimização no processo de sequenciamento de rebites. O Capítulo 4 demonstra entraves para implementação da heurística de Lin-Kernighan enquanto que o Capítulo 5 detalha a sua respectiva implementação computacional. No Capítulo 6 a coleta das instâncias acadêmicas e os desenvolvimentos necessários para a formação das instâncias industriais são detalhados. O Capítulo 7 apresenta os principais resultados numéricos das simulações realizadas. O Capítulo 8 desenvolve as conclusões e considerações a respeito do estudo desenvolvido e propõe trabalhos futuros acerca da temática abordada.

2 PROBLEMA DO CAIXEIRO VIAJANTE

Neste capítulo serão apresentados os principais conceitos do Problema do Caixeiro Viajante a partir da sua origem histórica, a modelagem matemática do problema e a revisão bibliográfica desenvolvida em torno de trabalhos recentes sobre o mesmo.

2.1 Um breve histórico

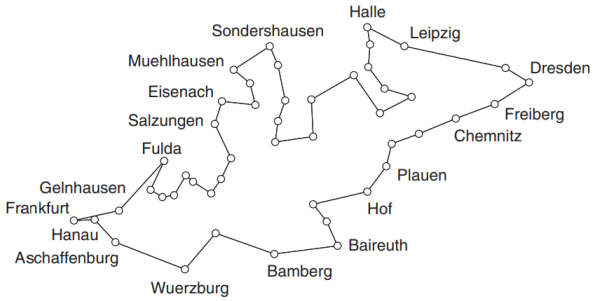
O Problema do Caixeiro Viajante é um dos principais problemas do campo da otimização combinatória justificado pela sua relevância em diversas áreas do conhecimento e a sua curiosa complexidade de resolução. Tal característica foi a motivação necessária para que pesquisadores ocasionalmente desenvolvessem métodos inovadores como, por exemplo, o Método dos Planos de Corte e a heurística Recozimento Simulado. No presente momento, o PCV ainda é motivo de interesse de estudiosos, tanto por sua aplicabilidade em diversos campos da ciência como pela busca por métodos alternativos de solução do problema.

O nome do problema remete especificamente aos caixeiros, viajantes comerciais responsáveis por visitar cidades realizando cobranças e encomendas, os quais planejavam estrategicamente suas rotas para encurtar o período de suas missões. Uma das primeiras referências escritas de tal atividade foi documentada em 1832 em um manual alemão contendo rotas otimizadas na região da Alemanha e Suíça, conforme ilustrado na Figura 1. Além dos caixeiros, Cook (2011) também destaca a atuação de advogados e padres norte-americanos no século 19 em suas viagens profissionais e a preocupação dos mesmos em obter rotas reduzidas.

Do ponto de vista matemático, o PCV tem suas origens na década de 30, quando Karl Menger descreve no Colóquio Matemático de Viena o então chamado Problema do Mensageiro como “a tarefa, dado um número finito de pontos com distância emparelhada conhecida, de encontrar o menor caminho conectando esses pontos”. No final da mesma década, Merrill Flood e o seu estudo de rotas de ônibus escolares popularizou e trouxe atenção ao PCV no continente americano (COOK, 2011).

A origem do nome do problema é indefinida, sendo a publicação de Robinson (1949) o documento mais antigo em que tal nome é encontrado, conforme imagem da Figura 2. Nesta ocasião o grupo RAND, uma corporação norte americana de auxílio à pesquisa e desenvolvimento, oferecia um prêmio em dinheiro para a resolução de diversos problemas, dos quais o PCV era um dos destaques.

Figura 1 – Rota do caixeiro viajante em 1832.



Fonte: (COOK, 2011).

Figura 2 – A publicação de Robinson e o nome do PCV.


U.S. AIR FORCE Project RAND

ON THE HAMILTONIAN GAME
(A Traveling Salesmen Problem)

Julia Robinson

RM-303

5 December 1949 Copy No. 70

Fonte: (ROBINSON, 1949).

A partir da década de 50, o PCV se tornou um problema conhecido em grande escala de forma que diversos pesquisadores estudaram e desenvolveram novos métodos de resolução para o mesmo. Uma das principais bases propostas nessa década foi a publicação de Dantzig, Fulkerson e Johnson (1954), na qual os autores utilizaram os conceitos da programação linear desenvolvida pelo próprio Dantzig combinado com um novo método denominado Método dos Planos de Corte. A partir de tal método, os autores demonstraram matematicamente pela primeira vez a otimalidade de uma rota complexa construída a partir dos estados norte americanos.

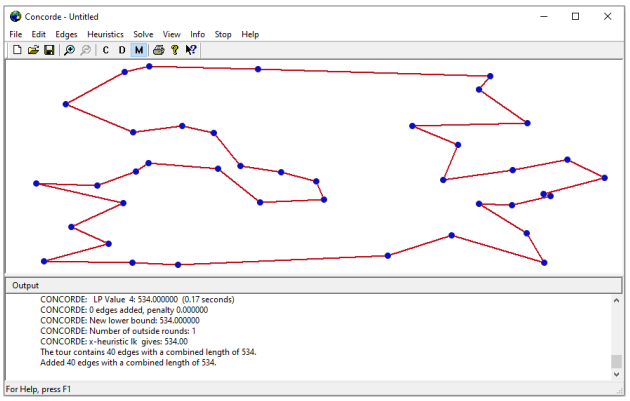
Na década de 70 dois trabalhos se destacaram entre métodos de solução aproximada do PCV: Kernighan e Lin (1973) desenvolveram a heurística de Lin-Kernighan, método tema deste trabalho, o qual é baseado em uma estratégia adaptável de troca de arestas a partir de uma rota inicial. Este método influenciou diversos pesquisadores e será mais bem detalhado no Capítulo 4. Um segundo método heurístico de destaque nesta década foi proposto por Christofides (1976), a partir da utilização do conceito de árvores de extensão mínima. Este método converge, no máximo, em uma solução com 3/2 vezes o custo da solução ótima global, sendo o melhor método de resolução do PCV em tempo polinomial até os dias atuais.

A partir da publicação de Kernighan e Lin (1973), outras duas publicações foram destaques no final da década de 90 e começo dos anos 2000 por apresentarem variações metodológicas que resultaram em uma significante evolução performática do método original. Helsgaun (2000) desenvolveu um algoritmo complexo denominado heurística de Lin-Kernighan-Helsgaun (LKH) a partir de refinamentos e expansões do método de busca por novas arestas apresentado na heurística inicial. Applegate, Cook e Rohe (2003) utilizaram uma metodologia de reinicialização da busca a partir de uma solução local modificada de modo a encontrar novas soluções reduzidas, o qual foi denominado *Chained*

Lin-Kernighan (CLK) e que obteve uma melhora performática considerável, principalmente em grandes instâncias do PCV.

Ainda na década de 90, o principal *software* para resolução do PCV, denominado Concorde, foi desenvolvido por uma equipe de pesquisadores e estudiosos liderados por William Cook e disponibilizado gratuitamente para plataformas operacionais como Windows e Linux. Esta ferramenta possui um método de resolução determinístico que converge para a solução ótima global a partir de métodos e modelagens desenvolvidas especificamente para o PCV. Além disso, o programa também possui diversos métodos heurísticos para resolução do PCV, como a própria implementação da heurística CLK proposta por Applegate, Cook e Rohe (2003). A Figura 3 demonstra a interface gráfica do *software* em uma simulação realizada com uma instância randômica de 40 pontos e a respectiva rota otimizada encontrada.

Figura 3 – Interface gráfica do *software* Concorde.



Fonte: O autor.

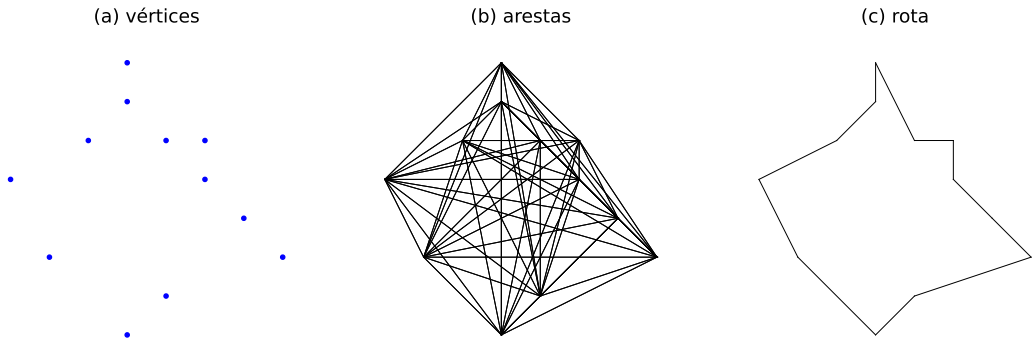
A partir dos anos 2000 até os dias atuais, com o advento de métodos baseados em inteligência artificial, novos algoritmos têm sido estudados para aplicação e resolução do PCV, como o Algoritmo Genético, Colônia de Formigas e Redes Neurais Artificiais. Por se tratarem de propostas mais modernas, com relevante impacto em estudos recentes, tais algoritmos foram analisados em uma revisão bibliográfica específica a ser apresentada na Seção 2.3.

2.2 Formulação matemática

Uma das principais maneiras de representação do PCV é feita a partir de um grafo, no qual os vértices definem as cidades a serem visitadas pelo caixeiro e as arestas definem o caminho conectando duas cidades, as quais possuem um custo de conexão, classicamente definido pela distância euclideana entre as mesmas. O objetivo do problema é encontrar

um conjunto de arestas conectando todos os vértices em um ciclo único, chamado de rota, de menor custo possível. A Figura 4 demonstra esses elementos de maneira gráfica de modo a facilitar a interpretação matemática de tais componentes e do próprio objetivo do problema.

Figura 4 – Grafos com os principais elementos do PCV.



Fonte: O autor.

Dada uma instância do PCV com n vértices, uma matriz quadrada de ordem n pode ser definida contendo o custo de conexão $c_{i,j}$, com $i, j \leq n$ e $i \neq j$, entre cada par de vértices (i, j). Para o caso em que $c_{i,j} = c_{j,i}$, o problema é denominado simétrico, uma vez que o sentido de conexão das arestas que formam a rota não afeta o custo final da mesma.

Uma segunda matriz quadrada de mesma ordem n, composta pelas variáveis de decisão do problema, pode ser construída com base na seleção das arestas $x_{i,j}$ a formarem uma determinada rota, conforme Equação 2.1. Tal matriz representa o processo de transformação do conjunto total de arestas em uma rota válida, ilustrado entre as imagens (b) e (c) da Figura 4.

$$x_{i,j} = \begin{cases} 1 & \text{a aresta entre os vértices } i \text{ e } j \text{ é selecionada para formar a rota} \\ 0 & \text{caso contrário} \end{cases} \tag{2.1}$$

A partir das definições do custo de cada aresta e os valores de seleção booleanos das mesmas, a função objetivo a ser minimizada para o PCV é definida a partir da expressão de programação linear inteira conforme Equação 2.2.

$$f_{obj} = min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} x_{i,j} \tag{2.2}$$

Uma vez definida a função objetivo do PCV, restrições devem ser adicionadas para limitar a seleção de arestas a fim de que a mesmas convertam em uma rota válida. O

primeiro grupo de restrições a ser considerado parte da seguinte observação: exatamente duas arestas devem conectar cada vértice para formar uma rota factível. Desta forma, as chamadas arestas de entrada e saída de cada vértice são definidas a partir das restrições destacadas na Equação 2.3 e Equação 2.4.

$$\sum_{i=1}^{n} x_{i,j} = 1 \quad j = 1, ..., n \tag{2.3}$$

$$\sum_{j=1}^{n} x_{i,j} = 1 \quad i = 1, ..., n \tag{2.4}$$

Um segundo grupo de restrições é definido com o objetivo de eliminar ciclos ou sub-rotas que possam ser formadas durante a seleção das arestas que atendem apenas às duas restrições definidas anteriormente. Considerando um subconjunto S de vértices com $n_1 < n$ que formam uma sub-rota, a somatória $x_{i,j}$ desse subconjunto é igual a n_1. A partir dessa observação, a restrição da Equação 2.5 adicionada à modelagem, aplicada para todo subconjunto S, garante que não existam ciclos intermediários na rota construída.

$$\sum_{S} x_{i,j} \leq n_1 - 1 \tag{2.5}$$

A modelagem matemática simplificada apresentada nesta seção foi proposta e desenvolvida ao longo de diversos anos e trabalhos, a partir das publicações de Robinson (1949) e Dantzig, Fulkerson e Johnson (1954). Atualmente, a melhor implementação computacional que resolve este modelo se encontra no *software* Concorde, o qual combina técnicas de resolução e restrições adicionais desenvolvidas por diversos pesquisadores da área ao longo de décadas de estudo do PCV.

2.3 Revisão bibliográfica

O objetivo desta seção é apresentar o estudo realizado sobre o Problema do Caixeiro Viajante destacando as principais técnicas de resolução utilizadas nas referências bibliográficas consultadas e pontuar características relevantes de cada abordagem de solução, as quais direcionaram o desenvolvimento computacional deste trabalho. A seleção das publicações foi feita com base na aplicação e ano de publicação, de modo a obter uma quantidade equilibrada entre estudos acadêmicos e industriais que representassem tendências modernas de resolução do PCV. Para facilitar o detalhamento e comparação de cada estudo, as publicações foram classificadas a partir dos seguintes critérios: (a) o algoritmo utilizado para resolução do problema, (b) o tipo do problema e (c) a aplicação do problema. A Tabela 1 demonstra os critérios de classificação utilizados e a Tabela 2 exibe a classificação de cada publicação consultada.

Tabela 1 – Critérios de classificação bibliográfica.

critério	**tipo**	**definição**
algoritmo	*K-Opt*	Alteração cíclica de k arestas de uma rota inicial
	IP	Programação Inteira
	RL	Aprendizado por Reforço e Redes Neurais Artificiais
	GA	Algoritmo Genético combinado com *K-Opt*
problema	*Padrão*	O problema com definições comuns e padronizadas
	Variação	O problema com novas definições customizadas
aplicação	*Acadêmica*	Utilização de instâncias acadêmicas do PCV
	Industrial	Utilização de instâncias industriais do PCV

Fonte: O autor.

Li et al. (2009) descrevem o desenvolvimento de uma aplicação computacional para sequenciamento de pontos de furação utilizando a linguagem de programação Delphi. Os autores exibem grafos para justificar as dificuldades encontradas no sequenciamento dos furos e apresentam o programa computacional desenvolvido que, além de realizar o processo de otimização, também gera um novo programa CNC a partir do sequenciamento computado. Os autores utilizaram como método de resolução do PCV o Algoritmo Genético.

Lima Junior (2010) utilizou 3 meta-heurísticas (GRASP, GA e RL) para resolução de instâncias acadêmicas do Problema do Caixeiro Viajante, sendo a principal delas o Aprendizado por Reforço. Através da união desses três métodos, o autor propôs um novo método cooperativo para resolução do PCV, utilizando as principais características e vantagens dos métodos escolhidos.

Li et al. (2014) utilizaram o Algoritmo Genético para resolver um tipo de PCV denominado Problema do Caixeiro Viajante Colorido (CTSP), no qual múltiplos "caixeiros" devem viajar simultaneamente por grupos de cidades distintas, conforme demonstrado em cada região colorida da Figura 5. A condição deste problema surge a partir do estudo de um equipamento industrial de usinagem, conforme Figura 6, o qual possui 2 cabeçotes (A) e (B) que trabalham em paralelo e devem estrategicamente atuar em regiões distintas e comuns realizando a operação de manufatura no menor tempo possível.

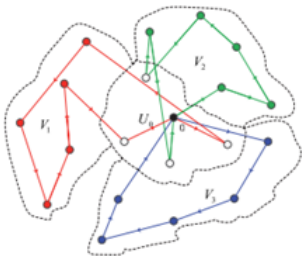
A publicação de Alatartsev (2015) é uma revisão de diversas publicações relacionadas ao sequenciamento de vértices do PCV e planejamento de movimentação aplicados no campo da robótica. O autor classifica diversos tipos de PCV e características do problema robótico, como o desvio de obstáculos para evitar colisões indesejadas, a fim de determinar técnicas específicas para resolução de cada um deles. A maior parte dos

Tabela 2 – Resumo da classificação bibliográfica.

referência	algoritmo				problema		aplicação	
	K-Opt	*IP*	*RL*	*GA*	*Pad.*	*Var.*	*Acad.*	*Ind.*
Li et al. (2009)	-	-	-	x	x	-	-	x
Lima Junior (2010)	-	-	x	x	x	-	x	-
Li et al. (2014)	-	-	-	x	-	x	-	x
Alatartsev (2015)	x	-	-	x	x	-	-	x
Imeson e Smith (2017)	-	x	-	-	-	x	x	-
Suarez-Ruiz (2018)	x	x	-	-	x	-	-	x
Malazgirt (2019)	-	-	x	-	x	-	x	-
Xing, Tu e Xu (2020)	-	-	x	-	x	-	x	-
Nedjatia e Vizvari (2020)	-	x	-	-	-	x	-	x
Costa et al. (2020)	x	-	x	-	x	-	x	-
Zheng et al. (2020)	x	-	x	-	x	-	x	-
Lu, Hao e Wu (2020)	x	-	-	x	-	x	x	-
Nammouchi (2020)	-	-	x	-	x	-	x	-
Bresson e Laurent (2021)	-	-	x	-	x	-	x	-

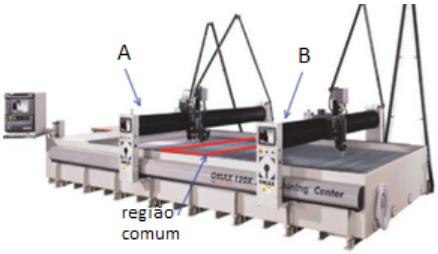
Fonte: O autor.

Figura 5 – Esquema do PCV colorido.



Fonte: (LI et al., 2014).

Figura 6 – Equipamento de estudo do PCV colorido.



Fonte: Adaptado de Li et al. (2014).

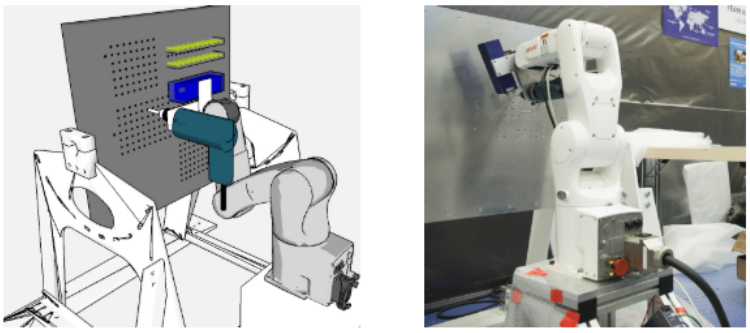
problemas abordados não envolve relação complexa entre os vértices de trabalho, condição que o autor propõe para trabalhos futuros e que se encaixa no problema de rebitagem estudado neste trabalho. Diversas técnicas de resolução do problema são mencionadas, como Algoritmo Genético, Recozimento Simulado e K-Opt.

Imeson e Smith (2017) desenvolveram uma técnica própria para resolução do PCV denominada *T-Clustering*, a qual realiza o agrupamento de vértices utilizando um fator numérico T com o objetivo de gerar e resolver subproblemas simplificados

definidos por esses grupos menores de vértices. O método básico para resolução do PCV é a Programação Inteira, a qual foi testada em instâncias acadêmicas do problema. Tal técnica de agrupamento é vantajosa em problemas nos quais os grupos de vértices não são bem definidos, porém no problema de rebitagem deste trabalho o agrupamento pode ser facilmente definido a partir de cada peça rebitada.

O estudo de Suarez-Ruiz (2018) faz parte de um desafio proposto por um fabricante aeronáutico no qual um conjunto aleatório de 245 pontos de furação deveriam ser executados da maneira mais rápida utilizando um robô industrial. O método de resolução deve ser capaz de definir o sequenciamento ótimo dos furos, evitar colisões do equipamento durante movimentação e definir a configuração ideal das juntas robóticas. Para solucionar o desafio, o autor desenvolveu um aplicativo denominado RoboTSP, o qual combina técnicas determinísticas e heurísticas para executar as simulações de otimização. A Figura 7 ilustra o ambiente virtual e o respectivo ambiente real nos quais os testes de furação robotizada foram executados.

Figura 7 – (a) Ambiente virtual e (b) ambiente real para testes de furação robotizada.



(a) (b)

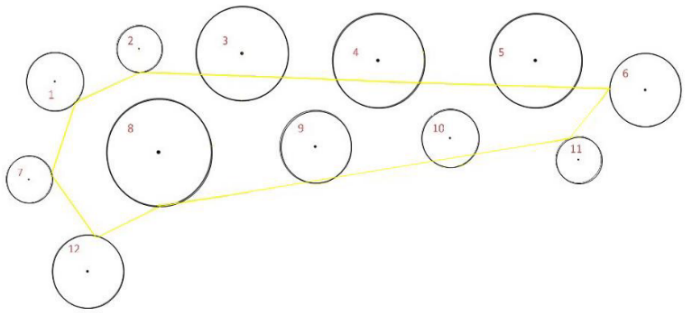
Fonte: (SUAREZ-RUIZ, 2018).

Malazgirt (2019) estudou a utilização da técnica de Aprendizado por Reforço combinada com Redes Neurais Artificiais, denominada DRL, para resolução de instâncias acadêmicas do PCV. A biblioteca de inteligência artificial utilizada na implementação computacional do sistema foi a Tensorflow, uma das principais ferramentas disponibilizadas pela Google para estudos e desenvolvimentos de IA.

A publicação de Xing, Tu e Xu (2020) apresenta um sistema de aprendizado por reforço combinado com a Árvore de Decisões de Monte Carlo (MCTS) para desenvolver a política de seleção e sequenciamento dos vértices do PCV. Os autores destacam que tal técnica apresenta eficiência computacional significativa, podendo ser aplicada em instâncias de grande dimensão do Problema do Caixeiro Viajante. As instâncias utilizadas para realização das simulações são instâncias acadêmicas.

Nedjatia e Vizvari (2020) realizaram a modelagem do PCV utilizando Programação Inteira e *solvers* de resolução dos modelos como o CPLEX e Knitro. O problema se baseia em uma célula industrial com 75 pontos de soldagem a serem executados por um robô automatizado. O tipo do PCV é definido como Problema do Caixeiro Viajante com Vizinhança Circular (TSPCN), no qual cada vértice é o centro de uma região circular que deve ser visitada com a menor rota possível. Um exemplo de resolução de uma instância com 12 vértices é exibido na Figura 8.

Figura 8 – Exemplo do TSPCN.



Fonte: (NEDJATIA; VIZVARI, 2020).

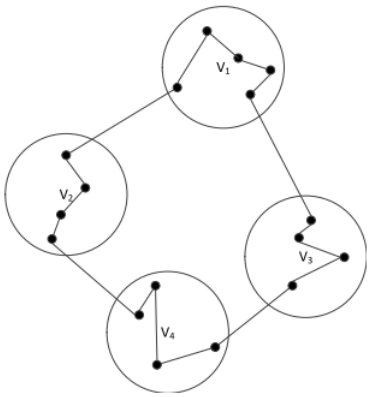
Costa et al. (2020) desenvolveram um sistema que combina o Aprendizado por Reforço com a heurística 2-Opt com objetivo de calcular estratégias de substituição de arestas utilizando um método de inteligência artificial. A aplicação do sistema foi realizada em instâncias acadêmicas do PCV. Além das técnicas principais, o sistema completo ainda inclui Redes Neurais Artificiais e subsistemas chamados de *encoders* e *decoders*, todos conectados entre si e responsáveis pela atividade de substituição das arestas da rota.

Zheng et al. (2020) utilizaram o Aprendizado por Reforço, mais especificamente os métodos específicos SARSA, Monte Carlo e *Q-Learning*, combinado à heurística LKH-3, para propor uma ferramenta de solução do PCV chamada VSR-LKH. Tal implementação alcançou melhora performática em relação à heurística LKH-3 a partir de diversas validações realizadas em instâncias acadêmicas do PCV.

Lu, Hao e Wu (2020) realizaram um estudo da variação agrupada do PCV utilizando métodos heurísticos, como a heurística LKH e o Algoritmo Genético. A modelagem do problema é feita de modo que as arestas entre vértices de diferentes grupos assumem valores que forçam a solução de cada grupo separadamente, conforme exemplo da Figura 9. Uma técnica similar de agrupamento de vértices foi utilizada na implementação do sistema neste trabalho para aplicação no problema industrial de rebitagem.

Nammouchi (2020) estudou uma técnica inovadora de inteligência artificial denominada GLN, na qual uma rede neural artificial é treinada com soluções de instâncias do PCV com o objetivo de identificar padrões e então ser capaz de resolver novas instâncias

Figura 9 – Exemplo do PCV agrupado.



Fonte: (LU; HAO; WU, 2020).

nunca observadas pelo sistema. O autor informa que foram utilizadas aproximadamente 50000 soluções de variadas instâncias para execução do treinamento da rede neural e a aplicação da inferência foi realizada em instâncias acadêmicas do PCV.

Bresson e Laurent (2021) utilizaram métodos NLP como base para implementação computacional do sistema proposto, no qual a linguagem original seriam os vértices e a sua respectiva tradução é a rota resultante do sequenciamento dos mesmos. O processo de tradução utiliza *encoders* e *decoders* computacionais e o treinamento é realizado com uso de técnicas de Aprendizado por Reforço. A aplicação do sistema desenvolvido foi realizada em instâncias acadêmicas do PCV.

A partir da bibliografia considerada na Tabela 2 e descrita nesta seção, é possível observar uma tendência nos últimos anos no uso de algoritmos de inteligência artificial para resolução do PCV, com destaque para o Aprendizado por Reforço. Tais algoritmos são baseados na estratégia de busca e construção de rotas do algoritmo K-Opt, com inclusão de regras e definições adicionais dos próprios sistemas de inteligência artificial. Além disso, são notórias, em praticamente todas as publicações, as menções à heurística de Lin-Kernighan e o reforço de sua importância e influência para o desenvolvimento de novos métodos heurísticos de resolução do PCV.

Em contrapartida à sua popularidade, não foi encontrada nenhuma implementação computacional da heurística de Lin-Kernighan conforme publicação de 1973. Ao investigar tal publicação o motivo é claro: apesar de se tratar de um método heurístico, que de modo geral tende a ser simplificado e de rápida implementação, a heurística de Lin-Kernighan é um método complexo que, por ter sido descrito em uma década com poucos recursos computacionais e linguagens de programação, possui poucos detalhamentos para suportar sua implementação.

Dessa última observação se origina um dos objetivos deste trabalho, mais especificamente, realizar a implementação computacional da heurística de Lin-Kernighan. Além de tal implementação servir como meio de resolução para o problema de sequenciamento de rebites aeronáuticos, a mesma também pode ser utilizada como base de consulta para novos pesquisadores interessados em tal método heurístico a fim de facilitar a compreensão do mesmo e inspirar novas implementações. A implementação computacional do método é detalhada no Capítulo 4 e Capítulo 5. O próximo capítulo apresenta os principais conceitos do processo de rebitagem, além de um breve histórico de aplicações automatizadas e a definição das funções de custo industrial aplicadas neste trabalho.

3 O PROCESSO DE REBITAGEM

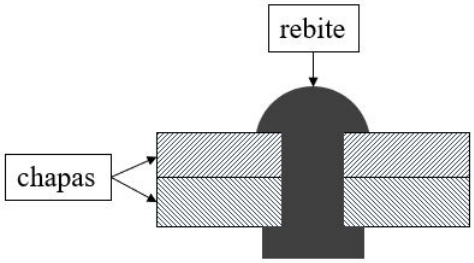
Neste capítulo serão apresentadas as bases conceituais do processo de rebitagem, destacando a evolução deste processo por meio de mecanismos automatizados e o paradigma do sequenciamento de rebites como um modelo customizado do PCV.

3.1 Aplicações aeronáuticas

O processo de rebitagem é definido pela união de dois ou mais componentes através do uso de rebites, elementos metálicos de fixação definitiva, que são instalados por meio da conformação de uma de suas extremidades. O uso de rebites durante a história pode ser observado no Egito Antigo, utilizados para fixação de objetos e ferramentas de arte, e nos séculos 7 e 8, utilizados para unir pranchas de madeira de embarcações. Durante o século 19, a rebitagem se tornou o principal processo de união de materiais nos mais diversos campos de aplicação (COLLETTE, 2011).

De acordo com Tong e Qu (2009), no meio da manufatura aeronáutica, a rebitagem é o principal processo de união de chapas metálicas, seguida do parafusamento e da colagem adesiva. Segundo Zhao et al. (2020), quando comparada aos outros dois processos citados, a rebitagem se destaca em termos de estabilidade, vida útil, confiabilidade, além de ser um processo simples e rápido que garante alta força de conexão entre os elementos unidos. Por esses motivos, ao longo de toda a estrutura de uma aeronave pode-se observar rebites, como nas asas, no revestimento da fuselagem e nos reforçadores internos. A Figura 10 ilustra um esquema simplificado demonstrando um rebite instalado em duas chapas.

Figura 10 – Esquema da instalação de rebite.



Fonte: O autor.

Diversos métodos de instalação de rebites foram desenvolvidos ao longo dos anos porém, na indústria aeronáutica, métodos de conformação mecânica são os mais utilizados. O principal método de instalação manual é realizado com o uso de marteletes e barras

encontradoras, no qual são necessários dois operadores para execução da operação. Conforme exemplificado na Figura 11, o operador à esquerda utiliza o martelete para realizar a conformação do rebite através da aplicação de forças de martelamento em uma de suas extremidades, ao mesmo tempo que o operador à direita utiliza a barra encontradora para sustentar a força aplicada pelo martelete aplicando assim a força contrária na extremidade do rebite a ser conformada. Em áreas de livre acesso, geralmente localizadas nas redondezas de componentes menores, é comum o uso do galifão, ferramenta que possui duas garras metálicas, utilizadas para a aplicação e sustentação da força de conformação, a qual, neste processo é aplicada de maneira contínua.

Figura 11 – Processo de instalação manual de rebites.

Fonte: (GARTSTEIN; PUTNAM; KLIEWER, 2016).

Para atender aos mais variados requisitos estruturais e de performance aerodinâmica, projetistas contam com uma grande variedade de tipos de rebites. A matéria-prima utilizada na fabricação desses elementos costuma ser um material equivalente ao utilizado em componentes estruturais a serem rebitados, evitando assim reações eletroquímicas que possam gerar efeitos de desgaste e corrosão nas juntas. Além da matéria-prima principal, é comum a aplicação de camadas de proteção superficial e tratamentos térmicos nos rebites, com o intuito de potencializar a sua proteção e isolamento químico. As principais ligas metálicas utilizadas na fabricação de rebites são compostas por alumínio e titânio.

Do ponto de vista geométrico, rebites exercem funções estruturais e aerodinâmicas na aeronave. Apesar do seu corpo comumente cilíndrico, a cabeça do rebite possui diversas formas, as quais implicam na complexidade do seu processo de instalação. Rebites chamados de boleados possuem a cabeça protuberante, geralmente em formato semi-esférico (conforme Figura 10). Como esta geometria da cabeça resulta em características aerodinâmicas desvantajosas, esses rebites são preferivelmente utilizados em junções internas ou carenadas. Rebites denominados escareados possuem cabeça fabricada escareada de tal forma que ao serem instalados apresentarão a superfície do topo da cabeça paralela à superfície das peças rebitadas. Apesar de possuir características estrutural e aerodinâmica vantajosas, a instalação de rebites escareados requer um processo de furação mais complexo, uma vez

que os furos devem apresentar diâmetro variável para assentamento da cabeça. A Figura 12 demonstra alguns tipos de rebites característicos utilizados em juntas aeronáuticas. O primeiro rebite, da esquerda para a direita, é um rebite boleado e o segundo é um rebite escareado.

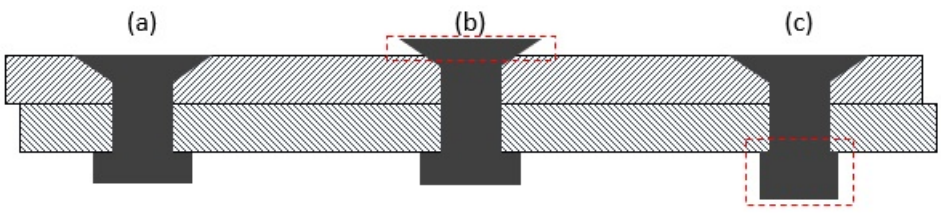
Figura 12 – Exemplo de rebites aeronáuticos.

Fonte: Adaptado de Australian Aerospace Engineering (2017).

Para facilitar a identificação, fabricação e o consumo de rebites, normas e padrões aeronáuticos foram desenvolvidos, como os padrões internacionais AN, NAS e MS. Nesses documentos, um código alfanumérico padronizado é definido para denominar as principais características de fabricação do rebite, como a geometria da cabeça, material, diâmetro, comprimento, entre outras características de fabricação.

O processo de instalação do rebite deve seguir requisitos de qualidade para garantia de sua correta instalação. Segundo Tong e Qu (2009), uma das características controladas é a geometria da cabeça conformada, mais especificamente sua altura e diâmetro. A partir do controle desses dois valores é possível concluir se a força de conformação aplicada no corpo do rebite é suficiente para garantir a correta união das peças. Outra característica controlada, específica para rebites escareados, é a altura da superfície da cabeça conformada em relação à superfície da peça unida. A correta instalação desse tipo de rebite deve garantir que a diferença entre essas superfícies seja a menor possível. A Figura 13 ilustra uma seção rebitada destacando possíveis defeitos de instalação de rebites.

Figura 13 – Condições de instalação de rebites: (a) instalação adequada, (b) altura da cabeça fabricada inadequada, (c) tamanho da cabeça conformada inadequada.



Fonte: O autor.

Apesar de ser um processo de união antigo, a rebitagem continua evoluindo e acompanhando as tendências mundiais da manufatura. O avanço tecnológico fez com que novos tipos de materiais, mais resistentes e leves, fossem desenvolvidos ao longo dos anos. Wu et al. (2021) destaca que a utilização desses novos materiais, com propriedades físicas e térmicas não convencionais, gera novos desafios e incentiva o estudo de novos processos de junção. Conforme Al-Lami, Hilmer e Sinapius (2018), o desenvolvimento de materiais em fibra de carbono (CFRP) representa um avanço em estruturas aeroespaciais ao mesmo tempo em que o processo de fabricação de peças complexas e o custo agregado a este material se tornam obstáculos a serem vencidos para adoção da fibra de carbono em larga escala. Avanços industriais no campo da automação também representam uma grande etapa da evolução do processo de rebitagem, a qual será discutida na próxima seção.

3.2 Soluções automatizadas

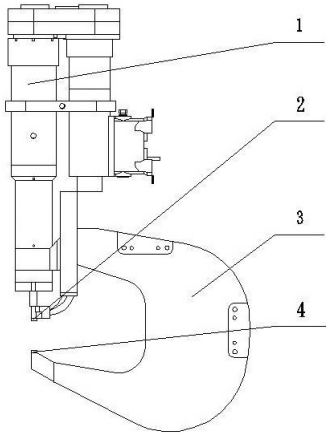
O conceito de automação e trabalho mecanizado e automatizado existe há muitas décadas porém o seu significado possui diferentes interpretações. Uma das definições exemplificadas por Janssen et al. (2019), retirada da enciclopédia britânica, é dada pela "aplicação de máquinas em tarefas antes realizadas por humanos ou em tarefas que seriam impossíveis de serem realizadas sem as mesmas". O termo mecanização difere do termo automação pois, enquanto o primeiro se refere à simples substituição do trabalho humano por máquinas, o segundo sugere a integração de máquinas em um sistema autônomo.

Segundo Hitomi (1994), a automação surgiu no ano de 1936 através de um estudo para transferência automatizada de peças em uma linha de produção veicular e evoluiu até a década de 90 com o surgimento de dois principais conceitos: *factory automation*, definido pelos processos de fabricação automatizados, e *office automation*, definido pela automação de atividades de negócios a partir do uso de computadores.

No caso do processo de rebitagem, por se tratar de uma atividade discreta, repetitiva e com riscos ergonômicos, diversas soluções automatizadas foram desenvolvidas com o objetivo de otimizar o ciclo de execução, reduzir o risco de acidentes e aumentar a qualidade final do produto. A patente ilustrada na Figura 14 demonstra um tipo de instalador automatizado de rebites, no qual o posicionamento do furo a ser rebitado é manual, enquanto a Figura 15 demonstra uma patente robotizada na qual o posicionamento é automatizado através do uso de um robô industrial manipulador.

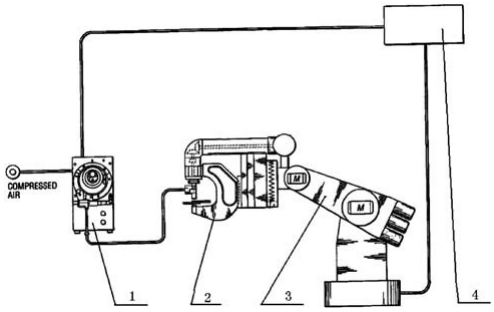
Meißner et al. (2018) desenvolveu um tipo de solução semi-automatizada de rebitagem com o uso de um robô colaborativo. Buscando melhorar a ergonomia dos montadores, este estudo utilizou um pequeno robô colaborativo localizado na estrutura interna a ser rebitada, substituindo assim o operador no processo de conformação. O teste executado ao longo de toda estrutura aeronáutica pode ser visto na Figura 16.

Figura 14 – Patente de rebitagem automatizada: (1) unidade de puncionamento, (2) punção, (3) quadro em formato "C", (4) molde.



Fonte: (TONG; QU, 2009).

Figura 15 – Patente robotizada de rebitagem: (1) alimentador de rebites, (2) mecanismo de instalação do rebite, (3) robô, (4) controlador.



Fonte: (TONG; QU, 2009).

É importante ressaltar que a origem da rebitagem automatizada está ligada ao processo de furação, uma vez que o mesmo forma o furo nas peças a serem unidas pelo processo de rebitagem. Diversos sistemas de furação automatizados são conhecidos e utilizados na indústria, porém a furação aeronáutica apresenta características, como tolerâncias do diâmetro do furo, que demandam um desenvolvimento específico para criação de sistemas automatizados. A partir dessa automação inicial, máquinas de rebitagem automatizadas foram propostas, estudadas e desenvolvidas. Um exemplo de máquina de furação automatizada é visto na Figura 17, a qual foi desenvolvida para executar furos nos reforçadores longitudinais do Boeing 747 na década de 90. Segundo Calawa e Culpepper (1994), o uso deste equipamento resultou na melhor qualidade e flexibilidade da furação, ao mesmo tempo em que reduziu o custo de fabricação, uma vez que a taxa de peças refugadas por não atender a tolerância de posicionamento dos furos diminuiu consideravelmente.

Figura 16 – Robô colaborativo para rebitagem.

Fonte: (MEIßNER et al., 2018).

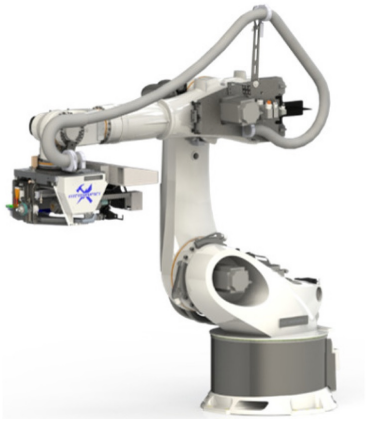
Figura 17 – Máquina de furação automatizada de reforçadores longitudinais.

Fonte: (CALAWA; CULPEPPER, 1994).

Além do uso de máquinas convencionais, outro tipo de equipamento amplamente estudado para aplicação de furação e rebitagem automatizada são os robôs industriais, conforme demonstrado anteriormente na patente da Figura 16. A aplicação deste tipo de tecnologia foi possível através da evolução da precisão de posicionamento desse tipo de equipamento e a redução do investimento necessário para sua aquisição. Segundo International Federation of Robotics (2021), o número de robôs trabalhando em fábricas alcançou a marca recorde de 2,7 milhões de unidades em 2019. O estudo de Holt e Clauss (2015) demonstra um sistema robotizado responsável pela furação de superfícies aeronáuticas de grande curvatura, com destaque aos subsistemas desenvolvidos para garantia do correto ângulo de furação. A Figura 18 ilustra o braço robótico e o cabeçote de furação desenvolvido para o sistema.

A partir do desenvolvimento de equipamentos automatizados de furação aeronáutica, desenvolvedores se depararam com o desafio de evoluir tal solução em uma capaz de automatizar o ciclo completo de instalação de rebites. Para que tal solução fosse desenvolvida seria necessário automatizar os seguintes passos principais do processo de rebitagem:

Figura 18 – Exemplo de robô de furação.



Fonte: (HOLT; CLAUSS, 2015).

- Posicionamento: Etapa que consiste na movimentação da máquina ou produto até o ponto de rebitagem. Nesta etapa, sensores podem ser utilizados para corrigir imprecisões existentes na máquina ou produto;
- Furação: Etapa na qual o equipamento deve executar o ciclo de furação de modo a obter o furo desejado nas peças a serem unidas. Determinados equipamentos ainda devem aplicar uma força perpendicular à superfície furada para garantir que não existam folgas entre as peças;
- Selagem: Etapa realizada através da aplicação do selante aeronáutico, um tipo de produto viscoso com função vedante que, em determinadas situações, deve ser aplicado no furo a ser rebitado;
- Alimentação: Etapa que consiste na transferência do rebite de um local de repouso até o mecanismo de inserção automatizado;
- Instalação: Etapa definida pelo processo de inserção do rebite no furo e sua conformação mecânica.

Somada às etapas principais, existe também a etapa de medição, na qual as tolerâncias dos furos e dos rebites instalados são medidas nas etapas principais de furação e rebitagem, respectivamente. A mesma não é considerada como etapa principal pois, de forma geral, o *setup* realizado no equipamento e a realização de testes em corpos de prova (CDP) tende a reduzir a necessidade da medição durante execução do trabalho no produto aeronáutico.

Por se tratar de uma tecnologia específica do ramo aeroespacial, existem poucos desenvolvedores de equipamentos automatizados para rebitagem no mundo, quando se

comparado a máquinas de usinagem convencionais. Como destaque, são citados alguns fabricantes como Broetje, Gemcor, Electroimpact e Kuka. Neste trabalho serão considerados os modelos de máquinas rebitadoras IPAC (do fornecedor Broetje) e G86 (do fornecedor Gemcor), ilustradas respectivamente na Figura 19 e Figura 20, uma vez que são as máquinas de maior relevância e uso na empresa patrocinadora desta pesquisa, além de apresentarem um processo equivalente de sequenciamento de rebites entre si.

Figura 19 – Rebitadora Broetje IPAC.

Fonte: Página do fabricante Broetje.[1]

Figura 20 – Rebitadora Gemcor G86.

Fonte: Página do fabricante Gemcor.[2]

A partir da definição das máquinas que fazem parte do escopo deste trabalho, na próxima seção será detalhado o processo de sequenciamento de rebites para esses equipamentos, de modo a evidenciar o problema a ser resolvido, os desafios a serem enfrentados e os resultados que se espera alcançar.

3.3 Atividade de sequenciamento de rebites

Uma das principais tecnologias que surgiram ao longo do período de evolução da automação industrial, detalhado na Seção 3.2, foi o chamado Controle Numérico. Segundo Smid (2007), esse pode ser definido como "a operação de ferramentas de determinada máquina através do uso de instruções codificadas em seu sistema de controle". O conjunto dessas instruções necessárias para fabricação de um determinado produto ou execução de determinada operação é chamado de Programa CN ou Programa CNC.

O Controle Numérico trouxe grandes mudanças na indústria de usinagem metálica. O uso deste tipo de técnica em equipamentos automatizados resulta em melhorias como a velocidade, precisão, eficiência e repetibilidade do processo produtivo (KRAR, 1990). Uma das principais aplicações do Controle Numérico são os centros de usinagem, equipamentos automatizados responsáveis pela fabricação de peças de alta complexidade, conforme exemplificado na Figura 21.

[1] <https://www.broetje-automation.com/>. Acesso em 30 de julho de 2023
[2] <https://www.ascentaerospace.com/>. Acesso em 30 de julho de 2023

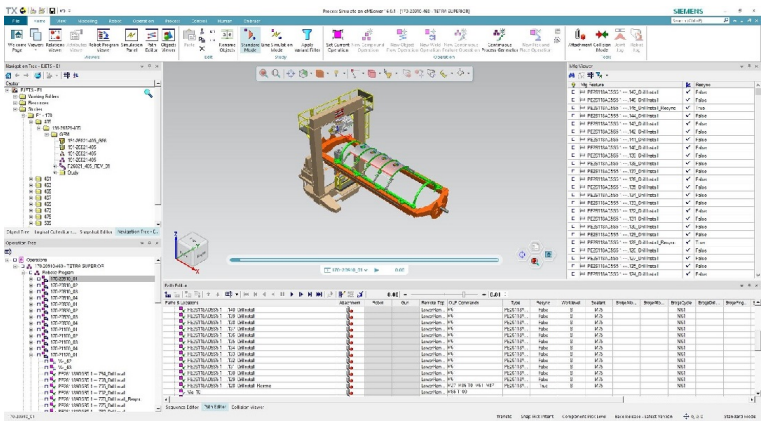
Figura 21 – Exemplo de centro de usinagem.

Fonte: (ROMI, 2022).

O desenvolvimento do programa CNC pode ser realizado a partir de dois métodos principais: o método *online*, no qual a programação é feita com uso do próprio equipamento e seus recursos nativos para gravação de coordenadas e comandos de máquina, e o método *offline*, o qual não necessita do equipamento e, com o avanço da computação, tem sido praticado com o auxílio de *softwares* computacionais de manufatura conhecidos como CAM.

O uso de ferramentas computacionais para programação CNC de máquinas de rebitagem automática é importante pois agiliza a tarefa do programador e garante a qualidade do programa CNC gerado através de recursos como validação lógica e simulação virtual, os quais evitam a execução de comandos incorretos e trajetórias indesejadas. Por se tratar de uma tecnologia muito específica, poucas soluções CAM existem para máquinas de rebitagem automatizada, das quais se destacam o Delmia, VDAF e o Tecnomatix, sendo este último o *software* utilizado pela empresa aeronáutica deste estudo e exemplificado na Figura 22.

Figura 22 – *Software* Tecnomatix.

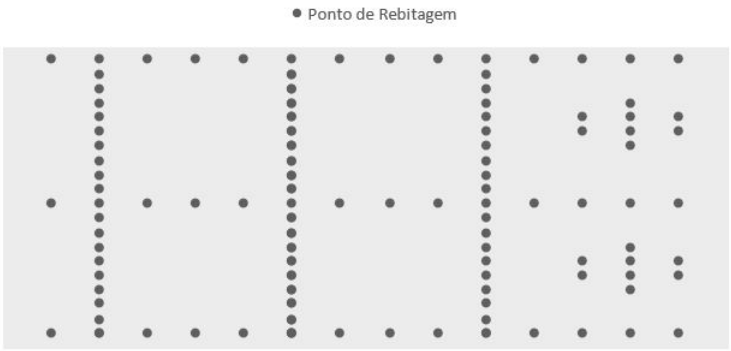


Fonte: O autor.

Apesar da atividade de programação CNC ter se tornado mais rápida com o auxílio de programas computacionais, isso não significa que a mesma seja uma tarefa trivial. Mesmo existindo diversas ferramentas que automatizam etapas do processo de programação, algumas dessas etapas devem ser executadas manualmente. Um exemplo de tarefa manual é o sequenciamento de rebites, na qual o programador deve ordenar a execução dos rebites que compõem o produto a ser rebitado.

Para exemplificar o processo de sequenciamento, a Figura 23 ilustra a vista superior de uma região rebitada. Tomando como base apenas a distância entre os pontos de rebitagem, encontrar o melhor sequenciamento dessa centena de pontos não é uma tarefa simples, enquanto que testar todas as combinações de sequências possíveis é uma tarefa computacionalmente inviável.

Figura 23 – Modelo de distribuição de rebites em um produto aeronáutico.

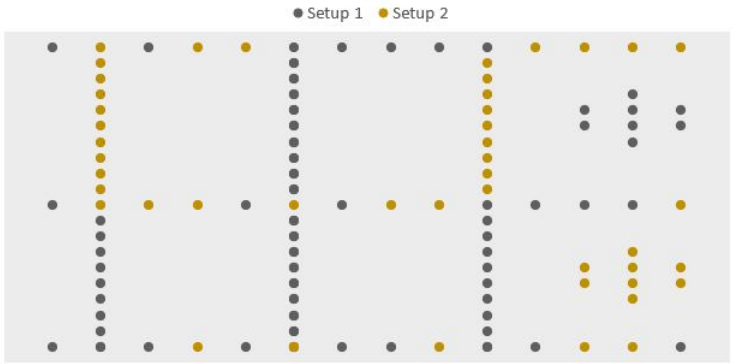


Fonte: O autor.

Uma maneira de simplificar a resolução desse problema é introduzir parâmetros do processo de rebitagem que auxiliem na divisão do mesmo em subproblemas menores. Um exemplo de divisão possível é definida pelo tipo de rebite instalado, uma vez que a alteração do tipo de rebite requer um tempo de *setup* do equipamento. Essa situação é exemplificada na Figura 24, a partir do agrupamento dos rebites em duas configurações de máquina necessárias. Para este problema, pode-se adotar uma estratégia de sequenciamento isolado de cada grupo, no caso de *setups* manuais e lentos, enquanto que uma estratégia de seleção mesclada pode ser adotada no caso de operações de *setup* rápidas e automatizadas.

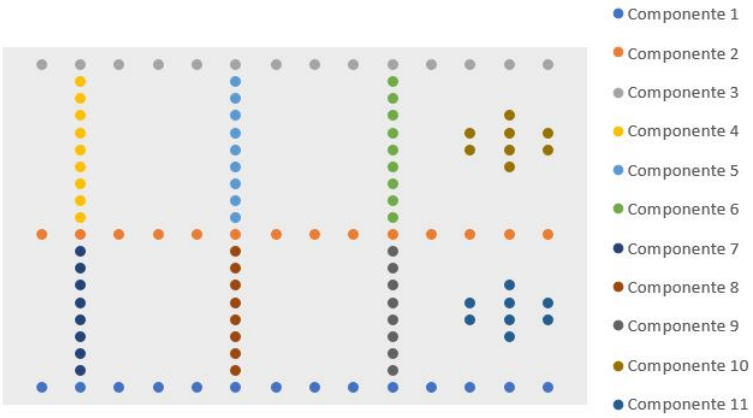
Outro tipo de agrupamento mais agressivo é demonstrado na Figura 25, na qual a divisão dos grupos de rebites é feita com base nos componentes aeronáuticos a serem rebitados. Com esta estratégia, o problema principal ilustrado foi divido em 11 subproblemas menores, os quais são resolvidos mais rapidamente e ainda podem ser combinados com outras estratégias de divisão para simplificar ainda mais a sua resolução. O ponto negativo desse agrupamento é que ele tende a resultar em soluções piores em relação às soluções do problema completo.

Figura 24 – Modelo de distribuição de rebites por *setup* em um produto aeronáutico.



Fonte: O autor.

Figura 25 – Modelo de distribuição de rebites por peça em um produto aeronáutico.



Fonte: O autor.

A partir das perspectivas descritas para o sequenciamento de rebites, um dos objetivos deste trabalho é o desenvolvimento de um método capaz de realizar este sequenciamento de maneira automatizada e que, ao mesmo tempo, garanta um ciclo otimizado do equipamento de rebitagem, a fim de reduzir o seu tempo operacional e o respectivo custo de fabricação do produto aeronáutico. Além do ganho fabril, a automação da tarefa de sequenciamento de rebites gera ganhos em horas de trabalho dos programadores CNC, uma vez que atualmente esta é a atividade mais onerosa no fluxo de programação CNC, a ser eliminada pelo processo automatizado desenvolvido neste trabalho. Na próxima seção será apresentada a definição das funções de custo industrial para o problema de sequenciamento de rebites, as quais deverão ser utilizadas em conjunto com o método heurístico a ser implementado neste trabalho.

3.4 Funções de custo industrial

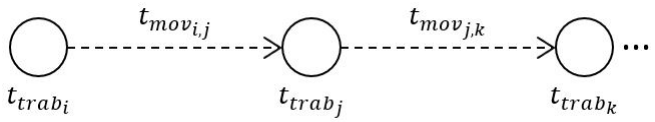
As funções de custo industrial para o problema de sequenciamento de rebites tem sua origem no grafo que define o PCV (em mais detalhes na Seção 2.2), no qual os vértices que definem as cidades a serem visitadas pelo caixeiro se transformam em pontos de rebitagem visitados pelo equipamento automatizado. A principal diferença entre o modelo industrial deste trabalho e o modelo canônico do PCV está na definição da função de custo de cada modelo: enquanto, de maneira geral, o modelo canônico tem como objetivo reduzir a distância percorrida entre os vértices visitados, o modelo industrial busca reduzir o tempo de visitação em cada ponto de rebitagem, chamado de custo temporal, a fim de minimizar o tempo total de operação do equipamento. A Equação 3.1 demonstra a função de custo de rebitagem para um rebite i baseada no tempo de execução de cada rebite (t_{reb}).

$$c_i = t_{reb_i} \tag{3.1}$$

O tempo de rebitagem pode ser divido em dois intervalos de tempo sequenciais: (a) o tempo de movimentação (t_{mov}) do equipamento entre dois rebites e (b) o tempo de trabalho (t_{trab}) no ponto de rebitagem, definido pelas tarefas necessárias para instalação do rebite. Assim, o tempo de rebitagem é reescrito a partir destes dois intervalos conforme Equação 3.2 e a Figura 26 exemplifica uma sequência de pontos de rebitagem e os respectivos tempos de movimentação e de trabalho em cada ponto.

$$t_{reb_{i,j}} = t_{mov_{i,j}} + t_{trab_i} \tag{3.2}$$

Figura 26 – Sequência de tempos de rebitagem.



Fonte: O autor.

O tempo exato de movimentação entre os rebites pode ser calculado a partir do perfil cinemático de cada um dos cinco eixos de movimentação do equipamento automatizado, porém, este método de cálculo requer que as coordenadas de cada eixo sejam pré-definidas, informação coletada durante a programação CN. Desta forma, para simplificar o cálculo do tempo de movimentação utiliza-se a velocidade média aproximada de movimentação ponto-a-ponto (v_{maq}) disponível no manual do equipamento robotizado e a distância euclideana entre pontos consecutivos de rebitagem i e j, a qual pode ser aproximada à distância

percorrida pelo equipamento ($\Delta s_{i,j}$), conforme Equação 3.3. Com tal valor coletado, o tempo de movimentação entre o i-ésimo rebite e o seu sucessor pode ser definido como a razão entre o deslocamento espacial do equipamento e sua respectiva velocidade média, conforme Equação 3.4.

$$\Delta s_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{3.3}$$

$$t_{mov_{i,j}} = \frac{\Delta s_{i,j}}{v_{maq}} \tag{3.4}$$

O tempo de trabalho em cada ponto de rebitagem também será aproximado nesta modelagem, uma vez que o detalhamento completo de tal tempo acarretaria em valores não-fixados de custo em cada ponto, tornando assim inviável a resolução do problema a partir do método proposto. Sendo assim, apenas as principais tarefas executadas pelo equipamento em cada ponto de rebitagem foram consideradas, sendo elas: (a) o tempo de *setup* do equipamento (t_{setup}) para configuração do rebite a ser instalado, (b) o tempo de alinhamento (t_{resync}) executado em determinados pontos de trabalho para correção da posição espacial e (c) o tempo do ciclo de rebitagem (t_{ciclo}), o qual consiste na operação mecânica de furação e instalação do rebite.

$$t_{trab_i} = t_{setup_i} + t_{resync_i} + t_{ciclo_i} \tag{3.5}$$

O tempo de *setup* é definido em três classes distintas conforme Equação 3.6, baseado na comparação entre os tipos de rebites a serem instalados, os quais podem demandar uma configuração específica de determinados subsistemas do equipamento automatizado. Quando o tipo do rebite não se altera, não existe *setup* agregado ($t_{setup} = 0$), já quando apenas o comprimento do rebite se altera, o *setup* é curto uma vez que os subsistemas do equipamento não se alteram totalmente, gerando apenas um pequeno intervalo de tempo de alimentação do novo rebite. Quando o tipo do rebite se altera, será necessário avaliar a configuração atual dos subsistemas de modo a concluir se existe a necessidade de um *setup* mais longo ou se o *setup* curto atende a configuração desejada.

$$t_{setup} = \begin{cases} 0 & \text{mesmo tipo de rebite} \\ t_{curto} & \text{mesmo tipo de rebite, porém o comprimento é diferente} \\ t_{longo} & \text{tipo diferente de rebite com alteração dos subsistemas} \end{cases} \tag{3.6}$$

O tempo de ciclo é definido por três etapas distintas conforme Equação 3.7, cada uma determinada pelo seu respectivo tempo: (a) o tempo de furação ($t_{furação}$), resultante do intervalo de furação da broca, (b) o tempo de selagem ($t_{selagem}$), etapa

em que determinados tipos de furos e rebites são selados antes da rebitagem e (c) o tempo de instalação ($t_{instalação}$), referente à inserção do rebite no furo e sua posterior conformação mecânica. Para simplificação desta modelagem, todos esses tempos serão tratados como constantes baseados no tipo de rebite a ser instalado. Variações existentes, como a espessura da junta rebitada, serão desconsideradas por se tratarem de valores pequenos que não afetam os cálculos do modelo adotado.

$$t_{ciclo_i} = t_{furação_i} + t_{selagem_i} + t_{instalação_i} \tag{3.7}$$

A partir das expansões definidas para o tempo de máquina, a função de custo industrial do problema pode ser reescrita em função dos tempos agregados ao processo de sequenciamento dos rebites conforme Equação 3.8. A Tabela 3 lista os valores adotados neste trabalho e possíveis para cada parâmetro com base no manual de operação da máquina rebitadora e em medições realizadas no próprio equipamento. O tempo de movimentação (t_{mov}) não é listado uma vez que este depende da distância entre os pontos de rebitagem.

$$c_{i,j} = t_{mov_{i,j}} + t_{setup_i} + t_{resync_i} + t_{furação_i} + t_{selagem_i} + t_{instalação_i} \tag{3.8}$$

Tabela 3 – Valores dos parâmetros do problema de rebitagem.

parâmetro	**valores**
v_{maq}	32mm/s
t_{setup}	0s $t_{curto} = 5s$ $t_{longo} = 20s$
t_{resync}	0s 5s
$t_{furação}$	1s
$t_{selagem}$	0s 1s
$t_{instalação}$	2s

Fonte: O autor.

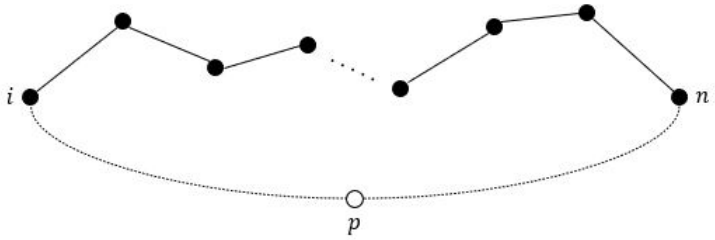
Para realizar a modelagem do problema em um grafo de modo a ser possível a resolução do mesmo pelo método proposto neste trabalho, algumas adequações devem ser realizadas no custo apresentado anteriormente. A primeira modificação parte da seguinte consideração: os custos ($c_{i,j}$) em um grafo estão agregados às arestas, as quais conectam

dois vértices (i, j) e, para o correto funcionamento do método de resolução, tais custos devem ser simétricos ($c_{i,j} = c_{j,i}$). Para que a função de custo industrial respeite esta propriedade, o custo entre dois rebites (i, j) é definido pela Equação 3.9. Utilizando tal definição combinada à Equação 3.2, pode-se verificar que a somatória do custo de todas as arestas de uma determinada rota terá valor igual ao dobro do tempo de máquina (t_{maq}).

$$c_{i,j} = 2 \times t_{mov_{i,j}} + t_{trab_i} + t_{trab_j} \tag{3.9}$$

Um outro ponto específico do problema de rebitagem está relacionado à geometria da solução obtida: para o PCV canônico a solução é uma rota que forma um caminho fechado, no qual todos os vértices conectam entre si, porém, para o problema de sequenciamento de rebites, a rota deve formar um caminho aberto de modo a se ter um vértice inicial e final que não se conectam entre si. Para obter tal característica, foi utilizado um vértice pivô p entre os vértices inicial i e final n com custos zerados ($t_{mov_p} = 0$ e $t_{trab_p} = 0$), conforme exemplificado na Figura 27, de modo que a rota entre i e n é artificialmente aberta.

Figura 27 – Esquema de sequenciamento com vértice pivô.



Fonte: O autor.

Além da principal função de custo industrial apresentada na Equação 3.8, com base no custo temporal entre cada ponto de rebitagem, outras duas funções de custo simplificadas foram utilizadas em simulações, a fim de avaliar determinadas características das instâncias industriais e o seu respectivo processo de resolução, conforme descrito a seguir:

- Custo euclideano: O custo euclideano para as instâncias industriais é definido apenas pelo deslocamento do equipamento automatizado (Δs). O cálculo deste custo é determinado pela distância euclideana tridimensional entre dois pontos, conforme Equação 3.10. A sua utilização tem como objetivo avaliar a dificuldade computacional do problema euclideano em três dimensões e comparar com os resultados euclideanos obtidos em instâncias acadêmicas e também com os resultados industriais obtidos utilizando as funções de custo agrupado e temporal;

$$c_{i,j} = \Delta s_{i,j} \tag{3.10}$$

- Custo agrupado: O custo agrupado, definido na função da Equação 3.11, é uma variante do custo euclideano com a inclusão de um fator de penalização $w_{peças} \geq 1$, conforme Equação 3.12, o qual amplia o valor do custo quando a peça rebitada entre dois rebites consecutivos é alterada. O objetivo no uso deste custo é avaliar a possibilidade da resolução de instâncias do PCV para cada peça separadamente, ao invés de uma instância única para o produto completo, e identificar se a mesma dificuldade computacional encontrada nos problemas acadêmicos agrupados será encontrada no problema industrial em condição agrupada.

$$c_{i,j} = \Delta s_{i,j} \times w_{peças_{i,j}} \tag{3.11}$$

$$w_{peças} = \begin{cases} 1 & \text{mesma peça rebitada} \\ > 1 & \text{peças diferentes rebitadas} \end{cases} \tag{3.12}$$

O processo de coleta das instâncias industriais e os parâmetros de rebitagem relevantes para definição dos custos das funções objetivo definidas neste capítulo será apresentado no Capítulo 6 e os resultados das simulações dessas instâncias são apresentados no Capítulo 7. No próximo capítulo serão apresentados os principais conceitos da heurística de Lin-Kernighan e determinados métodos variantes que influenciaram a implementação computacional neste trabalho.

4 HEURÍSTICA DE LIN-KERNIGHAN

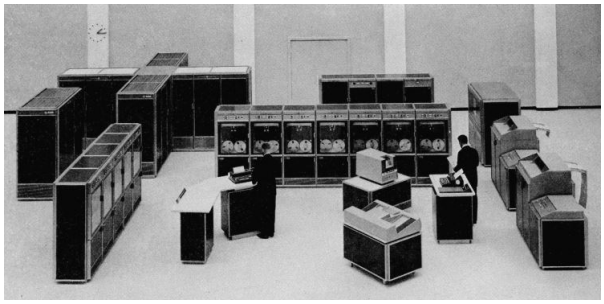
Neste capítulo serão apresentados os principais conceitos da heurística de Lin-Kernighan, detalhando as etapas necessárias para implementação computacional da mesma e variantes que serviram de base para a implementação computacional deste trabalho.

4.1 A publicação de Kernighan e Lin (1973)

A primeira fonte de informação que intuitivamente surge ao estudar a heurística de Lin-Kernighan é a sua primeira publicação no ano de 1973. A maior parte das fontes consultadas que tratam de métodos heurísticos para o Problema do Caixeiro Viajante irão citar esta publicação como um dos marcos mais relevantes nesse tipo de estudo, sendo a base e motivação para o desenvolvimento de derivações propostas a partir de sua publicação.

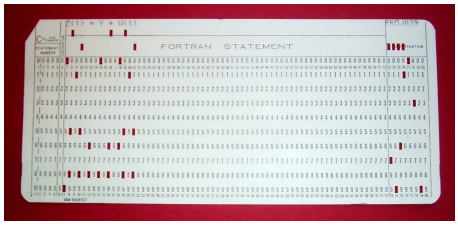
A linguagem de programação utilizada pelos autores foi o Fortran, uma das primeiras linguagens de programação da história, criada pela IBM. A execução foi realizada em um sistema de processamento GE635 utilizando cartões perfurados como meio de programação, conforme ilustrado na Figura 28 e Figura 29. Por se tratar de um período no qual os meios de computação ainda eram arcaicos, o artigo não faz nenhuma exibição de pseudocódigos da implementação computacional desenvolvida, apresentando apenas o conceito básico da heurística proposta, possivelmente com o objetivo de compartilhar a informação necessária para uma futura implementação em outra linguagem de programação.

Figura 28 – Sistema de processamento GE635.



Fonte: (General Eletric, 1964).

Figura 29 – Exemplo de cartão perfurado.



Fonte: Página sobre Fortran no Wikipedia.[1]

[1] <https://pt.wikipedia.org/wiki/Fortran>. Acesso em 30 de julho de 2023

Como preparação para o detalhamento da heurística de Lin-Kernighan, as suas notações básicas são definidas na Tabela 4 e serão utilizadas neste trabalho.

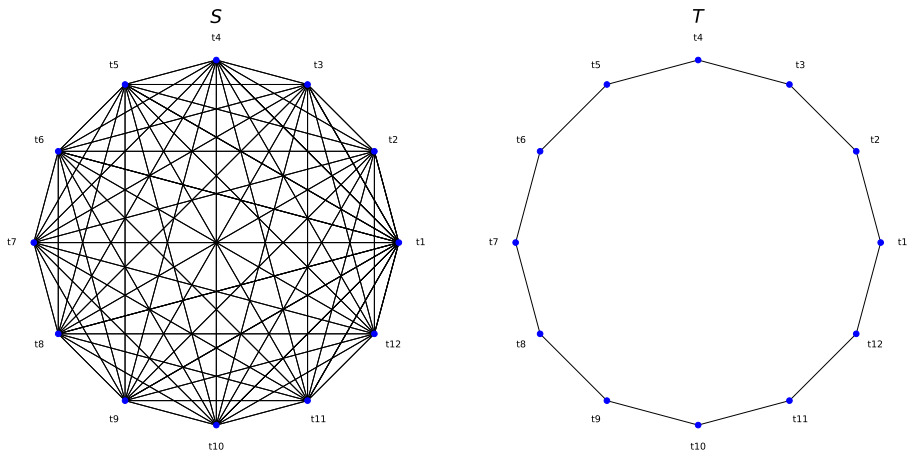
Tabela 4 – Notações para a heurística de Lin-Kernighan.

notação	**definição**
S	Conjunto de todas as arestas que compõem um problema com n vértices
T	Subconjunto de S composto por n arestas que formam uma rota
X	Conjunto $\{x_1, ..., x_k\}$ de k arestas a serem desconectadas de uma rota
Y	Conjunto $\{y_1, ..., y_k\}$ de k arestas a serem conectadas em uma rota
G	Ganho total ao remover de uma rota todas as k arestas do conjunto X e incluir todas as k arestas do conjunto Y $(G = \sum_1^k g_i)$
x_i	i-ésima aresta a ser desconectada de uma rota
y_i	i-ésima aresta a ser conectada em uma rota
t_i	i-ésimo vértice de uma rota
g_i	i-ésimo ganho dado pela diferença entre os módulos de uma aresta x_i desconectada da rota e uma aresta y_i conectada à rota $(g_i = \|x_i\| - \|y_i\|)$
$f(T)$	Função que calcula o custo de uma rota T

Fonte: O autor.

A Figura 30 exemplifica os conjuntos S e T para um problema definido com $n = 12$ vértices, no qual T é também a melhor solução global que minimiza $f(T)$. Para uma rota factível é possível constatar que: (a) cada vértice da rota deve possuir conexão com exatamente duas arestas e (b) existe um único ciclo que conecta todos os vértices entre si.

Figura 30 – Conjuntos S e T.

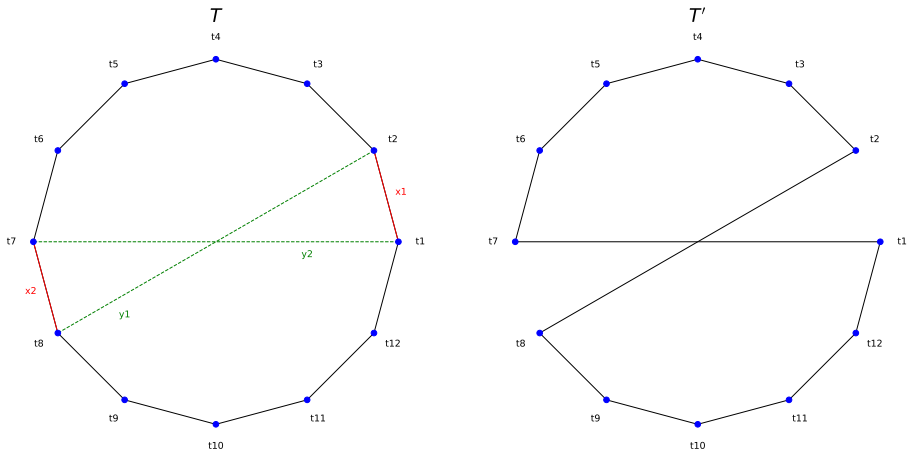


Fonte: O autor.

A estratégia proposta na heurística é realizada a partir da substituição sequencial de arestas: a k-ésima aresta x_k a ser removida da rota e a k-ésima aresta y_k a ser adicionada

compartilham um mesmo vértice t. Esta estratégia é adotada para facilitar a manutenção da factibilidade da rota, uma vez que a sua reconexão se torna mais simples quando comparada a um método de substituição aleatória de arestas. A Figura 31 exemplifica essa estratégia durante a substituição de duas arestas de uma determinada rota T, apesar do exemplo ilustrar uma transformação que aumenta o custo da rota. O vértice t_2 é compartilhado pelas arestas x_1 e y_1, em sequência o vértice t_8 é compartilhado pelas arestas y_1 e x_2, e finalmente o vértice t_7 é compartilhado pelas arestas x_2 e y_2, de maneira sequencial em que as arestas são removidas e adicionadas à rota: $x_1 \rightarrow y_1 \rightarrow x_2 \rightarrow y_2$.

Figura 31 – Transformação de uma rota T.



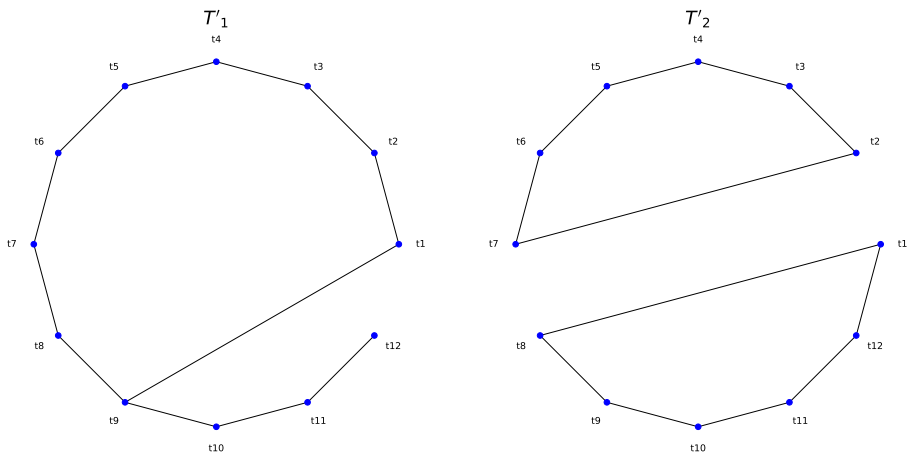
Fonte: O autor.

Para que uma rota T seja transformada em uma nova rota $T^{'}$ a partir dos conjuntos X e Y, cada conjunto deve possuir pelo menos duas arestas. A remoção de uma única aresta e adição de outra única aresta torna a rota $T^{'}$ infactível, uma vez que um dos vértices adjacentes à aresta adicionada possuirá conexão com três arestas, enquanto um dos vértices adjacentes à aresta removida possuirá conexão com apenas uma aresta. Essa condição afeta a propriedade cíclica da rota, na qual todos os vértices devem estar totalmente conectados entre si. Tal condição é demonstrada na rota $T_1^{'}$ da Figura 32.

Ao desconectar o conjunto de arestas $X = \{x_1, x_2\}$ da rota T, conforme observado na Figura 31, quatro vértices (t_1, t_2, t_7, t_8) ficam disponíveis para reconexão. Além das alternativas de reconectá-los originalmente conforme T, ou gerar uma nova rota conforme $T^{'}$, uma última alternativa seria realizar a ligação $t_1 \rightarrow t_8$ e $t_2 \rightarrow t_7$, conforme visto em $T_2^{'}$ na Figura 32. Essa alternativa gera duas sub-rotas isoladas, que tornam esta opção infactível, sendo possível constatar que: (a) existirá apenas uma nova alternativa de reconexão quando cada conjunto X e Y possuírem duas arestas.

A partir da definição das notações básicas e as exemplificações gráficas dos principais objetos e condições do problema, a adaptação do algoritmo descrito por Kernighan e Lin

Figura 32 – Rotas infactíveis.



Fonte: O autor.

(1973) é transcrita no Algoritmo 1. A mesma é apresentada em caráter não-computacional a fim de ressaltar as dificuldades interpretativas decorrente da inexistência de detalhamentos computacionais, os quais serão apresentados na Seção 5.1.

O passo 1 representa o início do algoritmo a partir de uma rota randômica, sendo indicado que a implementação em uma linguagem de programação possua um meio de embaralhamento de vértices e arestas de modo a formar novas rotas iniciais. Essa característica randômica da rota inicial é uma das propriedades que determinam que o método proposto pelos autores é heurístico.

O passo 2 inicializa a variável G, a qual recebe o valor acumulado do ganho obtido com cada modificação executada na rota T. Uma vez o ganho sendo positivo ($G_i > 0$) e crescente ($G_{i+1} > G_i$), a busca por novas arestas é permitida, sendo interrompida apenas quando o ganho decresce. Desta forma, a implementação de G deve ser feita a partir do uso de um vetor de valores de ganho, e não apenas uma variável numérica com o valor atual observado, de modo a facilitar os cálculos numéricos.

Os passos 3 e 4 definem a seleção das primeiras arestas x_1 e y_1 a ser removida e incluída na rota, respectivamente. A seleção deve ser feita sequencialmente, de modo que x_1 e y_1 compartilhem o mesmo vértice t_2. O custo de y_1 deve ser menor que x_1 para que o ganho g_1 definido pela troca das arestas seja positivo. A seleção arbitrária das arestas iniciais também denota o carácter heurístico do método.

O passo 5 representa o laço de repetição exploratório pelo qual novas arestas x_i e y_i ($i \geq 2$) são encontradas pelo algoritmo. Esse laço é repetido conforme indicado no passo 6 enquanto a condição do passo 7 for atendida, ou seja, enquanto o ganho acumulado na iteração atual for o melhor ganho (G^*) observado até então.

Algoritmo 1 Heurística de Lin-Kernighan (Kernighan e Lin (1973))

1: Inicialize uma rota randômica T_1
2: $G^* = 0$ (G^* é o melhor ganho encontrado durante a busca)
3: Escolha arbitrariamente um vértice t_1 e uma aresta x_1
4: Escolha y_1 a partir de t_2 (o outro vértice da aresta x_1) com $g_1 > 0$
5: Selecione o próximo par de arestas x_i e y_i seguindo os critérios:
   (a) x_i deve permitir a reconexão de t_1 em uma rota válida
   (b) x_i não pode ser uma aresta conectada previamente
   (c) y_i não pode ser uma aresta desconectada previamente
   (d) $G_i = \sum_1^i g_i > 0$
   (e) y_i deve permitir a remoção de um novo x_{i+1}
   (f) Verifique o ganho de fechar a rota ao invés de selecionar y_i. Caso seja melhor, feche a rota e cancele y_i
6: Repita o passo 5 selecionando um novo par de arestas x_{i+1} e y_{i+1}
7: Pare quando $G_i \leq G^*$
8: Se $G^* > 0$, atualize $T_1 = T_1^{'}$ e reinicie a busca a partir do passo 2
9: Se $G^* = 0$, continue a busca a partir das seguintes alternativas:
   (a) Escolha outro y_2
   (b) Escolha outro x_2
   (c) Escolha outro y_1
   (d) Escolha outro x_1
   (e) Escolha outro t_1
10: T_1^* é encontrada quando $G^* = 0$ para todas as escolhas possíveis
11: Reinicie j vezes a partir do passo 1 ($j \geq 0$)
12: $T^* = min\{f(T_1^*), f(T_2^*), ..., f(T_j^*)\}$

Os passos 5(a) e 5(e) são validações referente à factibilidade da rota. O passo 5(a) verifica que o vértice t_1, inicialmente aberto com a desconexão de x_1, possa ser reconectado a um dos vértices abertos em x_i gerando uma rota factível. O passo 5(e) é uma variação do passo 5(a), porém aplicado à seleção da aresta y_i, a qual é condicionada à aresta x_{i+1}, de modo que esta última, no futuro, atenda o critério do passo 5(a).

Os passos 5(b) e 5(c) atendem o critério de exclusividade dos conjuntos X e Y: as arestas pertentes a um conjunto não devem pertencer ao outro durante um mesmo ciclo do algoritmo, ou seja, enquanto uma nova rota não é formada. Esta estratégia serve para simplificar a busca de modo que não ocorram laços de remoção e inclusão da mesma aresta repetidamente. Após reinicialização do algoritmo a partir do passo 8, esses conjuntos são esvaziados.

O passo 5(d) representa a atualização do ganho total acumulado G_i obtido com a desconexão e reconexão dos pares de arestas (x_i, y_i). O ganho acumulado deve obrigatoriamente ser positivo e crescente para que a busca continue.

O passo 5(f) representa uma propriedade específica da busca onde a seleção de uma aresta a ser conectada y_i, com $i \geq 2$, pode: (a) fechar e gerar uma rota factível, se y_i possuir o vértice aberto t_1, ou (b) continuar a exploração por novas arestas, se t_1 se

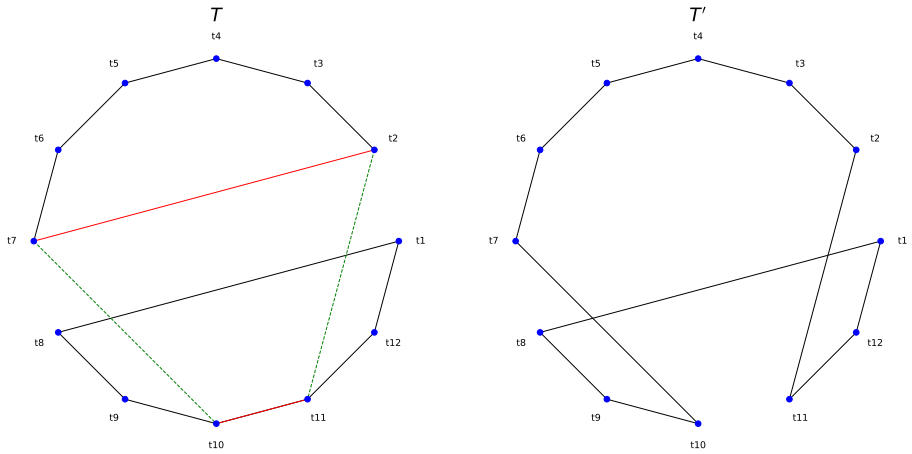
mantiver aberto. Esse passo verifica a opção de fechamento da rota, de modo que se o ganho com o fechamento for melhor que o ganho com a continuidade da exploração, o mesmo é adotado e uma nova rota $T^{'}$ é definida.

O passo 7 representa o critério de parada do algoritmo, atendido quando o ganho atual não é o melhor ganho observado, enquanto o passo 8 representa a reinicialização do algoritmo após atendimento do critério de parada a partir de uma nova rota obtida $T^{'}$.

O passo 9 apresenta o início do processo iterativo realizado quando nenhum ganho é encontrado na configuração atual analisada. Os passos 9(a) até 9(e) são realizados sequencialmente até que uma das alternativas resulte em $G > 0$. É importante ressaltar que a busca por novas opções de arestas e vértices é executada exclusivamente nos níveis 1 e 2, conforme indicado pelos índices dos elementos nos passos 9(a) até 9(e). Essa restrição dos níveis é feita de modo a otimizar computacionalmente o processo de busca por novas opções. O número de níveis será um parâmetro configurado na implementação desenvolvida neste trabalho.

O passo 9(b) representa uma seleção infactível, conforme observada na rota $T_2^{'}$ da Figura 32. Para este caso, uma estratégia específica é definida para seleção de novos vértices e arestas de modo a retornar a factibilidade da rota. Esse retorno somente é aceito se o ganho relativo à construção da nova rota for positivo. Essa exceção à factibilidade foi adotada para aumentar o potencial exploratório do algoritmo sem um comprometimento crítico de sua performance. A Figura 33 ilustra um exemplo no qual uma rota infactível T se transforma em uma rota factível $T^{'}$.

Figura 33 – Conversão de rota infactível.



Fonte: O autor.

O passo 10 representa a finalização do algoritmo, quando nenhuma seleção possível de vértices e arestas resulta em ganho na rota atual. Desta forma, conclui-se que $T = T^*$ e o algoritmo é encerrado. Os passos 11 e 12 são uma opção de reinicialização do processo a

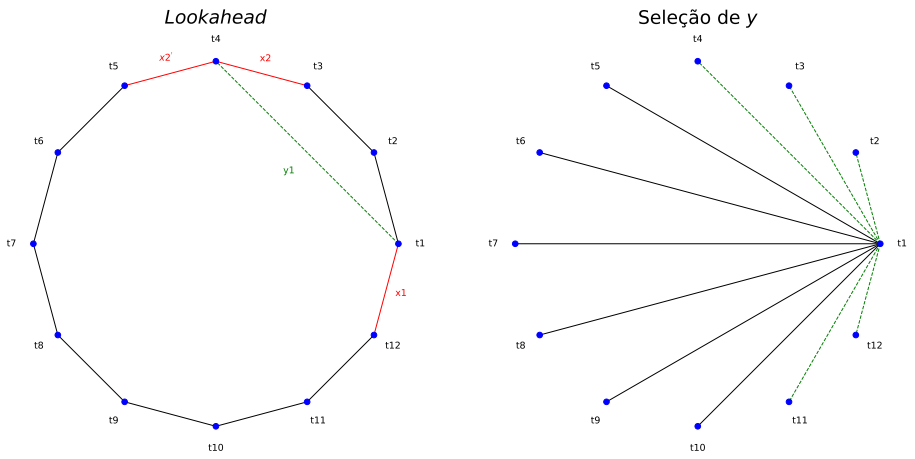
partir de nova rota inicial randômica. Com essa proposta, um conjunto de rotas mínimas é populado de modo que, após todas as reinicializações, a melhor rota seja selecionada entre as rotas disponíveis.

Os passos detalhados no Algoritmo 1 são o suficiente para implementar uma heurística computacionalmente eficiente e encontrar soluções satisfatórias. Porém, além desta base principal da heurística, os autores ainda propõem quatro refinamentos com objetivo de aumentar a performance computacional e direcionar a busca realizada, os quais são listados a seguir:

- *Checkout*: O tempo para concluir que determinada rota T não possui mais possibilidade de melhora (ou seja, que $T = T^*$) é chamado de *checkout*. Este refinamento propõem que, se uma solução repetida for encontrada, o mesmo evite este processo de verificação, uma vez que esta tarefa já foi executada em uma simulação prévia.

- *Lookahead*: Conforme demonstrado no algoritmo da heurística e no exemplo da Figura 34, a seleção de uma aresta y_i implica na desconexão de x_{i+1} até que a rota seja fechada e se torne factível. Desta forma, selecionar arestas apenas considerando o ganho $|x_i| - |y_i|$ não é eficiente, uma vez que x_{i+1} (a aresta à frente) pode ser uma aresta "boa" e anular o ganho previamente calculado. Para otimizar este processo, neste refinamento a seleção de y_i é feita com base em maximizar $|x_{i+1}| - |y_i|$. Como avaliar todas as arestas y_i possíveis consumiria muito tempo computacional, a seleção é limitada aos menores valores de y_i para cada vértice. A Figura 34 exemplifica a seleção das cinco melhores arestas y para o vértice t_1.

Figura 34 – Refinamento *lookahead*.



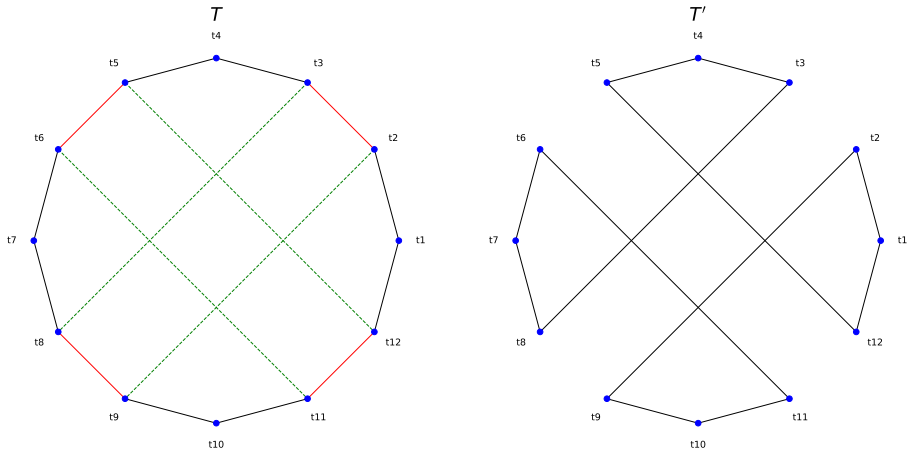
Fonte: O autor.

- *Reduction*: Este refinamento leva em consideração que o conjunto de rotas $\{T_1^*, T_2^*, ..., T_j^*\}$, encontradas a partir das j reinicializações do processo de busca no passo 11, possuem

a maior parte de suas arestas comuns entre si. Assim, a proposta deste refinamento é que a partir de um determinado número de soluções encontradas, arestas comuns à essas rotas não sejam mais removidas em futuras iterações. Além de otimizar a performance computacional, este refinamento direciona a busca por novas arestas em possíveis regiões não otimizadas da rota. A recomendação dos autores são de quatro ciclos de pré-redução, porém esse valor pode ser um parâmetro definido dependendo do tamanho e distribuição dos vértices em cada instância a ser executado o processo heurístico;

- *Double-bridge*: O último refinamento proposto está relacionado à execução de movimentos não-sequenciais de arestas, conforme o movimento chamado *double-bridge*, ilustrado na Figura 35. Ao aplicar este movimento em uma rota T^* (com $G > 0$), seu formato básico é alterado de tal forma que, ao reiniciar o algoritmo no passo 2 utilizando a rota alterada, novos resultados podem ser obtidos, escapando assim de soluções verificadas anteriormente.

Figura 35 – Refinamento *double-bridge*.



Fonte: O autor.

Com o detalhamento da heurística concluído e seus principais refinamentos expostos, todas informações computacionalmente relevantes da publicação de Kernighan e Lin (1973) foram descritas. Conforme destacado no início desta seção, esta publicação não detalha a implementação em linguagem de programação dos elementos que formam o grafo do PCV ou qualquer função necessária para substituição de arestas. Na próxima seção, uma segunda publicação é analisada, a qual não apenas contém uma versão simplificada da heurística de Lin-Kernighan, como também apresenta dados que auxiliam na implementação computacional desta heurística.

4.2 As publicações de Helsgaun (2000) e Mahéo (2017)

A partir da primeira publicação da heurística de Lin-Kernighan em 1973, novos estudos e publicações foram documentadas ao longo dos anos, ao mesmo tempo em que a computação e as linguagens de programação passavam por um processo evolutivo. Com o acesso facilitado a computadores portáteis e o surgimento de linguagens de programação modernas, um número maior de pesquisadores puderam estudar a heurística de Lin-Kernighan, implementar seus próprios métodos variantes e compartilhar os respectivos resultados alcançados com tais métodos.

A publicação de Helsgaun (2000) é uma das principais publicações inspiradas no trabalho de 1973. O objetivo do autor em sua tese é demonstrar variações na estratégia de busca que resultem em uma maior performance computacional e melhores resultados finais do processo heurístico. O autor apresenta também uma versão simplificada da heurística de Lin-Kernighan, exemplos de implementação computacional das estruturas que compõem uma rota e funções básicas de manipulação de vértices e arestas, sendo assim uma das principais fontes de dados para a implementação neste trabalho.

O algoritmo implementado pelo autor, denominado LKH, é atualmente a principal variação da heurística de Lin-Kernighan, sendo recordista em diversas instâncias de grandes dimensões, e continuamente atualizado pelo autor e seu grupo de pesquisa de modo a aumentar ainda mais sua performance. A implementação é disponibilizada gratuitamente no site do autor[2] e desenvolvida na linguagem de programação C. Apesar de ser um algoritmo baseado na heurística de Lin-Kernighan, possui um nível de complexidade elevado, utilizando uma grande quantidade de hiperparâmetros a serem refinados e adaptações para resolução de diversas variações do Problema do Caixeiro Viajante. Em sua primeira versão (LKH-1) ja possuía mais de 4000 linhas de código computacional.

O algoritmo simplificado proposto na publicação de 1998 é demonstrado no Algoritmo 2. Diferente do Algoritmo 1, este é apresentado de forma mais próxima a um pseudocódigo, sendo assim mais fácil o seu entendimento e conversão em um código computacional real.

A principal simplificação incluída no algoritmo está na reação ao encontrar um ganho positivo: enquanto a publicação de 1973 continua a busca por um novo ganho melhor, o modelo simplificado atualiza a rota a partir do primeiro ganho positivo encontrado. Essa condição é descrita no passo 6 do algoritmo simplificado, durante a seleção de x_i e y_i. Com esta simples modificação, a implementação computacional do algoritmo se torna muito mais simples, uma vez que evita, por exemplo, a necessidade de manter cálculos e resultados armazenados em memória e de pesquisar arestas em rotas infactíveis. O restante dos passos descritos são equivalentes aos passos definidos no Algoritmo 1.

[2] <http://akira.ruc.dk/~keld/research/LKH/>. Acesso em 30 de julho de 2023

Algoritmo 2 Heurística de Lin-Kernighan simplificada (Helsgaun (2000))

1: Inicialize uma rota randômica T
2: $i \leftarrow 1$. Selecione t_1
3: Selecione $x_1 = (t_1, t_2) \in T$
4: Selecione $y_1 = (t_2, t_3) \notin T$ de tal forma que $G_1 > 0$
   Caso não seja possível, vá para o passo 12
5: $i \leftarrow i + 1$
6: Selecione $x_i = (t_{2i-1}, t_{2i}) \in T$ de tal forma que
   (a) se t_{2i} é conectado a t_1, o resultado é uma rota $T^{'}$ e
   (b) $x_i \neq y_s$ para todo $s < i$
   Se $T^{'}$ é uma rota melhor que T, $T \leftarrow T^{'}$ e retorne ao passo 2
7: Selecione $y_i = (t_{2i}, t_{2i+1}) \in T$ de tal forma que
   (a) $G_i > 0$,
   (b) $y_i \neq x_s$ para todo $s < i$ e
   (c) x_{i+1} existe
   Se tal y_i existir, mova até o passo 5
8: Se existe uma opção não testada de y_2, $i \leftarrow 2$ e mova até o passo 7
9: Se existe uma opção não testada de x_2, $i \leftarrow 2$ e mova até o passo 6
10: Se existe uma opção não testada de y_1, $i \leftarrow 1$ e mova até o passo 4
11: Se existe uma opção não testada de x_1, $i \leftarrow 1$ e mova até o passo 3
12: Se existe uma opção não testada de t_1, mova até o passo 2
13: Pare (ou mova até o passo 1)

A partir do algoritmo simplificado de Helsgaun (2000), Mahéo (2017) desenvolveu a heurística de Lin-Kernighan simplificada na linguagem de programação Python. Esta foi a referência inicial utilizada para desenvolvimento da implementação deste trabalho. Como a base do algoritmo simplificado e do algoritmo completo são semelhantes, a conversão do primeiro em um pseudocódigo é detalhada a seguir.

A transformação do Algoritmo 2 em um pseudocódigo se inicia com a identificação do laço principal definido no passo 6. O algoritmo é reiniciado a partir do critério do ganho positivo $G_i > 0$, ou seja, sempre que uma rota $T^{'}$ pode ser obtida a partir de T com $f(T^{'}) < f(T)$. Desta forma, o laço de repetição principal deve ser executado enquanto uma nova rota melhorada $T^{'}$ for encontrada a partir da rota atual T, conforme demonstrado no Algoritmo 3.

Algoritmo 3 Laço principal da heurística de Lin-Kernighan simplificada (Mahéo (2017))

1: *rota_melhorada* ← **verdadeiro**
2: **enquanto** *rota_melhorada* = **verdadeiro faça**
3: *rota_melhorada* ← *melhorar_rota*()
4: **fim enquanto**

A função *melhorar_rota*, declarada no Algoritmo 3, é executada um número indefinido de vezes, retornando valor booleano verdadeiro quando uma nova rota melhor é encontrada e, consequentemente, reiniciando o laço de repetição. Quando o valor retornado

pela função é o booleano falso, o laço é interrompido e a rota atual é a melhor rota calculada. Os detalhamentos da função *melhorar_rota* são exibidos no Algoritmo 4.

Algoritmo 4 Função para encontrar uma rota melhor simplificada (Mahéo (2017))

```
1: função melhorar_rota
2:     para t_1 ∈ T faça
3:         para t_2 ∈ {t_1próximo, t_1prévio} faça
4:             x_1 ← (t_1, t_2)
5:             X ← {x_1}
6:             para t_3 ∈ vizinhos(t_2) faça
7:                 y_1 ← (t_2, t_3)
8:                 Y ← {y_1}
9:                 G ← x_1custo − y_1custo
10:                se G > 0 então
11:                    se encontrar_x() = verdadeiro então
12:                        retorna verdadeiro
13:                    fim se
14:                fim se
15:            fim para
16:        fim para
17:    fim para
18:    retorna falso
19: fim função
```

Os passos 2, 3 e 6 representam os testes de todas as respectivas opções possíveis para t_1, x_1 e y_1, conforme os passos 10, 11 e 12 descritos no Algoritmo 2. O passo 2 seleciona todos os vértices iniciais possíveis t_1, enquanto o passo 3 seleciona os dois vértices pertencentes à rota e adjacentes a t_1. O passo 6 seleciona todos os vértices possíveis e factíveis diferentes de t_2 de modo a formar a nova aresta conectada y_1 no passo 7, a qual substitui a aresta desconectada x_1 definida no passo 4.

Os conjuntos X e Y são inicializados respectivamente nos passos 5 e 8, e o valor do custo inicial é calculado no passo 9. A condição indicada no passo 10 verifica se a busca por novas arestas deve continuar ou ser interrompida. Se o ganho definido é negativo ou zero, as alternativas dos laços de repetição serão testadas sequencialmente a partir dos passos 6, 3 e 2. Quando $T = T^*$, nenhuma alternativa testada apresentará ganho maior do que zero, logo a função *melhorar_rota* retornará o valor booleano falso e o Algoritmo 3 será finalizado.

Uma vez o ganho encontrado pela substituição da aresta x_1 por y_1 sendo positivo, a seleção da próxima aresta a ser removida é executada através da função *encontrar_x*, a qual é descrita no Algoritmo 5. Para simplificar a demonstração deste algoritmo, a verificação dos critérios de factibilidade não será descrita, sendo implícita durante a seleção dos vértices e arestas. Como exemplo, a seleção de x_i pode formar uma rota infactível conforme a rota T_2' da Figura 32.

Algoritmo 5 Função para encontrar uma nova aresta x (Mahéo (2017))

1: **função** *encontrar_x*
2: **para** $t_{2i} \in \{t_{2i-1_{próximo}}, t_{2i-1_{prévio}}\}$ **faça**
3: $x_i \leftarrow (t_{2i-1}, t_{2i})$
4: **se** $x_i \notin Y$ **então**
5: $X \cup \{x_i\}$
6: $y_{fecha} = y_i \leftarrow (t_{2i}, t_1)$
7: $G \leftarrow G + (x_{i_{custo}} - y_{i_{custo}})$
8: **se** $G > 0$ **então**
9: $T^{'} = (T - X) \cup Y$
10: $T \leftarrow T'$
11: **retorna verdadeiro**
12: **senão**
13: **retorna** *encontrar_y*()
14: **fim se**
15: **fim se**
16: **fim para**
17: **retorna falso**
18: **fim função**

A seleção da próxima aresta x_i, com $(i \geq 2)$, a ser removida é baseada nos vértices adjacentes à t_{2i}, conforme passos 2 e 3 do algoritmo. O passo 4 demonstra a verificação do critério de exclusividade do conjunto Y, de modo que a nova aresta removida não pode pertencer a este conjunto de arestas conectadas. Se x_i é factível e atende o critério de exclusividade, esta aresta é selecionada e adicionada ao conjunto X de arestas removidas, conforme passo 5.

O passo 6 representa a principal simplificação proposta por Helsgaun (2000), na qual a próxima aresta a ser conectada y_i é a aresta que fecha a rota, nomeada y_{fecha}, ligando t_{2i} até o primeiro vértice aberto t_1. A partir das arestas x_i e y_i definidas, o ganho total acumulado é atualizado no passo 7 e, se o mesmo se mantiver positivo, uma nova rota $T^{'}$ melhorada pode ser obtida a partir da operação com os conjuntos de arestas demonstrado no passo 9. A partir da atualização da rota no passo 10, o valor booleano verdadeiro é retornado da função e propagado para os Algoritmos 4 e 3 de modo que o ciclo de busca seja reiniciado a partir da nova rota $T^{'}$.

Os passos 12 e 13 representam a condição na qual a seleção do par de arestas x_i e y_i resultam em $G < 0$, de tal forma que a substituição das arestas da rota atual resultaria em uma rota de maior custo. Neste caso, ao invés de fechar a rota, um processo de busca por outro $y_i \neq y_{fecha}$ é iniciado com o objetivo de atender o critério de ganho positivo. A seleção de y_i implica na futura seleção de uma nova aresta a ser desconectada x_{i+1} e, consequentemente, uma nova aresta de fechamento a ser conectada y_{i+1}. A partir desta propriedade, a função *encontrar_y* sugestivamente deverá executar novamente a função *encontrar_x* para realizar a seleção de x_{i+1} e y_{i+1} em um processo cíclico.

A seleção da nova aresta y_i é detalhada no Algoritmo 6. Como este algoritmo não determina uma nova rota, a qual só é definida a partir do Algoritmo 5, o mesmo pode ser descrito em poucas linhas de programação.

Algoritmo 6 Função para encontrar uma nova aresta y (Mahéo (2017))

1: **função** *encontrar_y*
2: **para** $t_{2i+1} \in vizinhos(t_{2i})$ **faça**
3: $y_i = (t_{2i}, t_{2i+1})$
4: $G \leftarrow G + (x_{i_{custo}} - y_{i_{custo}})$
5: **se** $G > 0$ e $y_i \notin X$ **então**
6: $Y \cup \{y_i\}$
7: **retorna** *encontrar_x*()
8: **fim se**
9: **fim para**
10: **retorna falso**
11: **fim função**

A partir do vértice t_{2i}, o qual previamente não pode ser fechado em t_1, um novo vértice t_{2i+1} é selecionado a fim de se obter y_i, conforme passos 2 e 3, que atenda o critério de ganho positivo e de exclusividade, verificação realizada no passo 5. Uma vez encontrada a aresta y_i, o passo 6 demonstra a inclusão desta no conjunto de arestas conectadas Y e então a função *encontrar_x* é executada novamente no passo 7. Caso nenhum y_i seja encontrado, a função retorna o valor booleano falso, consequentemente o Algoritmo 5 retorna o mesmo valor e o Algoritmo 4 continua a busca por uma nova opção possível de vértice ou aresta a ser testada.

Conforme dito anteriormente, as funções *encontrar_x* e *encontrar_y* são executadas alternadamente até que os conjuntos de arestas X e Y sejam definidos de modo a gerar uma nova rota T' com $f(T') < f(T)$. A partir da nova rota definida, a função *melhorar_rota* retorna o valor booleano verdadeiro e o ciclo principal da heurística é reiniciado através do Algoritmo 3 até que nenhuma rota melhor seja encontrada, concluindo então que $T = T^*$.

Com a definição detalhada do algoritmo simplificado nesta seção, o desenvolvimento da heurística completa proposto por Kernighan e Lin (1973), incluindo os quatro refinamentos de busca e detalhamentos relacionados às estruturas que compõem a rota serão abordados no próximo capítulo.

5 IMPLEMENTAÇÃO COMPUTACIONAL

Neste capítulo serão apresentadas as etapas necessárias para implementação computacional da Heurística de Lin-Kernighan e o resultado de tal implementação em um pacote na linguagem de programação Python.

5.1 Estruturação para implementação em Python

A primeira consideração e justificativa a ser feita antes de prosseguir com o conteúdo desta seção está relacionada com a motivação na implementação da heurística de Lin-Kernighan, uma vez que tal atividade se tornou um dos objetivos deste trabalho. Em meio às pesquisas bibliográficas e consulta de implementações computacionais, nenhuma implementação clara e objetiva da heurística de 1973 foi encontrada. Enquanto algumas implementações apresentam variações muito simplificadas da heurística completa, como a heurística 3-Opt proposta por Croes (1958) e a simplificação de Lin-Kernighan proposta por Helsgaun (2000), outras implementações, como a variação *Chained Lin-Kernighan*, proposta por Applegate, Cook e Rohe (2003) e LKH proposta por Helsgaun (2000), são muito complexas e não indicadas como ponto de partida para compreensão e estudo da heurística completa.

Considerando essa condição, a implementação da heurística de Lin-Kernighan, por si só, é relevante e de grande colaboração para a comunidade científica, principalmente aos estudiosos do PCV e seus respectivos métodos heurísticos de resolução. É possível que ao se depararem com a publicação de 1973, pesquisadores tenham se sentido intimidados com tamanha complexidade, algo muitas vezes inesperado tendo como referência métodos heurísticos mais simples e populares. Somada à essa dificuldade, a escassez de documentação nos códigos computacionais disponíveis torna ainda mais desafiador sua respectiva leitura e compreensão.

A seleção da linguagem de programação Python para implementação computacional foi feita com base em dois motivos principais: (a) a simplicidade da linguagem, de modo a obter uma sintaxe limpa e simples, facilitando assim o entendimento do algoritmo e (b) a popularidade da linguagem, de tal forma que a publicação do código computacional alcance um número maior de pesquisadores. Em contrapartida, é conhecido que esta linguagem possui gargalos de processamento que a torna mais lenta quando comparada à linguagens como C, Java ou Rust, porém estas não atendem tão bem quanto o Python os critérios de simplicidade e popularidade.

Conforme dito na seção anterior, determinados detalhes da implementação computacional não serão expostos neste trabalho, sendo indicada a consulta da página de documentação da heurística e seus respectivos arquivos de programação disponíveis em Castro (2022). Esses arquivos foram detalhados com comentários e definições de funções de modo que funcione como documentação auxiliar para compreensão da heurística.

Uma vez pontuados os motivos da implementação computacional deste trabalho, será detalhada a implementação dos objetos que compõem o Problema do Caixeiro Viajante, mais especificamente o *vértice*, a *aresta* e a *rota*. Além desses, um tipo de objeto controlador denominado *pcv* é definido para orquestrar a execução da heurística e seus refinamentos. A implementação utiliza o conceito de programação orientada a objetos para que cada objeto seja definido por uma classe de programação contendo os respectivos atributos e funções de cada elemento, mantendo assim o código computacional limpo, organizado e de fácil entendimento.

A classe *vértice* define o elemento primário utilizado para construção dos demais elementos do problema. Esta é desenvolvida computacionalmente a partir de um tipo de estrutura de dados chamada Lista Duplamente Encadeada (*Doubly Linked List*), importante para garantir a eficiência computacional do algoritmo uma vez que, a partir dela, substituições de arestas poderão ser feitas com complexidade computacional linear n. A Tabela 5 apresenta os principais parâmetros definidos para a classe *vértice* e a Figura 36 demonstra o conceito de uma Lista Duplamente Encadeada.

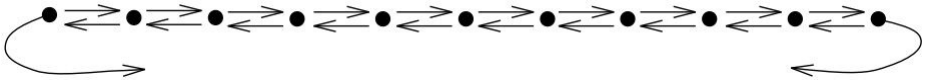
Tabela 5 – Definições para a classe *vértice*.

parâmetro	**tipo**	**definição**
id	número inteiro	identificador único
pos	número inteiro	posição na rota
pred	vértice	vértice predecessor
succ	vértice	vértice sucessor
x	número flutuante	coordenada cartesiana x
y	número flutuante	coordenada cartesiana y
z	número flutuante	coordenada cartesiana z
eq	função	comparação (=) entre dois vértices
gt	função	comparação (>) entre dois vértices

Fonte: O autor.

O parâmetro *id* é utilizado como identificador único do vértice e mantido constante durante todo o processo do algoritmo, com objetivo de facilitar operações indexadas.

Figura 36 – Lista Duplamente Encadeada.



Fonte: Adaptado de Applegate, Bixby e Cook (1994).

O parâmetro *pos* define a posição de cada vértice na rota, sendo alterado durante a substituição de arestas e construção de novas rotas. Este parâmetro pode assumir qualquer valor inteiro, contanto que seja sequencial entre o início e fim de uma rota, propriedade que simplifica determinados cálculos de reordenamento dos vértices.

Os parâmetros *pred* e *succ* definem a estrutura de dados conhecida como Lista Duplamente Encadeada, fazendo a amarração entre cada vértice e seus respectivos vizinhos predecessor e sucessor, e devem ser redefinidos a partir da formação de novas rotas. Os parâmetros x, y e z definem as coordenadas cartesianas do vértice, tanto para o caso em duas dimensões (instâncias acadêmicas) como para o caso em três dimensões (instâncias industriais).

A função comparativa *eq* é utilizada para identificar igualdade entre dois vértices, devendo ser cuidadosamente definida para absorver condições específicas de determinadas instâncias que possuem vértices com mesma coordenada espacial. A função *gt* é utilizada para garantir a propriedade simétrica das arestas no PCV simétrico, mantendo sempre o mesmo ordenamento dos vértices dentro da classe *aresta*.

A classe *aresta* é definida por dois objetos da classe *vértice*, v_1 e v_2, e uma função comparativa *eq* que executa a comparação direta e também a comparação simétrica entre duas arestas, na qual deve ser obtido que $(v_1, v_2) = (v_2, v_1)$. Os parâmetros para a classe *aresta* são definidos na Tabela 6.

Tabela 6 – Definições para a classe *aresta*.

parâmetro	**tipo**	**definição**
v_1	vértice	primeiro vértice
v_2	vértice	segundo vértice
eq	função	comparação (=) entre duas arestas

Fonte: O autor.

A partir da definição das classes *vértice* e *aresta*, a classe *rota* pode ser definida como um objeto composto por parâmetros derivados e combinados dessas duas classes básicas. Esses parâmetros são demonstrados na Tabela 7.

Tabela 7 – Definições para a classe *rota*.

parâmetro	**tipo**	**definição**
vértices	lista	lista de vértices que formam a rota
arestas	conjunto	conjunto de arestas que formam a rota
custo	número flutuante	custo da rota
dimensão	número inteiro	dimensão da rota
trocas	lista	lista com as trocas de arestas
embaralhar	função	embaralha a rota
restaurar	função	restaura a rota
entre	função	avalia se um vértice está entre outros dois
troca_factível	função	executa uma troca de arestas factível
troca_infactível	função	executa uma troca de arestas infactível
double_bridge	função	executa uma troca de arestas *double bridge*

Fonte: O autor.

O parâmetro *vértices* é determinado em uma coleção ordenada do tipo lista a fim de utilizar a sequência da mesma para definição da rota inicial e da indexação global dos vértices. O tamanho dessa lista resulta na quantidade de vértices da rota, valor registrado no parâmetro *dimensão*. O parâmetro *arestas* é definido em uma coleção não-ordenada do tipo conjunto com objetivo de otimizar operações de adição e subtração de elementos, como as operações realizadas nos conjuntos de arestas desconectadas (X) e conectadas (Y).

O parâmetro *custo* de uma rota recebe a somatória dos custos das arestas definidas no conjunto *arestas*. Esse cálculo é realizado através de um laço de repetição que passa por todas arestas acumulando o seu respectivo custo dentro do parâmetro *custo*. Para facilitar esse cálculo, uma matriz quadrada de custos (*dimensão* x *dimensão*) é inicializada uma única vez na classe *tsp* utilizando como indexação o parâmetro *id* de cada vértice.

O parâmetro *trocas* é uma lista que recebe o histórico de todas as operações de troca de arestas realizadas durante o processo heurístico, funcionando como um tipo de armazenamento e memória que permite o cancelamento de qualquer operação quando combinada com a função *restaurar*. Essa lista é esvaziada sempre que o algoritmo é reiniciado a partir de uma nota rota.

A função *embaralhar* é utilizada para gerar uma rota randômica, sendo este o primeiro passo definido na heurística de Lin-Kernighan. O embaralhamento é feito mantendo a indexação global dos vértices de modo que não seja necessário o recálculo da

matriz de custos. A função *entre* é utilizada para identificar se um determinado vértice está posicionado entre outros dois vértices da rota. O uso dessa função esta ligado à busca de vértices especiais em rotas infactíveis e pode ser definida através do parâmetro *pos* de cada vértice avaliado.

A função *troca_factível* realiza a troca de duas arestas da rota por outras duas novas arestas que resultem em uma nova rota factível, enquanto a função *troca_infactível* realiza operação equivalente, mas que resulta em uma nova rota infactível. O processo de troca factível de arestas implica em recálculo do parâmetro *pos* dos vértices, enquanto o processo de troca infactível dispensa este recálculo, o qual é feito apenas após retomada da factibilidade da rota. A eficiência computacional da heurística está intimamente ligada à eficiência destas funções.

A função *double_bridge* executa um tipo de troca de arestas não-sequencial, conforme definido pelo refinamento *double bridge*, a partir da definição de quatro arestas a serem desconectadas e quatro arestas a serem conectadas que satisfaçam a condição de não-sequencialidade.

O último objeto, definido como *pcv*, representa o objeto controlador do processo heurístico a ser realizado no Problema do Caixeiro Viajante, possuindo diversos parâmetros relativos aos refinamentos da heurística de Lin-Kernighan e a execução do processo de busca de arestas. Esta classe é definida na Tabela 8.

O parâmetro *custos* é representado por uma matriz quadrada simétrica populada com o custo de todas as combinações possíveis de pares de vértices. Como a linguagem de programação Python não oferece um tipo nativo para matrizes, foi utilizado o tipo de objeto dicionário e executado o chaveamento a partir de $(v_i.id, v_j.id)$ aonde id representa o parâmetro identificador único dos vértices i e j. A definição do custo das arestas é feita na classe controladora com o objetivo de otimizar e simplificar o acesso à este dado, o qual é computado uma única vez durante inicialização do algoritmo.

O parâmetro *vizinhos_próximos* é definido como um objeto do tipo dicionário, correlacionando cada vértice do problema aos seus respectivos vizinhos mais próximos, pré-requisito necessário para aplicação do refinamento, e respectiva função, *lookahead*. O parâmetro *profundidade* define o número máximo de vizinhos a serem considerados em cada nível de busca i, sendo este incrementado à cada nova seleção do par de arestas substituídos (x_i, y_i). A publicação de 1973 sugere a utilização do modesto valor $profundidade = (5, 5)$, resultando em apenas os dois primeiros níveis de busca ativos, cada um utilizando os cinco vizinhos mais próximos do vértice selecionado.

A parametrização do refinamento *reduction* é executada a partir dos parâmetros *nível_de_redução* e *ciclo_de_redução*. O primeiro define a profundidade de busca a partir da qual a redução de arestas é aplicada enquanto o segundo define o ciclo da heurística a

Tabela 8 – Definições para a classe *pcv*.

parâmetro	**tipo**	**definição**
custos	dicionário	custo entre todas as combinações de vértices
vizinhos_próximos	dicionário	melhores arestas a partir de cada vértice
profundidade	tupla	número máximo de vizinhos do *lookahead*
nível_de_redução	número inteiro	nível inicial para aplicação do *reduction*
ciclo_de_redução	número inteiro	ciclo inicial para aplicação do *reduction*
arestas_de_redução	conjunto	conjunto de arestas da aplicação do *reduction*
ciclo	número inteiro	ciclo atual de busca
ciclo_max	número inteiro	número máximo de ciclos de busca
soluções	conjunto	conjunto para aplicação do *checkout*
ganhos_fecha	lista	ganhos de fechamento da rota
lookahead	função	seleciona os melhores vizinhos de um vértice
busca_factível	função	executa busca por arestas factíveis
busca_infactível	função	executa busca por arestas infactíveis
busca_double_bridge	função	executa refinamento *double-bridge*

Fonte: O autor.

partir do qual o refinamento é aplicado, valor equivalente ao número de soluções a serem encontrados antes da redução. O parâmetro *arestas_de_redução* define o conjunto de arestas comuns às soluções locais encontradas, as quais são obtidas a partir da intersecção de suas respectivas arestas.

O parâmetro *ciclo* é um contador simples do ciclo do processo heurístico, o qual é incrementado a partir da reinicialização do algoritmo após obtida uma solução. A tarefa de reinicialização é executada um número máximo de vezes definido pelo parâmetro *ciclo_max*.

Ao final de cada ciclo, dois valores mapeados ($hash$) são armazenados no parâmetro *soluções* utilizando a sequência dos identificadores únicos de cada vértice na rota obtida, partindo sempre de um mesmo vértice inicial e utilizando as duas direções possíveis da rota. Como exemplo, se ao final de um ciclo de busca é obtida uma rota definida pela sequência de vértices $(v_1, v_4, v_3, v_5, v_2)$, o parâmetro *soluções* recebe os valores mapeados $hash(v_1, v_4, v_3, v_5, v_2)$ e $hash(v_1, v_2, v_5, v_3, v_4)$. Ao definir uma nova rota, em um novo ciclo, é verificado se o parâmetro *soluções* contém os dois valores mapeados da rota nova e, caso positivo, o processamento residual chamado de *checkout* é evitado.

Durante a busca por novas arestas a serem removidas e conectadas existirá um ganho relativo à continuidade de exploração por novas arestas e um ganho relativo à parada da busca e fechamento da nova rota. Enquanto o primeiro ganho é monitorado e atualizado ao longo do processo de busca, o segundo é armazenado sequencialmente na lista *ganhos_fecha*. Uma vez sendo o ganho exploratório menor que um ganho de fechamento prévio, o processo de busca é interrompido, a melhor rota possível é restaurada a partir do índice no qual o melhor ganho foi encontrado e a lista *ganhos_fecha* é esvaziada para reinicialização do processo de busca.

As funções *busca_factível* e *busca_infactível* representam as operações exploratórias para encontrar substituições de arestas e formar uma nova rota melhorada. A função *busca_infactível* é executada apenas no segundo nível de busca, quando a troca das arestas (x_1, x_2) pelas arestas (y_1, y_2) resulta em duas sub-rotas isoladas, tornando essa solução infactível. Para este caso, um processo de busca especial é realizado com objetivo de retornar a factibilidade da rota e, ao mesmo tempo, manter o ganho positivo. Para todos os demais casos e níveis de busca, apenas a função *busca_factível* é utilizada, a qual obrigatoriamente deve selecionar arestas que atendam a factibilidade da nova rota formada. A função *busca_double_bridge* executa o refinamento de mesmo nome, o qual procura por um conjunto de quatro arestas a serem desconectadas e reconectadas de maneira não-sequencial e que formem uma nova rota melhorada.

Uma vez definidas as principais classes que compõem o Problema do Caixeiro Viajante, os Algoritmos 3 e 4 podem ser adaptados de modo a incluir os detalhamentos propostos pela heurística completa. O Algoritmo 7 apresenta a adaptação do laço principal da heurística de Lin-Kernighan.

Algoritmo 7 Laço principal da heurística de Lin-Kernighan completa (Castro (2022))

1: **enquanto** $pcv.ciclo < pcv.ciclo\_max$ **faça**
2: $rota\_melhorada \leftarrow$ **verdadeiro**
3: **enquanto** $rota\_melhorada =$ **verdadeiro faça**
4: $rota\_melhorada \leftarrow melhorar\_rota()$
5: **fim enquanto**
6: **se** $T'.custo < T^*.custo$ **então**
7: $T^* \leftarrow T'$
8: **fim se**
9: $pcv.soluções \cup \{hash(T')\}$ ▷ refinamento *checkout*
10: $pcv.arestas\_de\_redução \cap \{T'\}$ ▷ refinamento *reduction*
11: $pcv.busca\_double\_bridge()$ ▷ refinamento *double-bridge*
12: $T \leftarrow T'.embaralhar()$
13: $pcv.ciclo \leftarrow pcv.ciclo + 1$
14: **fim enquanto**

A primeira diferença evidenciada pelo passo 1 é o laço de repetição baseado em uma quantidade máxima de ciclos de busca, ao invés do ciclo único utilizado na versão

simplificada. Os passos 2 até 5 são similares ao algoritmo simplificado, porém a função *melhorar_rota* executada no novo algoritmo é adaptada a partir da heurística completa. Após finalizado o processo heurístico, o passo 6 demonstra a verificação da melhor solução T^* e sua respectiva atualização caso a solução atual T' seja a melhor solução encontrada até o ciclo atual em processamento.

O passo 9 representa a preparação necessária para o refinamento *checkout*, executada a partir da inclusão dos valores mapeados da nova rota dentro do conjunto *soluções*. Assim, a busca por novas rotas a partir da função *busca_factível* pode ser interrompida caso o valor mapeado da nova rota encontrada dentro desta função seja igual a um dos valores do conjunto *soluções*.

O parâmetro *arestas_de_redução* é atualizado no passo 10 por meio da intersecção entre as arestas da nova rota T' e as arestas reduzidas de passos anteriores. Esse parâmetro é inicializado a partir da melhor solução obtida no primeiro ciclo de busca, então a intersecção passa a ser executada nos demais ciclos. Após este passo, a execução do refinamento *double-bridge* é realizada no passo 11. As rotas T' e T^* são atualizadas caso o custo das mesmas seja reduzido a partir da execução do refinamento.

Antes da reinicialização do laço principal, uma nova rota inicial é obtida pelo embaralhamento dos vértices e, consequentemente, das arestas que formam rota atual T' e o parâmetro *ciclo* é atualizado de modo que o laço externo do algoritmo seja interrompido quando o parâmetro atingir um valor máximo pré-definido.

Tendo definido os novos passos computacionais relacionados ao laço principal da heurística completa, as atualizações executadas na função *melhorar_rota* de modo a também aproximá-la da heurística de Lin-Kernighan completa são demonstradas no Algoritmo 8.

O passo 6 apresenta a primeira novidade desta função, relacionada à seleção do vértice t_4 em conjunto com o vértice t_3, de modo a definir $x_2 = (t_3, t_4)$ durante a seleção de $y_1 = (t_2, t_3)$. Esta estratégia de seleção representa o refinamento *lookahead*, no qual os melhores pares de arestas (x_{i+1}, y_i) são selecionados, ao contrário do algoritmo simplificado que apenas seleciona a aresta y_i. Tal refinamento continuará sendo aplicado internamente na função *busca_factível*, enquanto o nível de busca for menor que a quantidade de níveis definidos pelo parâmetro *profundidade*.

O passo 11 representa uma nova condição do algoritmo completo, na qual a primeira troca de arestas representada pelos vértices (t_1, t_2, t_3, t_4) é avaliada com relação à sua factibilidade e o processo de busca é direcionado com base nesta condição. No algoritmo simplificado todas as seleções de vértices são consideradas apenas quando a factibilidade da seleção é atendida, porém no algoritmo completo existe uma exceção à essa regra no primeiro nível de busca.

Algoritmo 8 Função para encontrar uma rota melhor completa (Castro (2022))

```
1: função melhorar_rota
2:     para t_1 ∈ T faça
3:         para t_2 ∈ {t_1_próximo, t_1_prévio} faça
4:             x_1 ← (t_1, t_2)
5:             X ∪ {x_1}
6:             para (t_3, t_4) ∈ pcv.lookahead(t_2) faça          ▷ refinamento lookahead
7:                 y_1 ← (t_2, t_3)
8:                 Y ∪ {y_1}
9:                 G ← x_1_custo − y_1_custo
10:                se G > 0 então
11:                    se a troca (t_1, t_2, t_3, t_4) é factível então
12:                        pcv.busca_factível()
13:                    senão
14:                        pcv.busca_infactível()
15:                    fim se
16:                    se max(pcv.ganhos_fecha) > 0 então
17:                        T' = (T − X) ∪ Y
18:                        T ← T'
19:                        retorna verdadeiro
20:                    fim se
21:                fim se
22:            fim para
23:        fim para
24:    fim para
25:    retorna falso
26: fim função
```

Durante execução das funções de busca, a lista *ganhos_fecha* é atualizada com os valores de ganho para o fechamento da rota em cada nível de busca. Ao final deste processo, conforme passo 16, o valor máximo dessa lista é usado para restaurar a melhor rota encontrada, a qual é utilizada como ponto de partida para reinicialização do processo de busca a partir do passo 3 do Algoritmo 7. Para o algoritmo simplificado, essa estratégia é simplificada pela atualização da rota a partir do primeiro ganho positivo encontrado, sem uma análise encadeada de uma sequência de ganhos.

Conforme dito no início desta seção, alguns detalhamentos específicos, como as funções de busca da classe *pcv*, não serão descritos neste trabalho, uma vez que requerem uma riqueza de detalhes além do escopo desta apresentação. Além disso, essas funções, bem como toda a implementação computacional, se encontram amplamente documentadas e disponíveis em Castro (2022). Com o detalhamento da implementação computacional da heurística de Lin-Kernighan completa tendo sido finalizado, as próximas seções apresentam os detalhes adicionais implementados em linguagem Python, iniciando pelos testes unitários escritos para validação das classes descritas nesta seção, em especial a classe *rota*.

5.2 Testes unitários

Uma das maiores dificuldades encontradas durante a implementação da heurística de Lin-Kernighan foi a validação das funções de substituição de arestas da classe *rota*. Essas funções devem ser capazes de atualizar os parâmetros *pred* e *succ* dos elementos da classe *vértice* de modo a representar a substituição das arestas desconectadas e conectadas e, ao mesmo tempo, ser operações com complexidade computacional reduzida para não afetar a performance do algoritmo. A heurística completa de Lin-Kernighan implica no desenvolvimento e validação de várias condições específicas da substituição factível e infactível de arestas e na possibilidade de reverter uma determinada cadeia de substituições, resultando em complexidades adicionais a serem verificadas e implementadas computacionalmente.

O desenvolvimento inicial dessas funções com arestas e verificações dos principais casos foi feito utilizando uma instância de pequeno porte de modo a analisar seu grafo e as respectivas substituições, avaliando as possibilidades de pivotamento de vértices e necessidades de correção do sentido de segmentos reconectados, a fim de se ter uma pré-validação de todas as funções a serem desenvolvidas. Essa aproximação tem a vantagem de possuir a observação gráfica da operação de substituição, porém é um trabalho moroso e de difícil replicação.

Considerando o grande número de combinações possíveis de substituição de arestas, as dificuldades da validação manual e a necessidade de automatizar este processo, foram desenvolvidos testes unitários para validação de cada função computacional de substituição de arestas utilizando a biblioteca Python chamada *unittest*. Os testes unitários, como definido no próprio nome, são utilizados para automatizar o processo de validação de unidades de código computacional, os quais se encaixam perfeitamente no contexto das funções de substituição de arestas. A Tabela 9 lista os principais testes unitários desenvolvidos para a classe *rota*, destacando os testes relativos às operações de substituição de arestas. Todos os testes desenvolvidos para as classes *vértice*, *aresta* e *rota* encontram-se disponíveis em Castro (2022).

5.3 Plotagem

Uma das principais características do PCV, ressaltada em instâncias com custo euclideano, é a possibilidade de visualização dos seus elementos em um grafo. Além da avaliação numérica do custo da rota final, é importante representá-la graficamente para obter conclusões explícitas a respeito da execução do algoritmo. A partir desta importância, uma ferramenta adicional de plotagem de rotas foi desenvolvida na própria linguagem de programação Python.

Tabela 9 – Principais testes unitários para a classe *rota*.

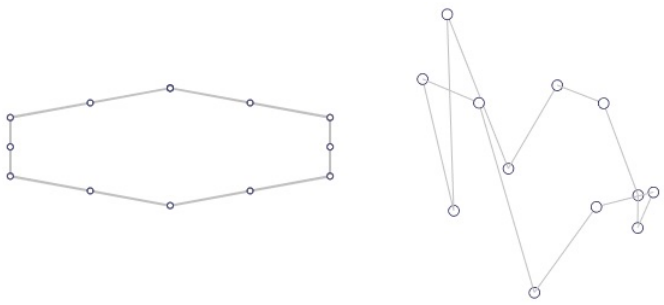
função	**definição**
factibilidade_da_troca	Verifica se a troca a ser executada é factível
infactibilidade_da_troca	Verifica se a troca a ser executada é infactível
troca_factível	Verifica o resultado de trocas factíveis
troca_infactível	Verifica o resultado de trocas infactíveis
troca_especial_1	Verifica o resultado de troca usada na *busca_infactível*
troca_especial_2	Verifica o resultado de troca usada na *busca_infactível*
troca_double_bridge	Verifica o resultado de trocas *double-bridge*
restaurar_rota	Verifica o resultado da restauração da rota

Fonte: O autor.

A biblioteca de visualização e construção de gráficos em Python denominada *matplotlib* é atualmente a biblioteca gráfica mais popular dessa linguagem justificado pela sua facilidade de uso, variedade de tipos de gráfico e qualidade das imagens geradas. Todos os grafos exibidos na Seção 4.1 foram construídos utilizando esta biblioteca. Contudo, uma de suas desvantagens é a limitação na construção de gráficos 3D, tendo apresentado baixa performance em testes realizados com instâncias de média e grande dimensão.

Por esse motivo, a ferramenta para construção de gráficos selecionada foi a biblioteca *plotly*, a qual também é de fácil utilização e que produz gráficos interativos de alta qualidade e performance, exibidos em um navegador *web*. A Figura 37 demonstra exemplos de gráficos 2D e 3D gerados para uma instância simplificada.

Figura 37 – Exemplo de gráficos 2D (esq.) e 3D (dir.) com *plotly*.



Fonte: O autor.

As funções de plotagem utilizam o arquivo com extensão *.tsp*, no padrão TSPLIB visto em Reinelt (1991), como entrada para geração do gráfico, fazendo as conversões necessárias para transformação dos elementos textuais em elementos gráficos. As funções

de plotagem foram incluídas e documentadas no módulo disponibilizado em Castro (2022) de modo a reforçar aos visitantes e usuários que, além da resolução numérica do problema, o pacote disponibilizado também pode gerar graficamente os resultados obtidos.

5.4 Publicação do pacote *lk-heuristic*

Uma vez finalizado o desenvolvimento de todos os *scripts* de programação, a última etapa de desenvolvimento da implementação é a conversão da estrutura de arquivos em um pacote instalável, o qual foi denominado *lk-heuristic*, e sua respectiva distribuição *online*. A Tabela 10 lista os arquivos de programação inclusos no pacote, destacando a quantidade de linhas de programação e documentação em cada arquivo, de modo a ressaltar o caráter documental do pacote, uma vez que a maior parte dos arquivos possuem mais linhas de documentação e comentários do que de código computacional.

Tabela 10 – Arquivos Python do pacote *lk-heuristic*.

arquivo	**quantidade de linhas**	
	programação	**documentação**
models/node.py	52	123
models/edge.py	17	36
models/tour.py	275	383
models/tsp.py	378	444
utils/cost_funcs.py	6	21
utils/io_funcs.py	44	47
utils/plot_funcs.py	84	41
utils/solver_funcs.py	70	60
utils/timer_funcs.py	14	15
total	940	1170

Fonte: O autor.

O pacote é distribuído com quatro pequenas instâncias do PCV no formato *.tsp*, de modo que o usuário possa executar uma rápida avaliação da ferramenta e suas funcionalidades sem a necessidade do uso de fontes de dados externas. O tutorial de utilização da biblioteca é apresentado na página principal do pacote, disponível em Castro (2022).

Em adição à heurística de Lin-Kernighan, dois métodos para resolução do PCV foram incluídos, os quais não foram citados anteriormente dada a sua simplicidade computacional: (a) a busca exaustiva, na qual todas as possibilidades de rotas são testadas e (b)

a heurística do vizinho mais próximo, na qual o ordenamento dos vértices é baseado no vértice mais próximo ao seu antecessor. O primeiro método é somente viável para instâncias reduzidas, sendo recomendado em problemas com menos de 10 vértices, enquanto o segundo método pode ser aplicado para inicialização da rota a ser reduzida pela heurística de Lin-Kernighan, aproximação que será realizada em determinada simulação de instância industrial neste trabalho.

O repositório utilizado para controle e versionamento do código-fonte é o Github, onde, adicionalmente aos arquivos de programação principais, também são disponibilizados o tutorial de uso, os testes unitários, a licença, entre outros conteúdos adicionais. Para facilitar a utilização e abrangência do pacote, o mesmo é distribuído pelo repositório oficial de bibliotecas PyPI, podendo ser instalado pelo gerenciador de pacotes PIP. Os principais dados de distribuição do pacote se encontram na Tabela 11.

Tabela 11 – Metadados do pacote *lk-heuristic*.

metadado	**valor**
nome	lk-heuristic
versão	0.0.2
resumo	Heurística de Lin-Kernighan em Python
autor	Frederico de Castro Neto
email	fred.cneto@hotmail.com
página	<https://github.com/kikocastroneto/lk_heuristic>
licença	MIT
palavras-chave	tsp,lk,lkh
tamanho	34KB

Fonte: O autor.

Todo o conteúdo textual, tanto do código-fonte como das documentações, foi escrito em inglês com o objetivo de aumentar o seu alcance e facilitar a consulta por pesquisadores do mundo todo. Além do compartilhamento do repositório em redes profissionais, como o Linkedin, o endereço do repositório foi incluído na página da Wikipedia[3] sobre a heurística de Lin-Kernighan.

O pacote *lk-heuristic* tem como base principal as instâncias acadêmicas e o respectivo padrão TSPLIB de entrada de dados. A maior parte de seu conteúdo é aplicável para as instâncias industriais porém, algumas particularidades dessas instâncias, como a leitura dos arquivos de entrada de rebites, o mapeamento de parâmetros específicos dos vértices

[3] <https://en.wikipedia.org/wiki/Lin-Kernighan_heuristic>. Acesso em 30 de julho de 2023

de rebitagem e as funções de custo customizadas para o problema industrial devem ser adicionalmente desenvolvidas. Por esse motivo, uma extensão do pacote principal foi criada a fim de receber os desenvolvimentos específicos para o caso industrial estudado. Tal pacote não foi tornado público por conter informações sigilosas da empresa patrocinadora deste trabalho. No próximo capítulo, o processo de coleta e construção das instâncias do PCV utilizadas neste trabalho, tanto para o caso acadêmico como para o caso industrial, será detalhado.

6 COLETA DE DADOS

Neste capítulo serão apresentados as motivações e detalhamentos do processo de coleta das instâncias acadêmicas do PCV e o processo de aquisição de dados de rebitagem para construção das instâncias industriais a serem utilizadas neste trabalho.

6.1 Coleta das instâncias acadêmicas

Dada a importância do PCV e o crescimento de sua popularidade ao longo dos anos, principalmente no meio acadêmico, Reinelt (1991) identificou a oportunidade de padronizar o estudo do problema e de suas respectivas instâncias com o objetivo de facilitar as análises e comparações de resultados dos diferentes trabalhos e algoritmos propostos. A fim de cumprir com este objetivo, foi desenvolvida a TSPLIB, uma biblioteca digital gratuita contendo dezenas de instâncias dos mais variados tipos de PCV, as quais foram amplamente aceitas pela comunidade científica e facilitaram as simulações e análises de resultados no meio acadêmico.

A biblioteca TSPLIB se baseia em um arquivo de texto na extensão *.tsp* composto por: (a) um cabeçalho que contém as definições básicas do problema, como sua dimensão, tipo de custo e tipo do PCV e (b) uma seção de dados, contendo as informações de cada vértice que compõem o problema. Além destes arquivos, determinadas instâncias possuem também, em um arquivo específico, a solução global do problema, sendo um importante ponto de referência a ser utilizado pelos pesquisadores que estudam tais instâncias.

A partir da coleção de instâncias disponíveis na biblioteca, cinco instâncias foram selecionadas para a validação do algoritmo heurístico a ser desenvolvido e se encontram listadas e resumidas na Tabela 12.

Tabela 12 – Instâncias acadêmicas selecionadas.

nome	**dimensão**	**custo**	**descrição**
berlin52.tsp	52	euclideano	cidades em Berlim
bier127.tsp	127	euclideano	bares localizados na Alemanha
a280.tsp	280	euclideano	pontos de furação
fl417.tsp	417	euclideano	pontos de furação em um PCB
d1291.tsp	1291	euclideano	pontos de furação em um PCB

Fonte: O autor.

As instâncias selecionadas possuem dimensão variada de modo a se validar a sensibilidade do algoritmo em termos do crescimento dimensional das instâncias. Além disso, tais dimensões selecionadas são representativas para o problema industrial. O resultado das simulações de cada instância será apresentado no Capítulo 7.

6.2 Coleta das instâncias industriais

A coleta de dados das instâncias industriais é um processo mais complexo quando comparado com a coleta das instâncias acadêmicas, uma vez que neste caso as instâncias deverão ser construídas a partir da extração de informações dos produtos aeronáuticos a serem estudados. Assim, esta seção descreve o fluxo padronizado desenvolvido para construção das instâncias industriais e as ferramentas automatizadas implementadas para extração das informações de rebitagem utilizando o programa de modelagem Catia V5 e o pós-processamento desses dados através de *scripts* na linguagem de programação Python. O objetivo deste fluxo é ser flexível e adaptável de modo que novas instâncias de produtos e famílias aeronáuticas distintas possam ser construídas e estudadas no futuro pelos próprios engenheiros da empresa em que o estudo é realizado.

O primeiro passo deste fluxo é a seleção dos produtos aeronáuticos a terem suas instâncias construídas. A seleção para este estudo foi realizada a partir de estruturas que compõem a principal aeronave comercial da empresa colaboradora, tendo sido selecionadas estruturas da região dianteira da fuselagem do avião, conforme destacado na Figura 38. Estes três produtos foram escolhidos por serem escopo do equipamento de rebitagem automatizada, possuírem quantidades de rebites em faixas distintas, além de apresentarem geometrias de peças e tipos de rebites variados, sendo também representativos para outras famílias de aviões e seus respectivos produtos aeronáuticos.

Figura 38 – Produtos aeronáuticos destacados na fuselagem do avião.

Fonte: O autor.

As principais informações a serem coletadas para cada rebite são o vetor posicional com as coordenadas espaciais de posicionamento e o código alfa-numérico que define o seu respectivo tipo. A partir deste último, é possível definir características como o

diâmetro, comprimento, geometria principal, entre outras características dos rebites. Estas informações podem ser extraídas diretamente do *software* de modelagem a partir de uma funcionalidade das bibliotecas de prendedores que extrai informações em um arquivo na extensão *.xml*. A partir deste arquivo os dados são transcritos para um arquivo separado por vírgulas (*.csv*) de modo a facilitar o manuseio e a edição do mesmo.

Além da coleta dos pontos de rebitagem a partir dos modelos 3D de cada produto aeronáutico, foi realizada a filtragem e coleta do sequenciamento atual desses rebites utilizando o programa CN em utilização no equipamento de rebitagem. Tal sequenciamento resulta na solução atual que será utilizada para comparação com as soluções encontradas pelo método heurístico, definindo assim o ganho obtido no final deste processo.

Uma vez definidos e coletados os parâmetro básicos de cada rebite, os seguintes atributos adicionais deverão ser mapeados para cada ponto de rebitagem: (a) os pré-furos contidos em determinadas peças e (b) o mapeamento de parâmetros relacionados ao *setup* da máquina rebitadora. Para que tais atributos sejam extraídos foram desenvolvidas aplicações automatizadas no *software* de modelagem utilizando o módulo de programação VBA, as quais serão detalhadas a seguir. A Tabela 13 demonstra o conjunto de todos os atributos coletados e mapeados para cada rebite e uma breve definição dos mesmos.

Tabela 13 – Propriedades do arquivo de rebites.

nome	**descrição**
part	nome do produto aeronáutico principal
id	identificador único do rebite
name	nome do rebite representado no *software* de modelagem
type	código alfa-numérico representando o tipo do rebite
sealant	booleano representando a necessidade de selagem do rebite
stack members	vetor contendo o nome de cada peça rebitada
location	vetor contendo a posição cartesiana do ponto de rebitagem
resync	booleano representando a operação de alinhamento
mac_drill	código alfa-numérico representando a broca de furação
mac_injector	código alfa-numérico representando o injetor do rebite
mac_upper_button	código alfa-numérico representando a pinça de inserção do rebite
gemcor_setup	valor numérico do *setup* explícito para rebitadoras Gemcor

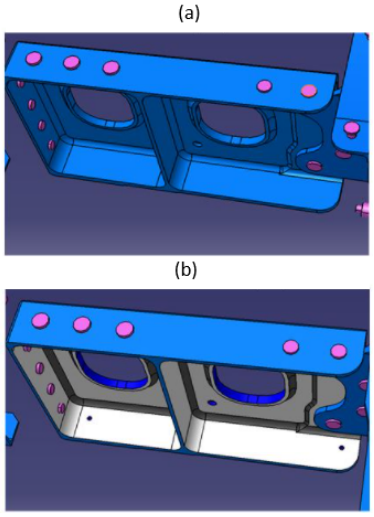
Fonte: O autor.

Para que a identificação de pré-furos em peças aeronáuticas seja realizada é necessário primeiramente executar a substituição de peças em condição de projeto por peças

em condição de manufatura. As peças de manufatura possuem condições específicas de montagem, da qual se destaca a existência de pré-furos utilizados para fixação temporária dessas peças, conforme exemplificado na Figura 39, na qual dois pré-furos podem ser identificados na peça de manufatura. Estes pré-furos estrategicamente também servem como alvo para correção do posicionamento da rebitagem a ser executada pelo equipamento automatizado, atividade denominada *resync*.

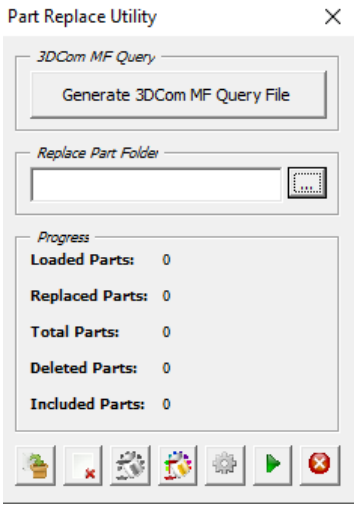
O processo de substituição de arquivos 3D envolve tarefas geométricas como espelhamento e definições de eixos de simetria complexos, os quais tornam desafiador o desenvolvimento de uma aplicação automatizada. A ferramenta resultante, demonstrada na Figura 40, implementa todos os requisitos necessários para a substituição das peças e, dada a aplicabilidade na rotina de trabalho, passou a ser utilizada por diversas equipes de técnicos e engenheiros de manufatura da empresa patrocinadora da pesquisa.

Figura 39 – Exemplo de peças: (a) em condição de projeto e (b) em condição de manufatura.



Fonte: O autor.

Figura 40 – Aplicação desenvolvida para substituição de peças de manufatura.



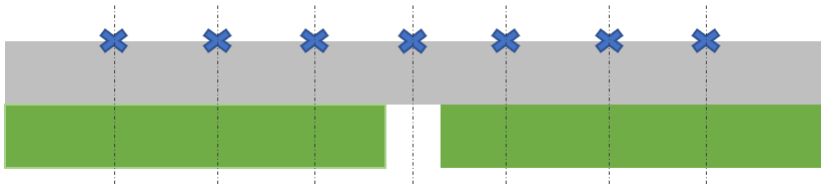
Fonte: O autor.

A partir de um modelo completo do produto contendo todas as suas respectivas peças em condição de manufatura, a coleta dos pontos de rebitagem que possuem pré-furos pode ser implementada a partir de uma nova ferramenta automatizada. Esta aplicação identifica pré-furos a partir da construção de linhas que seguem o vetor de rebitagem e que interseccionam as peças a serem rebitadas, conforme ilustrado na Figura 41.

Neste exemplo, uma vista em corte demonstra uma seção com sete pontos de rebitagem marcados em azul que possuem seus respectivos vetores direcionais projetados pela linha pontilhada, a qual intersecciona duas peças a serem unidas. Para os seis casos laterais a distância mínima entre a linha e cada uma das peças será zero, porém para o caso central, no qual existe um pré-furo na peça inferior, a distância mínima entre a linha

e a peça será próxima do raio do pré-furo, de tal forma a identificar que este ponto de rebitagem possui o pré-furo em uma das peças e, consequentemente, requer a atividade de *resync* pelo equipamento automatizado.

Figura 41 – Esquema de conexão entre rebites e peças aeronáuticas.



Fonte: O autor.

O mapeamento de atributos de *setup* do equipamento é uma etapa mais simples uma vez que é realizado através da conexão direta entre o tipo de rebite e o conjunto de equipamentos auxiliares necessários para execução da rebitagem, mais especificamente: (a) a broca de furação, (b) o alimentador de rebites e (c) a pinça de inserção dos rebites nos furos. Para realizar este mapeamento foi desenvolvido um programa em linguagem Python que correlaciona o tipo de rebite com o conjunto de ferramentas pré-definidas em um catálogo externo.

O atributo denominado *gemcor_setup* foi coletado a partir do *setup* definido para cada rebite no programa CN em operação pelas rebitadoras do fabricante Gemcor. Esta coleta foi realizada utilizando a solução atual uma vez que existe um grande número de combinações possíveis dos subsistemas neste equipamento, o que geraria uma complexidade adicional na resolução de cada instância, a qual deveria ser resolvida para cada combinação separadamente. Desta forma, a resolução é simplificada mantendo apenas um único *setup* possível, semelhante ao *setup* atual em operação, o que acelera a resolução do problema e também facilita a comparação entre os resultados encontrados.

A maior parte dos atributos exibidos na Tabela 13 estão diretamente correlacionados com a função de custo temporal do problema de rebitagem, conforme Equação 3.8. O atributo posicional do ponto de rebitagem (*location*), está correlacionado com o tempo de movimentação da máquina (t_{mov}), enquanto os atributos booleanos de selagem (*sealant*) e correção posicional (*resync*) estão correlacionados, respectivamente, com o tempo de selagem ($t_{selagem}$) e tempo de correção (t_{resync}) da função de custo. A combinação dos valores para os atributos de *setup* definem tanto o tempo de *setup* do equipamento (t_{setup}) como também os tempos de furação ($t_{furação}$) e instalação ($t_{instalação}$) dos rebites.

Uma vez definidas as instâncias a serem estudadas e coletados os principais dados de rebitagem para cada instância industrial, o estudo de sequenciamento de rebites será realizado utilizando tais dados como entrada do sistema heurístico implementado. A partir

dos atributos de rebitagem, as funções de custo industrial definidas na Seção 3.4 foram utilizadas a fim de comparar estratégias de sequenciamento de rebites baseadas em combinações e pesos desses atributos. É importante destacar que o fluxo padronizado desenvolvido nesta seção poderá ser reaproveitado em novas instâncias a serem estudadas, uma vez que ferramentas automatizadas para a coleta e processamento de tais informações foram desenvolvidas e disponibilizadas à empresa patrocinadora. As três instâncias industriais construídas para serem estudadas neste trabalho se encontram listadas na Tabela 14.

Tabela 14 – Instâncias industriais construídas.

nome	**dimensão**	**descrição**
i1360.csv	1630	fuselagem dianteira lateral do avião
i2751.csv	2751	fuselagem da porta do avião
i5435.csv	5435	fuselagem dianteira inferior do avião

Fonte: O autor.

No próximo capítulo serão apresentados os resultados computacionais obtidos utilizando a implementação da heurística de Lin-Kernighan detalhada no capítulo Capítulo 5. O sistema computacional foi utilizado para resolução das instâncias acadêmicas do PCV e instâncias industriais de rebitagem, conforme coleta detalhada neste capítulo, com o objetivo de comprovar a eficiência da implementação desenvolvida neste trabalho.

7 RESULTADOS NUMÉRICOS

Neste capítulo serão apresentados os resultados numéricos das simulações realizadas com as instâncias acadêmicas e industriais em conjunto com uma análise crítica de tais resultados em termos da performance computacional e qualidade das soluções obtidas.

7.1 Etapa 1: Instâncias acadêmicas

Para realizar as simulações das instâncias acadêmicas foi utilizada a biblioteca de resolução desenvolvida neste trabalho para problemas do tipo TSPLIB. O *hardware* utilizado foi um notebook com processador Intel Core i7 2.70GHz, 32GB de memória RAM e sistema operacional Windows 10.

Conforme detalhado nas etapas de desenvolvimento da heurística, existem parâmetros a serem definidos pelo usuário antes de executar o processo heurístico. De modo a avaliar a influência de cada parâmetro nos resultados computacionais obtidos, foi proposta a combinação de determinados parâmetros no seguinte plano de testes:

- Camadas: Para a definição do número de camadas de busca e a respectiva profundidade de cada camada, foram utilizados os valores propostos por Kernighan e Lin (1973) e os melhores valores definidos no estudo exploratório de Applegate, Cook e Rohe (2003). Ambos trabalhos exploram a influência da quantidade e profundidade das camadas na eficiência computacional da heurística, direcionando os valores a serem utilizados nas instâncias industriais;
- Redução: O parâmetro de redução foi utilizado com base no valor proposto por Kernighan e Lin (1973) de 4 ciclos iniciais de pré-redução. Por se tratar de um refinamento opcional que tende a reduzir a qualidade da solução obtida, simulações sem este parâmetro foram realizadas de modo a avaliar o benefício obtido com sua utilização. Na versão simplificada do algoritmo (lk2) esse parâmetro não é utilizado;
- Ciclos: Para definição do número de ciclos de busca para cada teste, simulações de validação foram realizadas em cada instância. A partir destas simulações, foi definido um valor fixo de 100 ciclos de busca por teste, valor limitado pela instância de maior dimensão;
- Algoritmos: Os algoritmos utilizados foram a heurística completa (lk1) e o modelo simplificado (lk2). Ambos os algoritmos foram utilizados com objetivo de avaliar a hipótese de que são igualmente eficientes, observando as vantagens e desvantagens de cada um a partir dos resultados obtidos;

- Inicialização: A rota inicial adotada para todos os testes é obtida a partir do embaralhamento dos vértices, de modo que cada ciclo de busca seja reiniciado com uma nova rota randômica a ser reduzida. Tal método de inicialização é o mais adequado para validação inicial da heurística pois evita o viés computacional que outros métodos apresentam, como a heurística do vizinho mais próximo.

A partir dos parâmetros de exploração definidos anteriormente, foi formatado um conjunto de 9 testes a serem realizados com cada instância acadêmica, por meio da combinação desses parâmetros, conforme detalhado na Tabela 15.

Tabela 15 – Conjunto de testes para as instâncias acadêmicas.

teste	**inicialização**	**ciclos**	**camadas**	**redução**	**algoritmo**
1	*randômica*	100	(5,5)[a]	4[a]	*lk1*
2	*randômica*	100	(4,3,3,2)[b]	4	*lk1*
3	*randômica*	100	(12,12)[b]	4	*lk1*
4	*randômica*	100	(5,5)	0	*lk1*
5	*randômica*	100	(4,3,3,2)	0	*lk1*
6	*randômica*	100	(12,12)	0	*lk1*
7	*randômica*	100	(5,5)	0	*lk2*
8	*randômica*	100	(4,3,3,2)	0	*lk2*
9	*randômica*	100	(12,12)	0	*lk2*

[a] valores de referência em Kernighan e Lin (1973).
[b] valores de referência em Applegate, Cook e Rohe (2003).

Fonte: O autor.

Os principais resultados coletados para o conjunto de simulações de cada teste são: (a) o melhor custo obtido, correspondendo ao menor valor do custo encontrado no conjunto de simulações, (b) o custo médio, definido pela média do custo de todas as simulações e (c) o tempo médio, definido pela média do tempo de todas as simulações realizadas. A coleta do tempo médio é importante para verificação da sensibilidade computacional da heurística em instâncias de variadas dimensões, funcionando como um valor aproximado para avaliação da complexidade computacional do algoritmo. O uso de um elevado número de ciclos do método heurístico garante um valor estável para estimativa do tempo de simulação em cada instância e potencializa a probabilidade de encontrar soluções com custos reduzidos e próximos à solução global do problema.

Uma vantagem na utilização das instâncias acadêmicas é que as mesmas possuem soluções ótimas globais conhecidas, de tal forma que os valores de custo obtidos nas simulações podem ser comparados a esses ótimos, definindo assim a precisão do teste realizado, dada pela razão entre o custo obtido (melhor ou médio) e o custo ótimo global.

7.1.1 Instância *berlin52*

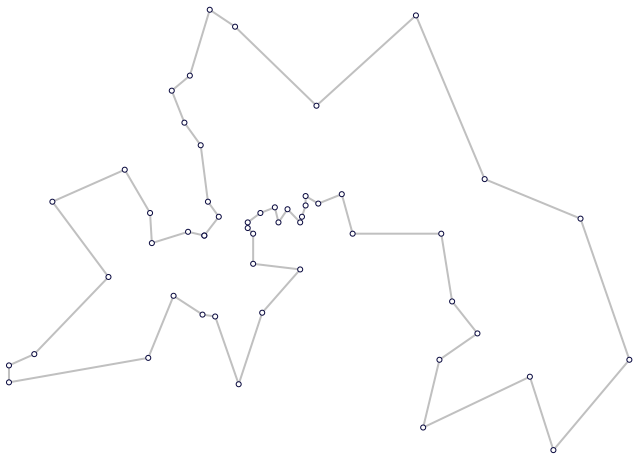
A primeira instância cujos testes serão detalhados é denominada *berlin52*. Esta instância possui apenas 52 pontos, que representam diferentes localizações em Berlim, capital da Alemanha. Os resultados das simulações estão demonstrados na Tabela 16 e a melhor rota obtida é ilustrada na Figura 42.

Tabela 16 – Resultados para a instância *berlin52*.

		custo			**precisão(%)**	
teste	***t*(s)**	*melhor*	*médio*	*global*	*melhor*	*médio*
1	0.02	7544.3	7767.7	7544.3	100	97.1
2	0.03	7544.3	7899.4	7544.3	100	95.5
3	0.13	7544.3	7647.6	7544.3	100	98.6
4	0.03	7544.3	7813.3	7544.3	100	96.5
5	0.02	7544.3	7920.9	7544.3	100	95.2
6	0.09	7544.3	7662.2	7544.3	100	98.4
7	0.04	7544.3	7705.3	7544.3	100	97.9
8	0.04	7544.3	7821.8	7544.3	100	96.4
9	0.09	7544.3	7675.5	7544.3	100	98.2

Fonte: O autor.

Figura 42 – Melhor rota obtida para a instância *berlin52*.



Fonte: O autor.

Em termos de performance computacional, é verificado que todos os testes foram resolvidos com extrema velocidade, com tempo médio de simulação na ordem de centésimos de segundo. Naturalmente, o conjunto de testes {3, 6, 9} apresentaram menor performance uma vez que possuem maior exploração combinatória de vértices e arestas, definida pelo

número e profundidade de camadas exploratórias. Apesar disso, estes mesmos testes são os que apresentaram maior precisão média, resultado esperado dada a maior exploração das configurações de camadas. Destaca-se ainda nesta instância a solução global encontrada em todos os testes executados e a grande precisão, acima de um valor médio de 95%, observada nos mesmos.

Apesar de produtos aeronáuticos possuírem muito mais do que 50 rebites, as peças que compõem esses produtos costumam possuir um número menor do que esta quantidade. Desta forma, uma estratégia que pode ser definida a partir da simulação da instância *berlin52* é a da resolução agrupada do problema industrial completo, com objetivo de otimizar a performance computacional mantendo resultados de custo satisfatórios.

7.1.2 Instância *bier127*

A segunda instância cujos resultados serão apresentados é denominada *bier127*. Assim como a primeira instância (*berlin52*), possui referência à Alemanha, porém nesta são listados 127 locais chamados de *Biergaerten* (um tipo de bar a céu aberto) na cidade de Augsburgo. Os resultados são demonstrados na Tabela 17 e a melhor rota obtida pelas simulações se encontra na Figura 43.

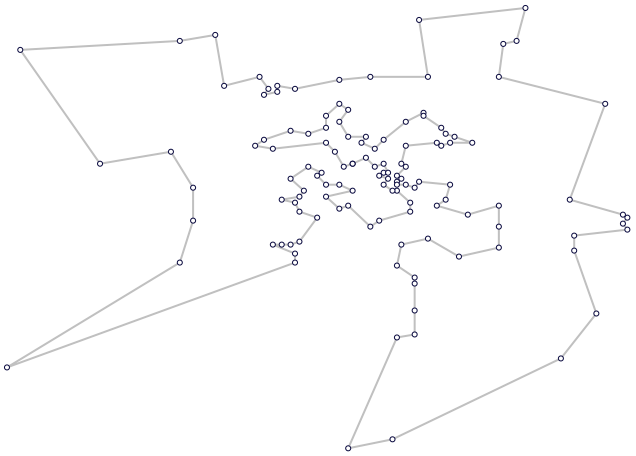
Tabela 17 – Resultados para a instância *bier127*.

		custo			**precisão(%)**	
teste	***t*(s)**	*melhor*	*médio*	*global*	*melhor*	*médio*
1	0.08	118904.3	122626.9	118293.5	99.4	96.4
2	0.06	119489.6	127482.6	118293.5	99.0	92.7
3	0.17	118293.5	120221.2	118293.5	100	98.4
4	0.08	118658.1	122870.9	118293.5	99.6	96.2
5	0.07	119082.1	126735.0	118293.5	99.3	93.3
6	0.25	118293.5	120298.4	118293.5	100	98.3
7	0.12	118513.2	120894.4	118293.5	99.8	97.8
8	0.11	119497.6	124399.5	118293.5	98.9	95.0
9	0.30	118629.0	120020.2	118293.5	99.7	98.5

Fonte: O autor.

A partir da coluna com o tempo médio de execução de cada teste, é possível notar que a velocidade para resolução do problema continua elevada, com valor máximo de resolução de aproximadamente 0.3 segundo. Se comparado ao valor máximo do problema anterior (*berlin52*), o tempo de resolução é aproximadamente 3 vezes maior, porém a dimensão do problema não foi triplicada. Esse comportamento demonstra o crescimento não-linear do tempo de resolução, situação esperada no PCV e a ser melhor analisada na Subseção 7.1.6.

Figura 43 – Melhor rota obtida para a instância *bier127*.



Fonte: O autor.

Comparando o conjunto de testes com 4 camadas de exploração {2, 5, 8} e com 2 camadas {1, 4, 7}, é verificado que o primeiro conjunto apresentou maior performance computacional, mesmo tendo maior quantidade total de vértices explorados pela configuração de suas camadas. Esse fato demonstra a importância do parâmetro de camadas e sua flexibilidade baseada no problema a ser resolvido. Apesar disso, os testes 3 e 6, que apresentaram os melhores resultados de custo e encontraram a solução ótima global do problema, foram os que utilizaram maior exploração de vértices.

Os resultados encontrados para *bier127* podem representar a solução do problema industrial de rebitagem agrupado por peça, assim como visto em *berlin52*, porém neste caso considerando peças mais robustas que contenham em torno de 100 rebites. Utilizando os valores de velocidade das simulações em *berlin52* e *bier127*, estima-se que o sequenciamento de rebites agrupados por peça sejam resolvidos entre 10 segundos para produtos aeronáuticos pequenos a 1 minuto para produtos grandes.

7.1.3 Instância *a280*

A terceira instância acadêmica é denominada *a280*, a qual representa 280 pontos de furação em um produto não especificado, onde cada coordenada espacial representa a localização do furo. É a primeira instância acadêmica baseada em um problema industrial. Os resultados das simulações com esta instância estão apresentados na Tabela 18 e a Figura 44 apresenta a melhor rota obtida nas simulações executadas.

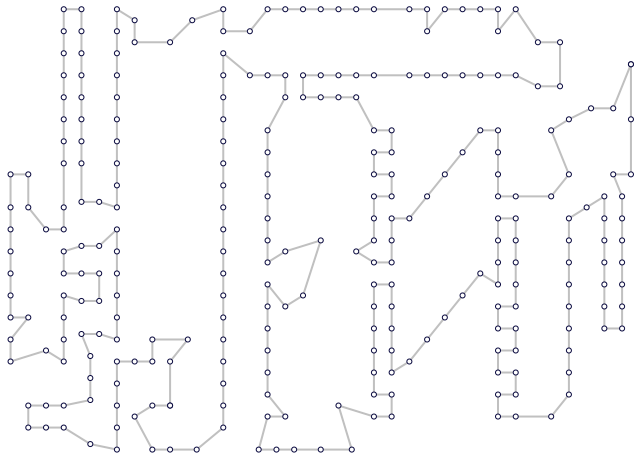
A performance computacional da heurística apresenta valores de tempo médio reduzidos assim como em *berlin52* e *bier127*, mesmo se tratando de uma instância com dimensão consideravelmente maior que as instâncias citadas. É possível notar uma leve

Tabela 18 – Resultados para a instância *a280*.

		custo			**precisão(%)**	
teste	***t*(s)**	*melhor*	*médio*	*global*	*melhor*	*médio*
1	0.15	2619.9	2833.2	2586.7	98.7	91.3
2	0.17	2655.0	2920.1	2586.7	97.4	88.5
3	0.31	2594.4	2658.0	2586.7	99.7	97.3
4	0.20	2599.2	2823.0	2586.7	99.5	91.6
5	0.14	2661.3	2928.3	2586.7	97.2	88.3
6	0.36	2589.0	2662.1	2586.7	99.9	97.1
7	0.48	2589.0	2681.5	2586.7	99.9	96.4
8	0.24	2589.0	2718.9	2586.7	99.9	95.1
9	0.54	2586.7	2638.1	2586.7	100	98.0

Fonte: O autor.

Figura 44 – Melhor rota obtida para a instância *a280*.



Fonte: O autor.

queda na precisão de algumas simulações, como no conjunto de testes {2, 5}, os quais apresentam precisão média inferior a 90%, porém é importante ressaltar que estes testes apresentaram tempo médio de simulação reduzidos, sendo, por exemplo, aproximadamente 4 vezes mais rápido que o teste 9. Este último, por sua vez, foi o teste que apresentou o melhor resultado em termos do custo obtido, encontrando a solução ótima global do problema. Além disso, o teste 9 utiliza o algoritmo simplificado, de tal forma a concluir que o mesmo é eficiente assim como o algoritmo completo, porém tende a levar mais tempo para convergência, conforme pode ser visto nos resultados do conjunto de simulações {7, 8, 9}, as quais são mais lentas quando comparadas ao conjunto de testes {4, 5, 6}, nos quais a única diferença é o algoritmo utilizado.

Um importante comparativo a ser traçado com o resultado alcançado para esta instância se refere aos resultados obtidos por Mahéo (2017). O algoritmo simplificado lk1, implementado neste trabalho, é baseado na heurística simplificada de Helsgaun (2000), a qual foi implementada pelo autor. Entretanto, existe uma diferença no algoritmo lk1, especificamente na estruturação dos objetos que compõem o problema e que resultam em uma grande diferença de performance dos algoritmos. Conforme demonstrado na Tabela 19, o autor desenvolveu e comparou diversos algoritmos para resolução do PCV, tanto em termos do custo obtido mas também em relação à performance computacional de cada algoritmo, utilizando a instância acadêmica *a280*.

Tabela 19 – Resultados de Mahéo (2017) para a instância *a280*.

algoritmo	**custo médio**	***gap*(%)**	***t*(s)**
Greedy	3148.11	22.07	0.02
*2-Opt**	3072.24	19.13	43.81
2-Opt	2874.38	11.45	26.84
3-Opt	2794.51	8.36	2343.79
LK	2642.55	2.46	210.05

Fonte: Adaptado de Mahéo (2017).

O resultado encontrado pelo autor para a heurística de Lin-Kernighan (LK) apresenta tempo médio de resolução de 210 segundos, aproximadamente 400 vezes maior que o tempo equivalente encontrado no teste 7 neste trabalho, demonstrado na Tabela 18. Apesar da diferença entre os *hardwares* utilizados para realizar as simulações, o principal motivo para tal diferença está no algoritmo que reorganiza a estrutura da rota ao realizar uma operação de troca de arestas, o qual possui complexidade computacional $n!$ na implementação do autor, e complexidade n na implementação deste trabalho. Como a heurística é fortemente baseada nessas operações de remoção e adição de arestas, o tempo total para execução acaba sendo diretamente afetado pela complexidade dessa operação. A precisão do modelo do autor apresenta resultados similares aos encontrados neste trabalho, com valor próximo de 97%, uma vez que a estratégia da heurística implementada em ambos algoritmos é similar.

Apesar de ser um problema industrial mais simples, a instância *a280*, e sua respectiva dimensão de 280 pontos, pode começar a representar pequenos produtos aeronáuticos, bem como grupos de rebites em pequenos *setups* nas instâncias industriais de maior dimensão. O tempo para resolução foi duplicado em relação à instância *bier127*, porém o valor absoluto máximo encontrado de aproximadamente 0.54 segundo ainda é extremamente baixo, de modo que esta instância representa uma potencial estratégia de agrupamento de rebites para as instâncias industriais.

7.1.4 Instância *fl417*

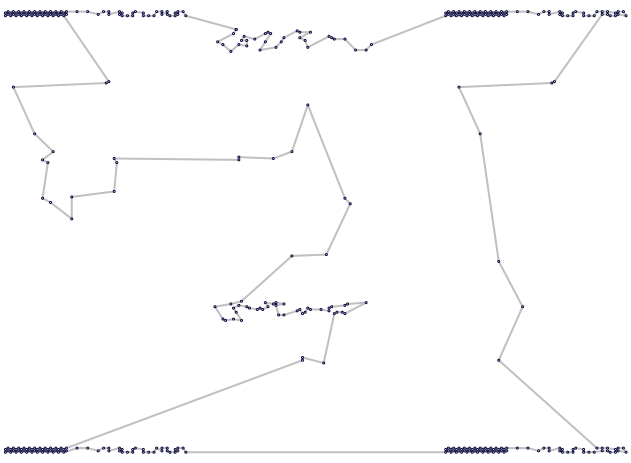
A quarta instância acadêmica, denominada *fl417*, trata também de um problema industrial de furação, porém aplicado em circuitos impressos (PCB), com 417 vértices. Os resultados estão apresentados na Tabela 20 e a melhor rota encontrada é ilustrada na Figura 45.

Tabela 20 – Resultados para a instância *fl417*.

		custo			**precisão(%)**	
teste	***t*(s)**	*melhor*	*médio*	*global*	*melhor*	*médio*
1	0.48	13464.5	16982.2	11914.7	88.4	70.1
2	0.28	15015.2	19470.6	11914.7	79.3	61.2
3	0.83	12079.1	13780.7	11914.7	98.6	86.4
4	0.36	13589.6	17093.2	11914.7	87.6	69.7
5	0.40	15386.3	19034.9	11914.7	77.4	62.6
6	1.14	12179.8	13756.6	11914.7	97.8	86.6
7	0.87	12449.4	15879.1	11914.7	95.7	75.0
8	0.69	13840.5	16980.9	11914.7	86.1	70.1
9	0.79	11967.8	13394.0	11914.7	99.5	88.9

Fonte: O autor.

Figura 45 – Melhor rota obtida para a instância *fl417*.



Fonte: O autor.

O crescimento dimensional do problema a partir desta instância começa a influenciar a performance computacional e os resultados obtidos de maneira um pouco mais agressiva. Pela primeira vez, o tempo médio de execução do algoritmo apresenta valor máximo maior do que 1 segundo, sendo aproximadamente 2 vezes maior que o tempo máximo em *a280*.

Em relação aos resultados de custo médio obtidos, é possível verificar uma queda considerável, de forma que os mesmos não ultrapassaram 90% de precisão. Os resultados médios obtidos nos testes com camadas de exploração mais enxutas apresentaram precisão menor que 75%. Apesar da queda de performance e da qualidade dos resultados obtidos, o melhor resultado obtido atingiu 99.5% de precisão e, assim como visto na instância *a280*, foi alcançado com uso do algoritmo simplificado, provando novamente a constatação feita por Helsgaun (2000) a respeito de sua eficiência.

A partir do grafo gerado com a melhor solução encontrada, é notado que esta instância apresenta característica de agrupamento espacial dos vértices, concentrados nos 4 cantos superiores e inferiores do grafo. A partir da análise desta propriedade, é possível concluir que os testes executados com maior profundidade de exploração tendem a apresentar melhores resultados, uma vez que estes acabam por conectar os grupos de vértices de maneira mais eficiente, porém com alta requisição computacional, conforme observado pelo tempo médio dessas simulações.

Outra propriedade importante de problemas com vértices agrupados é que costumam ser mais difíceis de serem resolvidos, uma vez que quaisquer alterações nos vértices de início e fim da região agrupada requer o recálculo da rota definida pelos vértices do grupo, processo que pode também influenciar grupos de vértices vizinhos de tal forma que toda a rota do problema seja modificada. Esse tipo de propriedade deve ser considerada ao resolver o problema industrial de rebitagem, uma vez que ao se agrupar rebites por peça ou *setup* do equipamento, tal condição de agrupamento será gerada, possivelmente dificultando a solução do problema. Para reduzir a complexidade nesta situação, o recomendável é resolver cada grupo de vértices separadamente, de tal forma que a solução obtida possivelmente não será o ótimo global, porém manterá um resultado satisfatório em um tempo computacional reduzido.

7.1.5 Instância *d1291*

A última instância selecionada para os testes acadêmicos é denominada *d1291*, possuindo dimensão de 1291 vértices, representando pontos de furação em circuitos impressos, aplicação industrial equivalente à instância *fl417*. Os resultados são apresentados na Tabela 21 e a melhor rota encontrada para esta instância é ilustrada na Figura 46.

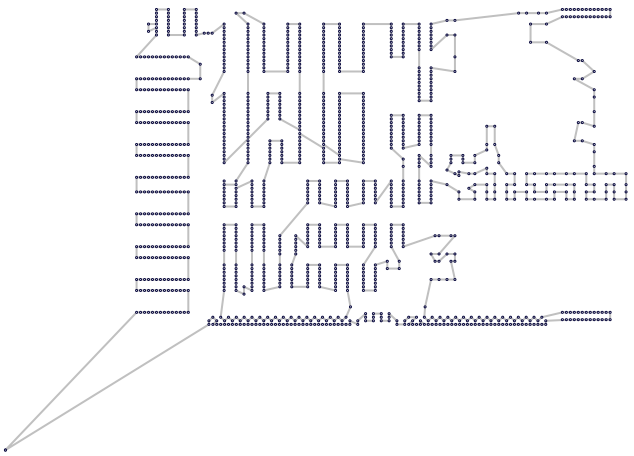
O tempo médio para execução do algoritmo heurístico em todos os testes apresentam um aumento considerável quando comparados com os valores encontrados na instância *fl417*, uma vez que a dimensão do problema foi triplicada e o seu valor absoluto ultrapassa 1000 vértices. O menor tempo de resolução foi apresentado pelo teste 4, com 0.81 segundo, valor 2 vezes maior que o tempo necessário pelo mesmo teste na instância *fl417*, enquanto que o teste 9, responsável pelo melhor custo obtido, foi realizado em aproximadamente 10 segundos.

Tabela 21 – Resultados para a instância *d1291*.

		custo			**precisão(%)**	
teste	***t*(s)**	*melhor*	*médio*	*global*	*melhor*	*médio*
1	2.48	83728.6	88608.9	51165.4	61.1	57.7
2	2.22	87032.1	95111.8	51165.4	58.7	53.7
3	7.02	54965.5	57817.3	51165.4	93.0	88.4
4	0.81	85932.7	90610.4	51165.4	59.5	56.4
5	1.94	89390.1	95353.3	51165.4	57.2	53.6
6	6.09	56589.6	59008.9	51165.4	90.4	86.7
7	3.85	79895.5	84971.8	51165.4	64.0	60.2
8	3.56	84978.2	89583.3	51165.4	60.2	57.1
9	9.96	52196.4	54950.7	51165.4	98.0	93.1

Fonte: O autor.

Figura 46 – Melhor rota obtida para a instância *d1291*.



Fonte: O autor.

Os melhores resultados são encontrados a partir do conjunto de testes de maior profundidade {3, 9}, obtendo precisão média maior de 90%. Apesar da instância *fl417* possuir menor dimensão, os resultados obtidos nesta possuem menor precisão quando comparados com *d1291* uma vez que a distribuição agrupada dos vértices no problema afetam a solução obtida pela heurística. A instância *d1291* também apresenta agrupamento de vértices, porém estes estão mais próximos entre si, gerando assim melhores resultados.

O grafo ilustrado com a melhor rota obtida demonstra uma grande similaridade com o problema industrial de sequenciamento de rebites, uma vez que apresentam geometria linear de tal forma que a melhor trajetória tende a ser uma linha reta, conforme visto

entre os cantos inferior e superior esquerdo do grafo. Cada linha horizontal definida nesta região do grafo pode ser interpretada como uma peça aeronáutica do problema industrial. Além disso, as regiões agrupadas são próximas entre si assim como as peças que compõem o produto aeronáutico tendem a ser dispostas desta maneira.

7.1.6 Análise dos resultados para as instâncias acadêmicas

A partir dos resultados apresentados nas seções específicas para cada instância acadêmica, a Tabela 22 destaca o melhor resultado obtido para cada instância e a respectiva configuração do teste e precisão alcançada. Como determinadas instâncias apresentaram os mesmos resultados, a precisão média foi utilizada como critério de desempate.

Tabela 22 – Melhor resultado para cada instância acadêmica.

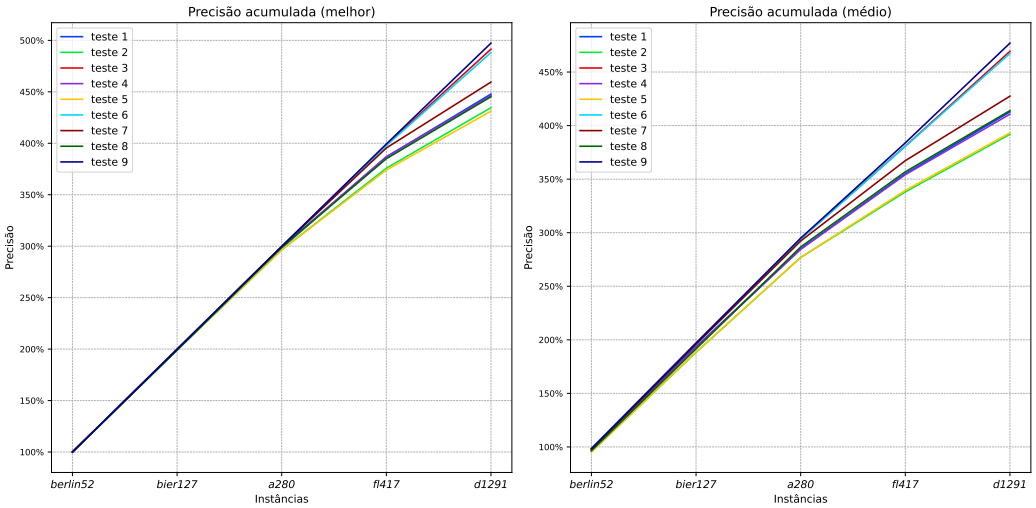
instância	**teste**	**camadas**	**redução**	**algoritmo**	**precisão(%)**
berlin52	3	(12,12)	4	*lk1*	100
bier127	3	(12,12)	4	*lk1*	100
a280	9	(12,12)	0	*lk2*	100
fl417	9	(12,12)	0	*lk2*	99.5
d1291	9	(12,12)	0	*lk2*	98.0

Fonte: O autor.

Analisando apenas a configuração das camadas é possível concluir que a profundidade de cada camada está diretamente relacionada com a maior precisão nos resultados obtidos. A configuração mais profunda, com 12 vértices explorados em 2 camadas, obteve o melhor resultado em todas as instâncias acadêmicas avaliadas. Em contrapartida, definir tal configuração não é uma tarefa simples, pois depende da quantidade e distribuição dos vértices na instância a ser resolvida. Em instâncias como *fl417* e *d1291*, as quais apresentam agrupamento de vértices, o parâmetro de camada pode ser definido a partir da distância entre esses grupos e a quantidade de vértices por grupo, de modo que a profundidade de cada camada seja grande o suficiente para calcular a melhor rota entre esses grupos e não apenas sua trajetória interna.

A Figura 47 apresenta os gráficos com a precisão acumulada por teste executado para cada instância acadêmica. A precisão dos melhores resultados até a instância *a280* se mantém aproximadamente constante e, dado o crescimento da dimensão nas instâncias *fl417* e *d1291*, a diferença na precisão é perceptível e observada através da dispersão das curvas de cada teste. Para os melhores resultados médios, essa diferença de precisão é notada a partir da instância *a280*, porém neste caso com uma dispersão mais aguda. Como resultado geral, o conjunto de testes com 12 vértices de profundidade de busca {3, 6, 9} apresentam as maiores precisões obtidas, fato observado anteriormente na Tabela 22.

Figura 47 – Precisão acumulada para os resultados médio e melhor de cada instância.



Fonte: O autor.

A partir dos resultados apresentados pelo teste 9, principalmente nas instâncias *a280*, *fl417* e *d1291*, é possível concluir que o algoritmo simplificado (lk2) é tão eficiente quanto o algoritmo completo (lk1) do ponto de vista dos custos obtidos. Conforme visto nos gráficos de precisão acumulada, a curva do teste 9 demonstra que esta configuração apresentou os melhores resultados gerais e médios. Entretanto, o algoritmo completo apresenta maior performance computacional se comparado ao algoritmo simplificado, uma vez que este não possui refinamentos de busca que otimizem o tempo necessário para convergência da solução.

Segundo Kernighan e Lin (1973), a complexidade computacional da heurística é aproximadamente $n^{2.2}$, informação que os autores consideram relevante, uma vez que o valor está entre a complexidade das heurísticas 2-Opt, proposta por Croes (1958), e 3-Opt, proposta por Lin (1965), as quais apresentam complexidade n^2 e n^3 respectivamente.

Para aproximar o valor da complexidade computacional, a Tabela 23 apresenta o crescimento dos tempos de resolução de cada instância no conjunto de testes {1, 2, 3} de modo a avaliar a proporção de crescimento temporal e compara-lá com a proporção do crescimento dimensional de cada instância. Os valores de crescimento são dados pela razão entre dois valores consecutivos de cada linha da tabela, as quais são ordenadas pelo crescimento dimensional de cada instância.

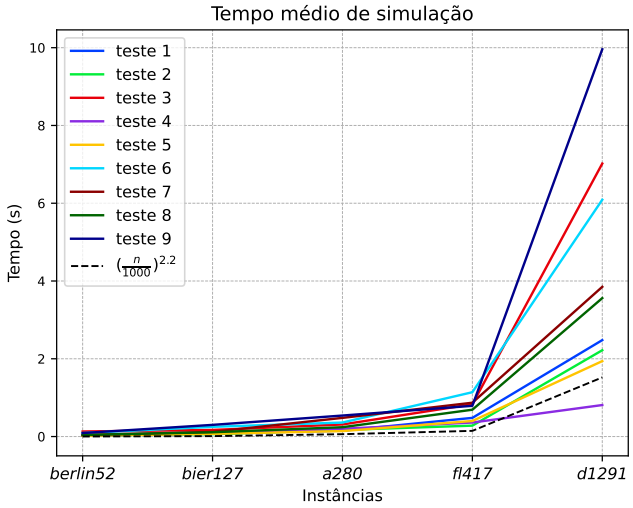
Além dos dados tabulares, outra maneira de avaliar a complexidade computacional da heurística é disposta no gráfico da Figura 48, no qual o tempo médio de simulação para cada instância é plotado a partir da instância de menor dimensão até a de maior dimensão e uma curva aproximada com a complexidade $n^{2.2}$ é adicionada para comparação.

Tabela 23 – Taxa de crescimento dimensional e temporal para resolução das instâncias acadêmicas.

		t(s)			taxa de crescimento			
instância	n	t_1	t_2	t_3	$n^{2.2}$	t_1	t_2	t_3
berlin52	52	0.02	0.03	0.13	-	-	-	-
bier127	127	0.08	0.06	0.17	7.13	4.0	2.0	1.30
a280	280	0.15	0.17	0.31	5.69	1.87	2.83	1.82
fl417	417	0.48	0.28	0.83	2.40	3.2	1.64	2.67
d1291	1291	2.48	2.22	7.02	12.0	5.16	7.92	8.45

Fonte: O autor.

Figura 48 – Tempo médio de simulação para cada instância.



Fonte: O autor.

Apesar dos valores tabulados dos testes {1, 2, 3} não serem exatamente proporcionais e variarem dependendo do teste aplicado, existe uma relação de proporcionalidade exponencial, visto principalmente entre as instâncias *fl417* e *d1291*, na última linha da tabela, uma vez que, considerada a grande dimensão das mesmas, tal efeito exponencial da complexidade da heurística se torna mais evidente.

No caso gráfico, é possível notar que as três instâncias de menor dimensão são pouco sensíveis ao crescimento dimensional. A diferença entre o tempo de execução da instância *berlin52* e *a280* é quase imperceptível, apesar do crescimento dimensional ser próximo de 400%. A partir da instância *fl417*, a dispersão das curvas de cada teste demonstra a complexidade polinomial do algoritmo, de maneira mais aguda nos testes com maior profundidade de camadas, uma vez que mais iterações de busca são realizadas.

Tendo sido expostos os resultados das instâncias acadêmicas e verificado os principais parâmetros que afetam o resultado do processo computacional da heurística de Lin-Kernighan, a próxima seção detalha o conjunto de testes realizados utilizando as instâncias industriais e os respectivos resultados encontrados para o problema de sequenciamento de rebites.

7.2 Etapa 2: Instâncias industriais

As simulações com as instâncias industriais foram realizadas a partir da biblioteca de resolução desenvolvida para problemas customizados de rebitagem. O *hardware* utilizado foi um notebook com processador Intel Core i7 2.70GHz, 32GB de memória RAM e sistema operacional Windows 10, o mesmo equipamento utilizado para as simulações dos testes acadêmicos.

As funções de custo das arestas utilizadas para as simulações industriais são as funções euclideana, agrupada e temporal, conforme modelagem do problema de sequenciamento de rebites apresentada na Seção 3.4. A Figura 49 ilustra os grafos tridimensionais contendo os vértices das instâncias industriais. Os parâmetros de execução da heurística de Lin-Kernighan para os testes desta seção foram adotados utilizando como referência a validação e parâmetros das instâncias acadêmicas apresentada na Seção 7.1. Esses valores se encontram resumidos na Tabela 24 e o detalhamento de cada parâmetro é explicado a seguir:

- Camadas: Todos os testes industriais foram realizados com 2 camadas exploratórias e 25 vértices cada uma, com objetivo de atender a condição de agrupamento de vértices nas peças aeronáuticas que definem essas instâncias. Este valor poderia ser refinado para cada instância a fim de reduzir o tempo das simulações, contudo foi utilizado um valor grande o suficiente para atender a definição de uma vizinhança ampla para cada vértice explorado;

- Redução: O parâmetro de redução não foi utilizado nos testes industriais, uma vez que os resultados apresentados nas instâncias acadêmicas não foram satisfatórios. O uso deste parâmetro apresentou piora da performance em determinados casos, além de, naturalmente, convergir para uma solução de maior custo;

- Ciclos: A partir da execução de simulações de validação com as instâncias industriais, a quantidade de ciclos de busca foi fixada em um valor constante de 25 ciclos para todos os testes executados, com o objetivo de reduzir o período total para execução dos mesmos e manter um processo de busca equilibrado entre as diferentes instâncias e tipos de custo utilizados;

- Algoritmos: O algoritmo utilizado para execução de todas as simulações foi a heurística de Lin-Kernighan completa (lk1), uma vez que a mesma apresentou resultados favoráveis que balancearam custo e performance computacional. O algoritmo simplificado (lk2) apresentou excelentes resultados, porém nas instâncias acadêmicas de maior dimensão a performance computacional foi levemente afetada, o que poderia ser um problema no caso industrial;

- Inicialização: No caso das duas instâncias de menor dimensão a rota inicial foi obtida a partir do embaralhamento de seus respectivos vértices. Para o caso da instância de maior dimensão, por uma questão de performance computacional, foi utilizada a heurística do vizinho mais próximo como método de inicialização, a fim de reduzir o tempo total de simulação;

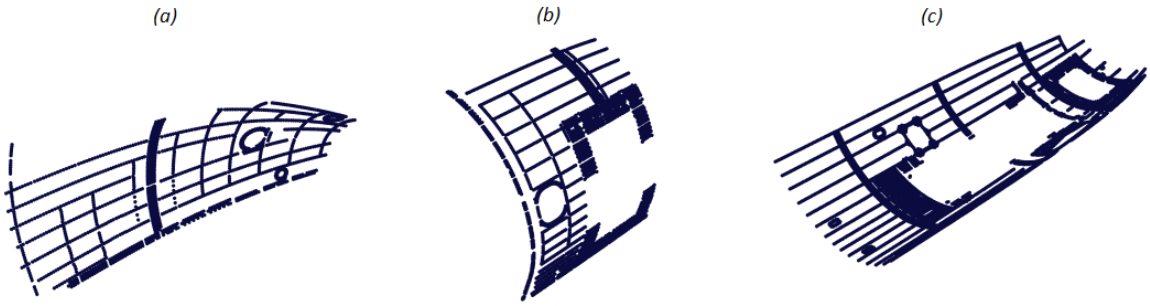
- Pesos: Para as simulações com custo agrupado foram definidos três valores distintos do peso de agrupamento ($w_{peças}$), conforme Equação 3.11, a fim de avaliar a influência desse parâmetro na performance de resolução e na qualidade das soluções obtidas nessas simulações.

Tabela 24 – Conjunto de testes para as instâncias industriais.

teste	**custo**	**ciclos**	**camadas**	**redução**	**algoritmo**	$w_{peças}$
1	*euclideano*	25	(25,25)	0	*lk1*	-
2	*agrupado*	25	(25,25)	0	*lk1*	1.5
3	*agrupado*	25	(25,25)	0	*lk1*	2.0
4	*agrupado*	25	(25,25)	0	*lk1*	6.0
5	*temporal*	25	(25,25)	0	*lk1*	-

Fonte: O autor.

Figura 49 – Grafo com as instâncias industriais: (a) *i1630*, (b) *i2751* e (c) *i5435*.



Fonte: O autor.

Os resultados com o melhor custo, custo médio e tempo médio, coletados para as instâncias acadêmicas, também foram coletados para os testes industriais. O sequenciamento de rebites definido no programa CNC desenvolvido pelos programadores, e em operação no equipamento automatizado, foi utilizado para definição do custo atual de cada instância. A partir da comparação entre esse custo atual e o custo simulado é possível determinar o ganho obtido pela simulação. Além disso, para as melhores rotas resultantes das simulações com custo euclideano e agrupado, foram calculados os respectivos custos temporais, com o objetivo de avaliar as consequências do sequenciamento obtido por cada uma dessas funções de custo no tempo final de operação do equipamento.

7.2.1 Instância *i1630*

Os primeiros resultados industriais a serem apresentados foram obtidos a partir das simulações com a menor instância industrial, a qual contém 1630 pontos de rebitagem localizados na região lateral dianteira do avião. Os resultados obtidos são exibidos na Tabela 25 e as comparações temporais para cada resultado são exibidas na Tabela 26. A Figura 50 ilustra as melhores rotas encontradas para cada teste executado.

Tabela 25 – Resultados para a instância *i1630*.

		custo			**ganho(%)**	
teste	***t*(s)**	*melhor*	*médio*	*atual*	*melhor*	*médio*
1	35.81	38624.3	39360.7	55721.9	30.7	29.4
2	45.06	40920.3	41816.5	65226.4	37.2	35.8
3	34.00	42486.3	43857.9	74730.8	43.1	41.3
4	44.43	55072.6	59464.8	150766.6	63.4	60.5
5	98.86	20619.1	20759.3	21552.6	4.3	3.6

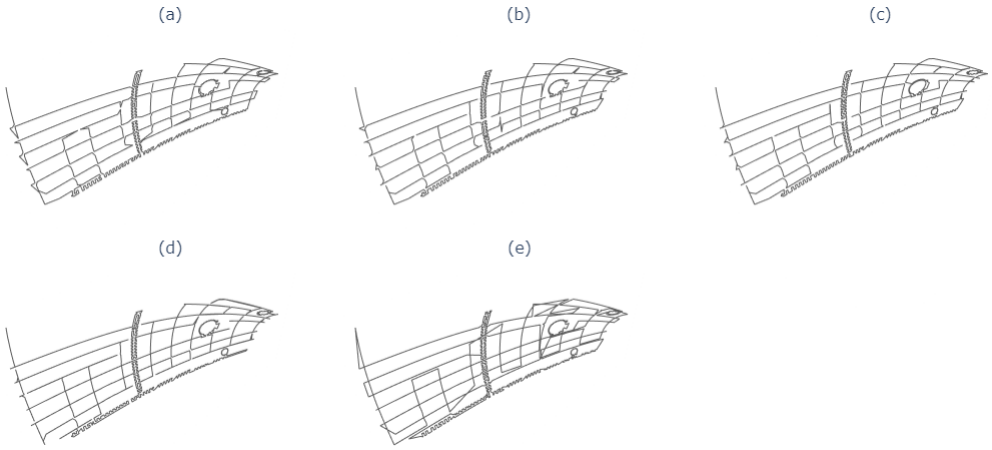
Fonte: O autor.

Tabela 26 – Resultados temporais para a instância *i1630*.

	custo temporal		
teste	*melhor*	*atual*	**ganho(%)**
1	180674.0	21552.6	-738.2
2	164421.9	21552.6	-662.8
3	169830.1	21552.6	-687.9
4	96210.8	21552.6	-346.4
5	20619.1	21552.6	4.3

Fonte: O autor.

Figura 50 – Melhores rotas para a instância *i1630*: (a) teste 1, (b) teste 2, (c) teste 3, (d) teste 4 e (e) teste 5.



Fonte: O autor.

O teste 1 apresenta um crescimento não-linear no tempo de simulação quando comparado à instância acadêmica *d1291*, fato justificado pelo aumento dimensional e exploratório da instância industrial, por meio da maior profundidade nas camadas de busca do método heurístico. Em comparação com a solução atual em operação no equipamento de rebitagem, obteve-se uma redução próxima de 30% na movimentação do equipamento. Em contrapartida, conforme Tabela 26, a redução no tempo de movimentação gera um efeito contrário no custo temporal da rota obtida, a qual teria um tempo de execução 7 vezes maior quando comparada ao tempo de execução atual em operação. Tal fato decorre da grande quantidade de *setups* que ocorrem na rota obtida quando apenas a distância entre os rebites é utilizada como estratégia de sequenciamento.

Os testes 2, 3 e 4 apresentam os resultados das simulações com custo agrupado, nos quais é possível confirmar a tendência de maior dificuldade de resolução do PCV em instâncias com vértices agrupados, comportamento observado anteriormente em simulações de determinadas instâncias acadêmicas. O crescente ganho percentual agrupado a partir do aumento do fator de agrupamento ($w_{peças}$) demonstra que a estratégia atual possui fatores adicionais para sequenciamento dos rebites. Em contrapartida, o agrupamento mais agressivo do teste 4 gerou a menor perda no ganho temporal, demonstrando uma correlação entre as peças aeronáuticas e o *setup* do equipamento nesta instância.

A execução do teste 5 apresentou tempo médio de simulação próximo a 100 segundos, valor consideravelmente maior quando comparado aos outros testes. Este comportamento é comum em instâncias não geométricas do PCV, condição existente pela utilização do custo temporal entre rebites. O ganho temporal obtido se aproxima de 5%, o qual, conforme Equação 3.9 e modelagem do problema industrial, representa um ganho absoluto

igual à $(21552.6 - 20619.1)/2 = 466.7$ segundos, aproximadamente 7 minutos, no tempo de operação do equipamento. Apesar de pequeno quando analisado isoladamente, tal valor é representativo considerando a grande quantidade de produtos aeronáuticos a serem otimizados e de máquinas rebitadoras existentes no parque fabril da empresa patrocinadora deste trabalho.

Apesar da dificuldade em representar rotas tridimensionais em um plano bidimensional, é possível observar pela Figura 50 determinadas características geométricas de cada teste realizado. A rota euclideana, por exemplo, possui diversas arestas anguladas em diagonais para evitar, no final da rota, o cruzamento entre arestas. Para o caso agrupado, tais conexões anguladas ocorrem com menor frequência, principalmente no teste 4, no qual o fator de agrupamento é maior. Para a rota temporal, a partir da estratégia de evitar *setups* de máquina, é possível observar diversas arestas geometricamente longas, mas que, na realidade, apresentam custo temporal reduzido.

7.2.2 Instância *i2751*

O segundo grupo de simulações industriais foi realizado com uma instância de tamanho médio contendo 2751 pontos de rebitagem localizados na região da porta dianteira do avião. Os resultados obtidos com as simulações desta instância são exibidos na Tabela 27 e as comparações temporais para cada resultado são exibidas na Tabela 28. A Figura 51 ilustra as melhores rota encontradas para cada teste executado.

Tabela 27 – Resultados para a instância *i2751*.

		custo			**ganho(%)**	
teste	***t*(s)**	*melhor*	*médio*	*atual*	*melhor*	*médio*
1	93.83	72345.7	73570.5	109523.0	33.9	32.8
2	144.86	73931.3	75367.6	117321.8	36.9	35.8
3	136.30	74809.4	77176.2	125120.6	40.2	38.3
4	149.85	82707.7	85229.1	187511.3	55.8	54.5
5	308.59	30221.8	30326.3	33203.1	8.9	8.6

Fonte: O autor.

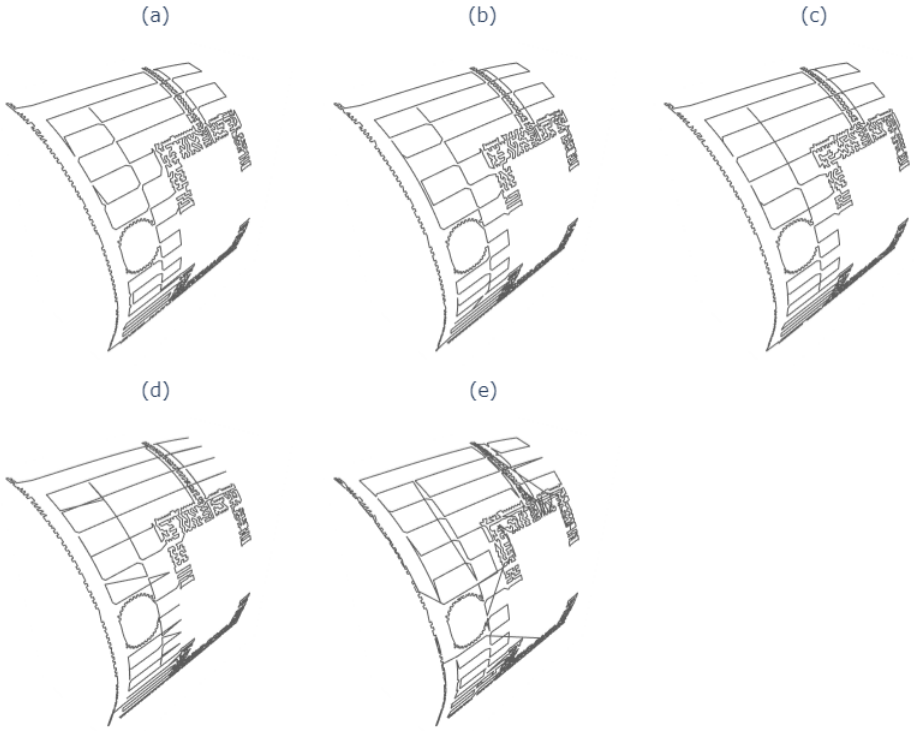
O teste 1, realizado com a função de custo euclideano, apresentou resultado equivalente ao observado no teste euclideano da instância de menor dimensão, obtendo a maior performance computacional entre todos os testes realizados com a instância *i2751*. Além da performance computacional, os valores do ganho percentual obtido pelo processo heurístico e da perda temporal resultante do sequenciamento euclideano puro também foram similares aos valores encontrados para a instância *i1630*, fato curioso uma vez que as rotas de tais instâncias apresentam características geométricas distintas.

Tabela 28 – Resultados temporais para a instância *i2751*.

	custo temporal		
teste	*melhor*	*atual*	**ganho(%)**
1	248879.6	33203.1	-649.5
2	252464.9	33203.1	-660.3
3	248985.1	33203.1	-649.8
4	227719.9	33203.1	-585.8
5	30221.8	33203.1	8.9

Fonte: O autor.

Figura 51 – Melhores rotas para a instância *i2751*: (a) teste 1, (b) teste 2, (c) teste 3, (d) teste 4 e (e) teste 5.



Fonte: O autor.

Os testes 2, 3 e 4 também obtiveram resultados similares quando comparados aos testes que utilizaram as mesmas funções de custo para a instância de menor dimensão, nos quais pode-se observar novamente a maior dificuldade de resolução do PCV em instâncias agrupadas, condição evidenciada pelo salto dimensional considerável entre ambas as instâncias. O teste 4, com maior peso de agrupamento, apresentou maior perda temporal quando comparado ao resultado obtido com a instância *i1630*, demonstrando uma menor vantagem no agrupamento de vértices para a instância de tamanho médio.

O tempo médio de simulação do teste 5 apresentou o mesmo padrão observado na instância *i1630*, nos quais se verifica a maior dificuldade de resolução do PCV industrial quando utilizada a função de custo temporal para definição das arestas. Em contrapartida, o ganho temporal encontrado para a instância *i2751* foi aproximadamente o dobro do ganho encontrado para a instância de menor dimensão, chegando, em valores absolutos, próximo de 25 minutos de redução temporal, resultado excelente para um único produto aeronáutico.

Novamente, os grafos com os melhores resultados ilustrados na Figura 51 demonstram características observadas anteriormente nas rotas da instância *i1630*, principalmente no caso temporal, no qual diversas arestas longas são definidas para evitar *setups* do equipamento automatizado. A simulação euclideana e as simulações com menor peso de agrupamento apresentam diversos saltos entre peças, representados pelas pequenas arestas anguladas, as quais são mais raras na melhor rota resultante do teste 4, uma vez que este utiliza maior peso de agrupamento entre peças.

7.2.3 Instância *i5435*

A maior instância industrial, contendo 5435 vértices, representa a região dianteira inferior do avião. Os resultados do processo heurístico estão apresentados na Tabela 29 e os resultados em função do tempo de operação do equipamento estão presentes na Tabela 30. A Figura 52 ilustra as melhores rotas obtidas para as simulações desta instância industrial.

Tabela 29 – Resultados para a instância *i5435*.

		custo			**ganho(%)**	
teste	***t*(s)**	*melhor*	*médio*	*atual*	*melhor*	*médio*
1	426.20	129298.6	130452.2	195114.4	33.7	33.1
2	451.50	131530.4	133095.3	220774.2	40.4	39.7
3	342.37	133255.9	135522.7	246434.0	45.9	45.0
4	466.40	149952.2	155110.8	451712.3	66.8	65.7
5	2754.48	56465.6	56699.1	60786.6	7.1	6.7

Fonte: O autor.

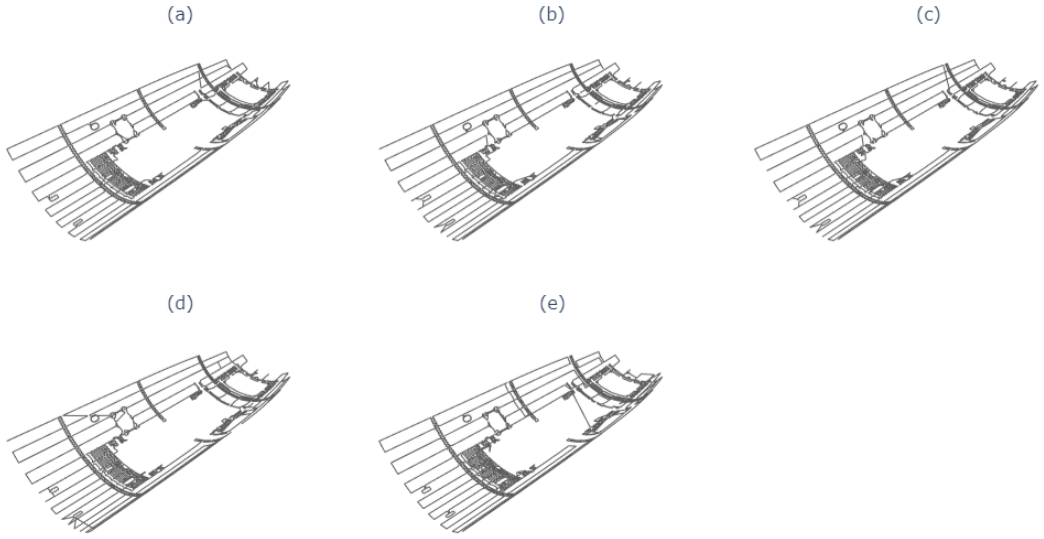
O teste 1, realizado com uso do custo euclideano, novamente apresenta um ganho próximo de 30% na movimentação do equipamento, resultado não esperado entre instâncias industriais com geometrias tão distintas entre si. O aumento considerável do tempo de simulação em relação às instâncias *i1630* e *i2751* representa a complexidade não-linear do método heurístico. A perda temporal encontrada foi menor em relação às duas instâncias menores uma vez que a maior parte dos vértices pertencem a um único *setup*, diminuindo assim a probabilidade de ocorrência desses eventos de reconfiguração da máquina.

Tabela 30 – Resultados temporais para a instância *i5435*.

	custo temporal		
teste	*melhor*	*atual*	**ganho(%)**
1	117343.1	60786.6	-93.0
2	124568.3	60786.6	-104.9
3	120984.2	60786.6	-99.0
4	124996.6	60786.6	-105.6
5	56465.6	60786.6	7.1

Fonte: O autor.

Figura 52 – Melhores rotas para a instância *i5435*: (a) teste 1, (b) teste 2, (c) teste 3, (d) teste 4 e (e) teste 5.



Fonte: O autor.

Os testes 2, 3 e 4 também apresentaram um padrão similar aos testes obtidos com as instâncias de menor dimensão, principalmente com relação ao ganho percentual crescente, até um limite aproximado de 60% para o teste em que se utiliza o maior valor do peso de agrupamento entre peças. O tempo de simulação para o teste 3 foi o menor entre as simulações com custo agrupado, resultado também ocorrido nos testes com as demais instâncias quando utilizado o peso de agrupamento $w_{peças} = 2$.

O teste 5 demonstra um elevado crescimento no tempo de simulação médio quando comparado aos testes equivalentes realizados nas instâncias de menor dimensão, sendo aproximadamente 7 vezes maior em relação ao teste com a instância *i2751*, com duração média próxima de 45 minutos. Apesar do longo tempo, é importante ressaltar que a instância *i5435* representa uma exceção dimensional de determinados produtos aeronáuticos, sendo

raros os exemplares que possuem mais de 5000 rebites em um único programa CN. O ganho temporal manteve uma proporção similar aos testes das instâncias menores, com melhor resultado percentual de 7%, o que gera uma redução próxima de 36 minutos no tempo de execução do equipamento.

Devido à grande quantidade de vértices da instância *i5435*, a visualização de seu grafo é comprometida, porém as mesmas considerações feitas para os grafos anteriores são válidas neste caso, apesar de menos visíveis. Um diferencial notável ao observar a Figura 52 é a similaridade geométrica entre todas as rotas, fato justificado por uma observação destacada anteriormente, na qual verifica-se que a maior parte dos vértices de rebitagem pertencem ao mesmo *setup* do equipamento, existindo assim uma menor quantidade de arestas longas visíveis em seus respectivos grafos.

7.2.4 Análise dos resultados para as instâncias industriais

Os resultados euclideanos e agrupados obtidos através das simulações com as instâncias industriais reforçam a complexidade polinomial de resolução da heurística de Lin-Kernighan em relação ao crescimento dimensional das instâncias, além de ter sido verificado novamente a maior dificuldade na resolução de instâncias agrupadas, conforme visto nas simulações acadêmicas.

A Tabela 31 apresenta um resumo dos resultados temporais obtidos para cada instância industrial. A coluna $\Delta/2$ apresenta o ganho temporal em segundos (conforme modelagem do problema de sequenciamento de rebites na Seção 3.4), sendo este convertido em minutos e porcentagem na coluna dos ganhos, com objetivo de facilitar a interpretação de tais resultados.

Tabela 31 – Resumo dos resultados temporais.

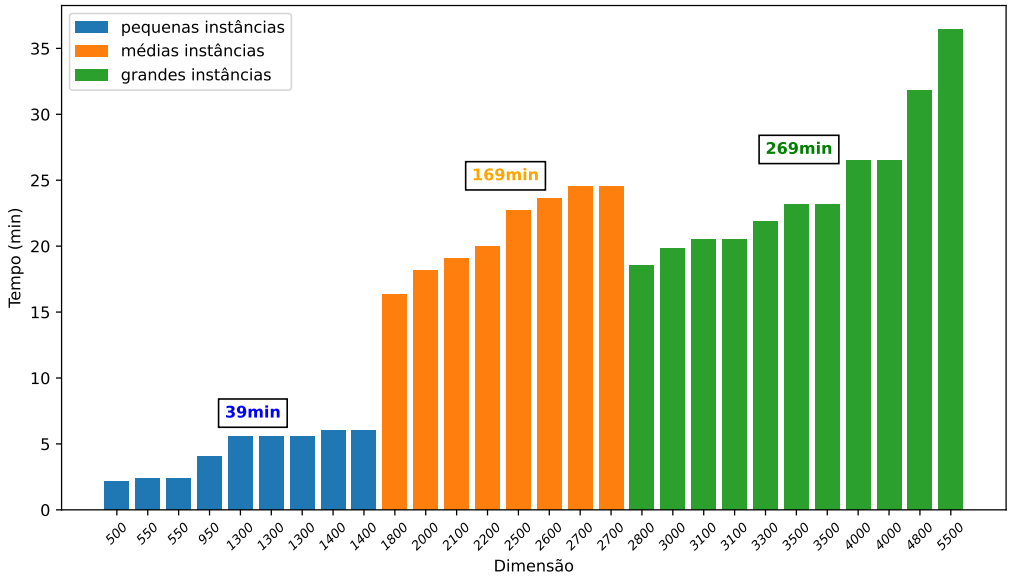
	custo temporal			**ganho**	
instância	*atual*	*melhor*	$\Delta/2$	*minutos*	*percentual*
i1630	21552.6	20619.1	466.7	7.7	4.3
i2751	33203.1	30221.8	1490.6	24.8	8.9
i5435	60786.6	56465.6	2160.5	36.0	7.1

Fonte: O autor.

Ao somar os ganhos temporais obtidos de todas as instâncias industriais, é encontrado uma redução total de operação do equipamento automatizado maior que 1 hora, porém existem mais produtos aeronáuticos que fazem parte do avião deste estudo que são parte do escopo da rebitagem automatizada, os quais podem ser considerados em uma estimativa total de ganho temporal. Para estimar tal ganho foi utilizado um valor

aproximado da quantidade de aplicações automatizadas para a aeronave e a respectiva dimensão de cada aplicação, sendo assim possível estimar o ganho temporal de cada produto. Através do valor da dimensão estimada, os ganhos temporais são calculados em diferentes faixas utilizando como referência a redução temporal obtida em cada instância industrial simulada neste capítulo. Os resultados dessas estimativas estão apresentadas na Figura 53.

Figura 53 – Ganho temporal estimado para o avião.



Fonte: O autor.

Através da estimativa de ganho temporal para cada produto aeronáutico, foi obtida uma redução temporal total próxima de 8 horas de trabalho do equipamento automatizado por avião, resultado importante para redução do custo de fabricação da aeronave e aumento da eficiência produtiva do equipamento automatizado. Além disso, os resultados encontrados podem ser expandidos para outras famílias de aeronaves fabricadas pela mesma empresa que também possuam produtos rebitados pelo mesmo equipamento automatizado, o que ampliaria consideravelmente o ganho temporal obtido.

Em contrapartida aos resultados alcançados, é importante ressaltar que a modelagem da função de custo industrial deste estudo foi simplificada com a finalidade de obter resultados preliminares do sequenciamento de rebites e comprovar o potencial de tal estudo de otimização. Para que o custo temporal seja mais representativo, determinados critérios qualitativos de rebitagem e condições pré-fixadas do equipamento automatizado devem ser incluídos na modelagem definida para o sequenciamento de rebites. A partir do acréscimo de novos parâmetros, é esperado que o resultado final flutue, mas ainda alcance valores positivos de ganho temporal, uma vez que grande parte do sequenciamento deve ser mantido nesta nova estratégia.

8 CONCLUSÕES

O PCV é um dos principais problemas do campo da Pesquisa Operacional, aplicável à diversas áreas industriais e científicas que necessitam do apoio de algoritmos de otimização para resolver seus respectivos problemas e desafios. Apesar de ser um problema antigo e bastante explorado por diversos pesquisadores ao longo do século 20, o PCV continua sendo relevante nos dias atuais. Conforme observado nas referências bibliográficas, novos métodos e estudos foram desenvolvidos nos últimos anos com o objetivo de otimizar a performance computacional para sua resolução, utilizando principalmente métodos heurísticos derivados da heurística de Lin-Kernighan.

A implementação computacional da heurística de Lin-Kernighan foi importante para a resolução do problema industrial de sequenciamento de rebites e para servir como base de consulta e inspiração para novos pesquisadores que pretendem estudar este método heurístico. Através da implementação em uma linguagem de programação de fácil acesso e entendimento, o uso de documentação no próprio código computacional e o compartilhamento da ferramenta em diversos meios eletrônicos, pode-se concluir que o objetivo acadêmico de compartilhar conhecimento foi executado com sucesso.

Os testes realizados em instâncias acadêmicas demonstram que o algoritmo implementado é totalmente funcional e pode ser utilizado para encontrar rotas otimizadas do PCV. Os resultados numéricos apresentados nas simulações se aproximam dos resultados de Kernighan e Lin (1973) e soluções globais são encontradas com uma frequência considerável. Os testes executados com as instâncias industriais demonstraram potencial da ferramenta para resolução do problema de sequenciamento de rebites, tendo obtido uma redução significativa no tempo de operação do equipamento em todos os produtos simulados. A estratégia de agrupamento de vértices por peça rebitada, de modo a formar instâncias menores e isoladas do PCV, demonstrou potencial para acelerar as simulações sem um grande comprometimento do custo da solução final.

Como sugestão para trabalhos futuros, do ponto de vista da implementação computacional, poderiam ser utilizadas técnicas de inteligência artificial, como o Aprendizado por Reforço, em combinação com a heurística implementada para se obter novos resultados. Essa tendência foi observada em trabalhos recentes envolvendo o PCV e indica um potencial para otimizar ainda mais o sistema computacional implementado neste trabalho. Melhorias no desenvolvimento realizado, como a implementação de estruturas de dados mais eficientes e o aumento na eficiência computacional de refinamentos, como o *checkout*, também são opções para trabalhos futuros. Tais demandas foram documentadas e propostas para pesquisadores na página oficial do pacote computacional. Do ponto

de vista industrial, um estudo mais detalhado da estratégia de agrupamento de rebites por *setup* representa um potencial novo método para acelerar a resolução do problema industrial. Outra possibilidade para estudos futuros é a inclusão de restrições e fatores qualitativos para sequenciamento de rebites, os quais devem acarretar no crescimento do custo temporal das soluções quando comparadas às obtidas neste trabalho.

REFERÊNCIAS

AL-LAMI, A.; HILMER, P.; SINAPIUS, M. Eco-efficiency assessment of manufacturing carbon fiber reinforced polymers (CFRP) in aerospace industry. *Aerospace Science and Technology*, Elsevier Masson SAS, v. 79, p. 669–678, 2018. Citado na página 39.

ALATARTSEV, S. Robotic Task Sequencing Problem: A Survey. *Journal of Intelligent and Robotic Systems: Theory and Applications*, v. 80, n. 2, p. 279–298, 2015. ISSN 15730409. Citado 2 vezes nas páginas 30 e 31.

APPLEGATE, D.; BIXBY, R.; COOK, W. Finding Tours in the TSP. *Institute for Discrete Mathematics, Universitat Bonn*, p. 59, 1994. Citado na página 67.

APPLEGATE, D.; COOK, W.; ROHE, A. Chained Lin-Kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, v. 15, n. 1, p. 82–92, 2003. Citado 5 vezes nas páginas 26, 27, 65, 85 e 86.

Australian Aerospace Engineering. *Full-Line Catalogue*. 2017. 224 p. Disponível em: <www.aaestore.com.au>. Citado na página 38.

BRESSON, X.; LAURENT, T. The Transformer Network for the Traveling Salesman Problem. 2021. Citado 2 vezes nas páginas 31 e 34.

CALAWA, R.; CULPEPPER, J. *Automatic Stringer Drilling System*. 1994. 1–8 p. Citado 2 vezes nas páginas 40 e 41.

CASTRO, F. *Lin-Kernighan Heuristic in Python*. 2022. Disponível em: <https://github.com/kikocastroneto/lk_heuristic>. Citado 5 vezes nas páginas 66, 71, 73, 74 e 76.

CHENG, H. et al. Optimization method of fixture layout for aeronautical thin-walled structures with automated riveting. *Assembly Automation*, v. 32, n. 4, p. 323–332, 2012. Citado na página 23.

CHRISTOFIDES, N. Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. *Technical Report*, v. 388, p. 10, 1976. Citado na página 26.

COLLETTE, Q. Evolution of historical riveted connections: Joining typologies, installation techniques and calculation methods. *WIT Transactions on the Built Environment*, v. 118, p. 295–306, 2011. ISSN 17433509. Citado na página 36.

COOK, W. *In Pursuit of the Traveling Salesman*. [S.l.]: Princeton University Press, 2011. 228 p. Citado 2 vezes nas páginas 25 e 26.

COSTA, P. R. d. O. da et al. Learning 2-opt Heuristics for the Traveling Salesman Problem via Deep Reinforcement Learning. 2020. Citado 2 vezes nas páginas 31 e 33.

CROES, G. A Method for Solving Traveling-Salesman Problems. *Operations research*, v. 6, n. 6, p. 791–812, 1958. ISSN 26883627. Citado 2 vezes nas páginas 65 e 96.

DANTZIG, G.; FULKERSON, R.; JOHNSON, S. Solution of a Large-Scale Traveling-Salesman Problem. *Operations Research*, v. 2, n. 4, p. 393–410, 1954. Citado 2 vezes nas páginas 26 e 29.

GARTSTEIN, M. A.; PUTNAM, S.; KLIEWER, R. The effects of feed force on rivet bucking bar vibrations. *Physiology And Behavior*, v. 176, n. 3, p. 139–148, 2016. Citado na página 37.

General Eletric. *GE635 System Manual*. [S.l.]: General Eletric Computer Department, 1964. 75 p. Citado na página 52.

HELSGAUN, K. An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, v. 126, n. 1, p. 106–130, 2000. ISSN 0377-2217. Citado 8 vezes nas páginas 21, 26, 60, 61, 63, 65, 91 e 93.

HITOMI, K. Automation - short history. *Technovation*, v. 14, n. 2, p. 121–128, 1994. Citado na página 39.

HOLT, S.; CLAUSS, R. *Robotic Drilling and Countersinking on Highly Curved Surfaces*. 2015. 1–4 p. Citado 2 vezes nas páginas 41 e 42.

IMESON, F.; SMITH, S. L. Clustering in discrete path planning for approximating minimum length paths. *Proceedings of the American Control Conference*, p. 2968–2973, 2017. ISSN 07431619. Citado na página 31.

International Federation of Robotics. *World Robotics Report 2021*. [S.l.], 2021. 1–32 p. Disponível em: <https://ifr.org/>. Citado na página 41.

JANSSEN, C. P. et al. History and future of human-automation interaction. *International Journal of Human Computer Studies*, Elsevier Ltd, v. 131, p. 99–107, 2019. Citado na página 39.

KERNIGHAN, B.; LIN, S. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, v. 21, n. 2, p. 498–516, 1973. Citado 15 vezes nas páginas 21, 24, 26, 34, 52, 53, 55, 56, 57, 59, 64, 85, 86, 96 e 108.

KRAR, S. Computer numerical control programming. *Choice Reviews Online*, v. 28, n. 01, p. 342, 1990. Citado na página 43.

LI, J. et al. *Colored traveling salesman problem and solution*. [S.l.]: IFAC, 2014. v. 19. 9575–9580 p. ISSN 14746670. ISBN 9783902823625. Citado 2 vezes nas páginas 30 e 31.

LI, X. et al. Research on application of NC program optimization based on TSP. *International Conference on Mechatronics and Automation*, IEEE, p. 1493–1498, 2009. Citado 2 vezes nas páginas 30 e 31.

Lima Junior, F. C. D. *Hybrid Metaheuristics Using Reinforcement Learning Applied to Salesman Traveling Problem*. [S.l.]: InTech, 2010. 213–236 p. ISBN 9789533074269. Citado 2 vezes nas páginas 30 e 31.

LIN, S. Computer Solutions of the Traveling Salesman Problem. *Bell System Technical Journal*, v. 44, n. 10, p. 2245–2269, 1965. ISSN 15387305. Citado na página 96.

LU, Y.; HAO, J. K.; WU, Q. Solving the Clustered Traveling Salesman Problem via TSP methods. 2020. Citado 3 vezes nas páginas 31, 33 e 34.

MAHÉO, A. *Implementing Lin-Kernighan in Python*. 2017. Disponível em: <https://arthur.maheo.net/implementing-lin-kernighan-in-python/>. Citado 8 vezes nas páginas 17, 21, 60, 61, 62, 63, 64 e 91.

MALAZGIRT, O. S. TauRieL: Targeting Traveling Salesman Problem with a deep reinforcement learning inspired architecture. 2019. Citado 2 vezes nas páginas 31 e 32.

MEIßNER, D. W. I. J. et al. Smart Human-Robot-Collaboration in Mechanical Joining Processes. *Procedia Manufacturing*, Elsevier B.V., v. 24, p. 264–270, 2018. ISSN 23519789. Citado 2 vezes nas páginas 39 e 41.

MERLUZZI, J.; BAHR, I. *Automatic Tool Change System for Stringer Side Rivet and Bolt Anvils on a D-Frame or C-Frame Fuselage Fastening Machine*. 2017. 1–5 p. Citado na página 23.

NAMMOUCHI, A. A Generative Graph Method to Solve the Travelling Salesman Problem. 2020. Citado 2 vezes nas páginas 31 e 33.

NEDJATIA, A.; VIZVARI, B. Robot Path Planning by Traveling Salesman Problem with Circle Neighborhood: modeling, algorithm, and applications. *Computational Research Progress in Applied Science and Engineering*, v. 6, p. 288–293, 2020. ISSN 24234591. Citado 2 vezes nas páginas 31 e 33.

Precedence Research. *Industrial Automation Market*. [S.l.], 2021. 150 p. Disponível em: <https://www.precedenceresearch.com/industrial-automation-market>. Citado na página 23.

REINELT, G. *Tsplib 95*. 1991. 1–16 p. Disponível em: <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/>. Citado 2 vezes nas páginas 75 e 79.

ROBINSON, J. On the Hamiltonian Game (A Traveling Salesman Problem). *Project Rand*, p. 1–10, 1949. Disponível em: <https://apps.dtic.mil/sti/citations/AD0204961>. Citado 3 vezes nas páginas 25, 26 e 29.

ROMI. *Centro de usinagem vertical ROMI D1500*. 2022. Disponível em: <https://www.romi.com/produtos/romi-d-1500/>. Citado na página 44.

SMID, P. *Programming Handbook Third Edition*. [s.n.], 2007. 577 p. Disponível em: <www.industrialpress.com>. Citado na página 43.

SUAREZ-RUIZ, F. RoboTSP - A Fast Solution to the Robotic Task Sequencing Problem. 2018. Citado 2 vezes nas páginas 31 e 32.

TERZIC, I. et al. A survey of distributed intelligence in automation in european industry, research and market. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, p. 221–228, 2008. Citado na página 23.

TONG, Y.; QU, L. Recent Patents in Riveting and Applications. *Magpulse Technologies*, v. 3, p. 220–227, 2009. Citado 3 vezes nas páginas 36, 38 e 40.

WEI, T. et al. Auto-normalization algorithm for robotic precision drilling system in aircraft component assembly. *Chinese Journal of Aeronautics*, Chinese Society of Aeronautics and Astronautics, v. 26, n. 2, p. 495–500, 2013. Citado na página 23.

WU, J. et al. Recent development of the novel riveting processes. *International Journal of Advanced Manufacturing Technology*, v. 117, n. 1-2, p. 19–47, 2021. Citado na página 39.

XING, Y. et al. Operations Research (OR) in Service Industries: A Comprehensive Review. *Systems Research and Behavioral Science*, v. 30, n. 3, p. 300–353, 2013. Citado na página 23.

XING, Z.; TU, S.; XU, L. *Solve Traveling Salesman Problem by Monte Carlo Tree Search and Deep Neural Network*. 2020. Citado 2 vezes nas páginas 31 e 32.

ZHAO, H. et al. A review on solid riveting techniques in aircraft assembling. *Manufacturing Review*, v. 7, 2020. ISSN 22654224. Citado na página 36.

ZHENG, J. et al. Combining Reinforcement Learning with Lin-Kernighan-Helsgaun Algorithm for the Traveling Salesman Problem. 2020. Citado 2 vezes nas páginas 31 e 33.