IMPLEMENTAÇÃO DA HEURÍSTICA DE LIN-KERNIGHAN E SUA APLICAÇÃO NO SEQUENCIAMENTO DE PONTOS DE REBITAGEM

Frederico de Castro Neto (UNESP) frederico.neto@unesp.br
Edilaine Martins Soler (UNESP) edilaine.soler@unesp.br

Resumo

O Problema do Caixeiro Viajante é um dos problemas de otimização combinatória mais conhecidos e de difícil solução computacional, relacionado principalmente com o estudo de rotas otimizadas. Este trabalho apresenta a implementação e aplicação de um método heurístico eficiente para resolução deste problema: a heurística de Lin-Kernighan. Esta heurística foi implementada utilizando a linguagem de programação *Python*, de modo que um *framework* de fácil utilização e customização pudesse ser disponibilizado para pesquisadores que desejem estudar e utilizar a heurística. A aplicação da heurística é feita em uma base de dados de rebitagem automática disponibilizada por uma indústria aeronáutica nacional com objetivo de otimizar o sequenciamento desses pontos de trabalho, reduzindo assim o tempo total de execução do equipamento.

Palavras-Chaves: Problema do Caixeiro Viajante, Rebitagem automatizada, heurística de Lin-Kernighan.

1. Introdução

De acordo com Rita (2017) e Guariente (2017), o meio industrial moderno e seus processos de manufatura têm gerado ambientes cada vez mais competitivos, nos quais fabricantes buscam exaustivamente a melhoria contínua e excelência de produção com o objetivo de se manterem eficientes dentro dos mais variados mercados.

Um dos principais processos de manufatura na indústria aeronáutica é a rebitagem. Conforme Wang (2012), este processo de união é amplamente utilizado em componentes aeronáuticos por ser estruturalmente mais eficiente e economicamente mais viável do que outros métodos de união. Dada sua relevância, soluções robotizadas foram desenvolvidas ao longo das últimas décadas para automatizar este processo de manufatura, conforme exemplificado na Figura 1.

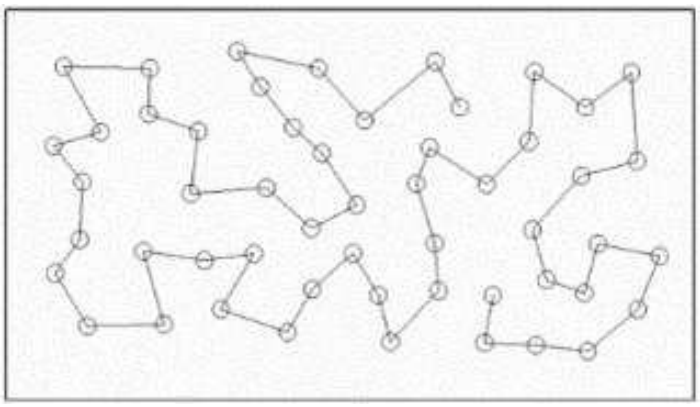
Figura 1 - Rebitadora automática

Fonte: Bigoney e Huddleston (2017)

A automação de processos de manufatura discretos (ponto-a-ponto) gera benefícios de produtividade e qualidade ao mesmo tempo em que cria um desafio peculiar para seus respectivos programadores CNC: definir o sequenciamento otimizado dos pontos de trabalho que resulte no menor tempo de uso do equipamento. Estudos com essa abordagem em processos de manufatura industrial, como soldagem e furação, foram realizados por Suarez-Ruiz (2018), Li (2009), J. Li (2014) e Nedjatia (2020).

Esse tipo de problema, no qual se busca determinar a rota para percorrer uma série de pontos que resulte no menor custo possível, se tornou objeto de estudo a partir da década de 30 e ficou conhecido popularmente por Problema do Caixeiro Viajante, conforme mostrado na Figura 2. Este é um dos problemas de otimização combinatória de maior aplicação real, sendo relevante em diversas áreas como processos de manufatura, rotas veiculares, astronomia, sequenciamento genético, entre outras áreas.

Figura 2 - Problema do Caixeiro Viajante



Fonte: Adaptado de X. Li (2009)

Dentre os estudos de diversos métodos para resolução deste problema, Kernighan (1973) propôs uma heurística que ganhou destaque a partir da década de 70, conhecida como heurística de Lin-Kernighan. Desde a sua primeira publicação, diversos pesquisadores criaram variações das implementações e estratégias adotadas pelo algoritmo original, com objetivo de obter melhores resultados com uma performance computacional maior. O exemplo mais conhecido e de maior sucesso na atualidade é o chamado algoritmo LKH, proposto por Helsgaun (1998). Mesmo com sua fama, o volume de implementações desta heurística é baixo, sendo a maior parte de difícil compreensão e intimidadoras em um primeiro momento, principalmente para pesquisadores que desejam desenvolver suas próprias implementações.

O objetivo deste trabalho é realizar a implementação da heurística de Lin-Kernighan e aplicá-la na otimização do sequenciamento automatizado de rebites aeronáuticos. A implementação foi realizada em linguagem Python, uma linguagem de programação mais moderna e de fácil acesso e entendimento de modo a se disponibilizar uma base computacional para uso de pesquisadores que, a partir da mesma, possam então criar suas próprias variações e desenvolver seus próprios algoritmos. A utilização da heurística gerou resultados computacionais satisfatórios dentro da aplicação industrial estudada convergindo em uma perspectiva promissora no uso da mesma para o sequenciamento automatizado de rebites aeronáuticos.

A seção 2 deste artigo apresenta os conceitos gerais da heurística de Lin-Kernighan e os detalhes da implementação na linguagem de programação *Python*. Na seção 3, o fluxo de coleta automatizado dos dados de rebitagem realizado na indústria aeronáutica é apresentado. A seção 4 demonstra os resultados numéricos obtidos em pequenas instâncias do problema encontradas na literatura e os resultados obtidos nas instâncias industriais. A seção 5 apresenta as principais conclusões em torno da implementação da heurística e dos resultados numéricos obtidos e ressalta as possibilidades para estudos futuros.

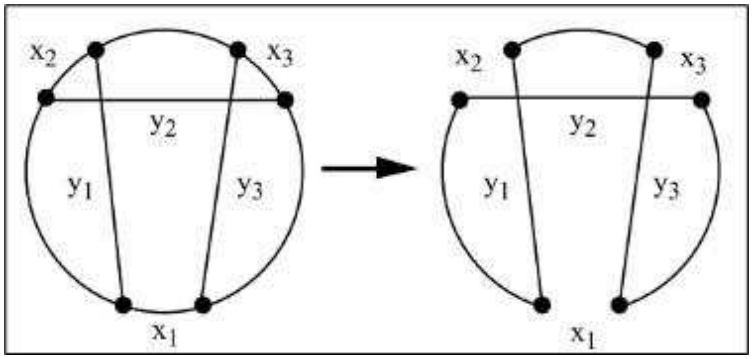
2. Heurística de Lin-Kernighan

2.1. Conceitos gerais

A base da heurística de Lin-Kernighan foi proposta por trabalhos de abordagem exploratória de soluções, entre as décadas de 50 e 60, conforme detalhado por Cook (2011). Nesses trabalhos, os movimentos computacionais chamados de *2-Opt* e *3-Opt* (Figura 3) foram propostos e investigados, de modo que, a partir de uma rota inicial representada por um grafo, duas ou três arestas entre vértices fossem "quebradas" (indicadas na Figura 3 como x_1, x_2, x_3) e "reconectadas" (indicadas na Figura 3 como y_1, y_2, y_3) formando assim uma nova rota de custo

reduzido, a qual passará pelo mesmo processo até que não existam mais movimentos que otimizem a rota em processamento.

Figura 3 - O movimento *3-Opt*



Fonte: Adaptado de Helsgaun (1998)

Na Figura 3 é importante observar que o custo das arestas *y* não é definido pela sua distância euclidiana, uma vez que estas arestas são graficamente mais longas que as arestas *x*. Outro ponto de observação importante é a seleção *sequencial* de arestas quebradas e reconectadas, ou seja, a partir do último vértice de reconexão parte a próxima aresta a ser quebrada, estratégia que também é utilizada na heurística de Lin-Kernighan.

Diferente dos movimentos citados anteriormente, a heurística de Lin-Kernighan propõe um tipo de movimento *adaptável*, baseado na diferença dos custos das arestas quebradas e reconectadas (chamada de *ganho*), de modo a selecionar duplas sequenciais dessas arestas que apresentem maior potencial de ganho, a fim de convergir para boas soluções de maneira mais rápida e eficiente. Segundo Kernighan (1973), resultados experimentais demonstram que a complexidade computacional da heurística se aproxima de $n^{2.2}$ (em que *n* define o número de vértices do problema), valor muito próximo da complexidade computacional do movimento *2-Opt*, porém convergindo para soluções melhores do que este último movimento.

Além da base exploratória explicada anteriormente, Kernighan (1973) também propôs 4 refinamentos principais para o método de modo a reduzir o tempo de processamento e direcionar a busca por arestas promissoras, os quais são listados a seguir:

a) Identificação de rotas repetidas: Ao se realizar a busca por novas arestas, rotas previamente descartadas podem reaparecer durante essa exploração e o tempo de execução para descarte das mesmas, chamado de *checkout*, pode ser significativo. Portanto, este refinamento propõe que rotas descartadas sejam memorizadas ao longo do processo de modo que não sejam mais consideradas em buscas futuras;

b) Ordenação de arestas promissoras: Durante a seleção de arestas a serem quebradas e reconectadas, propõe-se selecionar primeiro as duplas que gerem o maior ganho possível. Para que este processo de seleção, chamado de *lookahead*, não seja longo, o refinamento é proposto com uma quantidade limitada de duplas consideradas;
c) Permanência de arestas "boas": Conforme a busca alcança novas rotas otimizadas em relação às suas antepassadas, algumas arestas são mantidas sem modificação. A partir de um determinado momento, este refinamento propõe que estas arestas passem a não ser mais analisadas e alteradas, uma vez que provaram serem arestas "boas";
d) Movimentos não sequenciais: A heurística se baseia na conexão e reconexão de arestas sequenciais conforme explicado anteriormente para o movimento *3-Opt*. Este refinamento sugere que em determinados casos essa regra seja violada de modo a explorar soluções que não seriam alcançadas a partir de movimentos sequenciais.

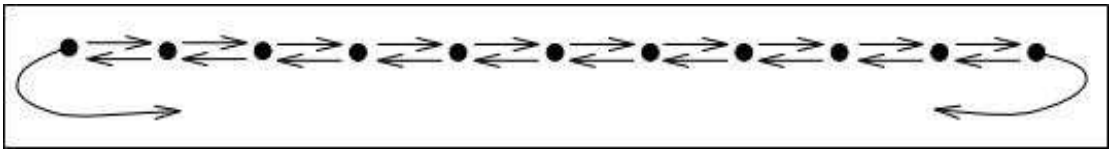
A partir dos conceitos básicos da heurística definidos, na próxima seção serão apresentados os fundamentos técnicos de implementação na linguagem de programação *Python*.

2.2. Implementação em *Python*

A implementação da heurística foi executada utilizando a linguagem de programação *Python*, justificada por sua simplicidade, popularidade e características como orientação à objeto que permitem uma implementação clara e de fácil entendimento, servindo assim também como material de consulta e apoio para outros pesquisadores interessados na heurística de Lin-Kernighan.

Um dos desafios de implementação que precede a própria heurística é a estruturação dos objetos que compõem o Problema do Caixeiro Viajante: o *vértice*, a *aresta* e a *rota*. Diversas modelagens foram propostas ao longo dos anos, e podem ser encontradas nos documentos de Helsgaun (1998), Applegate (1994) e Johnson (1990). Para a implementação neste trabalho, a principal estrutura extraída desses documentos é chamada de *Doubly Linked List*, utilizada para definição dos vértices e exemplificada na Figura 4.

Figura 4 - *Doubly Linked List*



Fonte: Applegate (1994)

Nessa lista duplamente conectada cada vértice possui uma propriedade de conexão ao vértice sucessor e predecessor, definindo assim uma sequência de vértices que dará origem às arestas e a rota do Problema do Caixeiro Viajante. Do ponto de vista computacional, a lista duplamente conectada permite operações eficientes de quebra e reconexão de arestas, o que tende a otimizar a execução do algoritmo, uma vez que o mesmo é fortemente baseado nesse tipo de operação.

Nesta implementação do algoritmo, os refinamentos chamados de *checkout* e *lookahead* propostos por Kernighan (1973) e detalhados na seção 2.1 foram implementados de modo a se otimizar a performance do algoritmo. Os outros 2 refinamentos, bem como outras melhorias existentes nas variações da heurística de Lin-Kernighan propostas por Helsgaun (1998) e Applegate (2003) serão considerados em trabalhos futuros.

Toda a implementação foi organizada para distribuição de um possível pacote *Python*, de fácil utilização e até mesmo revisão por outros usuários. A partir dos padrões globais de grupos de pesquisa do Problema do Caixeiro Viajante, o pacote inclui um conversor de arquivos de entrada *.tsp* padronizados. As principais instâncias estudadas por esses grupos se encontram disponíveis *online* na biblioteca *TSPLIB* (REINELT, 1995).

Apesar de não possuir uma interface gráfica, o sistema desenvolvido conta com recursos como execução interativa ou silenciosa pelo terminal do sistema operacional, relatórios de execução com maior ou menor nível de detalhamento, execução em batelada para avaliação de métricas de execução e um gerador de gráficos 2D/3D para visualização das rotas geradas. O pacote com todos os arquivos *Python,* guia de uso e exemplos de execução se encontra disponível na página do autor (CASTRO, 2022).

3. Coleta de dados

A coleta de dados realizada neste trabalho consistiu em uma etapa crítica e de longo ciclo pois teve como objetivo a definição de um fluxo de aquisição de dados padronizado e automatizado, a fim de que a mesma pudesse ser facilmente reaplicada por responsáveis da própria empresa aeronáutica em quaisquer produtos a serem investigados. Para realização desta etapa foi utilizado o programa de modelagem 3D *CATIA V5 R27*, fornecido pela *Dassault Systemes* (o mesmo programa utilizado pela empresa aeronáutica) e suas respectivas ferramentas de automação na linguagem *Visual Basic Automation* (VBA). Além disso, a empresa também forneceu os modelos 3D de três produtos aeronáuticos (Figura 5), contendo suas respectivas bibliotecas 3D de rebites, as quais foram utilizadas para extração das informações necessárias para realização do estudo.

Figura 5 - Produtos aeronáuticos disponibilizados para o estudo

Fonte: Os autores

A partir dos recursos disponibilizados, a extração de determinadas informações como a posição espacial do rebite e o seu tipo básico são de fácil acesso, uma vez que são propriedades do próprio objeto 3D. Ao contrário, informações como as peças ligadas a cada rebite e a presença de pré-furos ("furos guias") nos pontos de rebitagem não são explicitamente expostas por cada objeto 3D. Para automatizar a extração destas últimas informações foi necessário o desenvolvimento das seguintes automações em VBA listadas abaixo:

a) Ferramenta para substituição dos modelos 3D das peças que não possuem pré-furos pelas respectivas peças que possuem pré-furos (para mapeamento dos pontos de visão durante a rebitagem automatizada);

b) Ferramenta para identificação dos rebites localizados em pontos com pré-furos nas peças (para mapeamento dos pontos de visão durante a rebitagem automatizada);

c) Ferramenta para identificação das peças unidas por cada rebite (para possível análise agrupada dos pontos de rebitagem). A interface desta ferramenta é exemplificada na Figura 6.

Figura 6 - Ferramenta para identificação das peças unidas por cada rebite

Fonte: Os autores

Ao final do processamento das informações, cada produto aeronáutico teve sua biblioteca de rebites exportada para um arquivo de texto separado por vírgulas, no qual cada linha representava um rebite do produto e cada coluna representava um atributo mapeado para o rebite, de modo a ser utilizado como entrada para o processamento do Problema do Caixeiro Viajante. Na próxima seção os resultados numéricos do processamento de 2 instâncias serão apresentados.

4. Resultados numéricos

4.1. Instâncias da literatura

Os resultados iniciais apresentados a seguir fazem parte da validação da implementação da heurística de Lin-Kernighan em instâncias de pequena dimensão disponíveis na biblioteca *TSPLIB*. A utilização dessas instâncias é importante para avaliação da performance do algoritmo e também para medição de sua assertividade em relação às soluções globais disponíveis. As instâncias selecionadas foram a *att48.tsp* e *a280.tsp*, com 48 e 280 vértices respectivamente. Os resultados de 100 simulações realizadas são apresentados na Tabela 1 .

Tabela 1 - Resultados para att48.tsp e a280.tsp

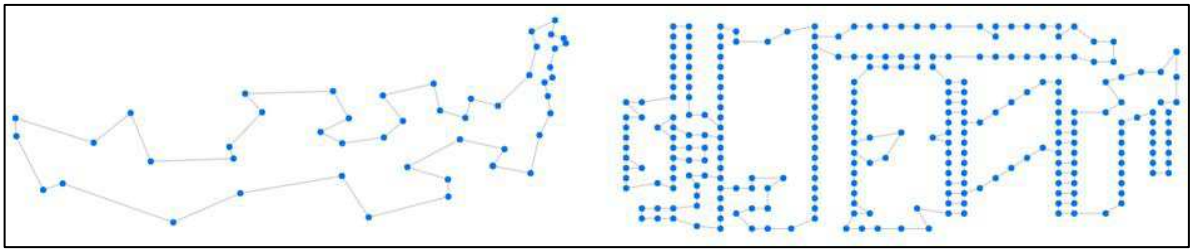
	att48.tsp	**a280.tsp**
Número de vértices	48	280
Simulações	100	100
Tempo médio	0.1s / simulação	2.2s / simulação
Ótimo global	33.522	2579
Melhor solução obtida	33.588 (0.19%)	2593 (0.56%)
Solução média obtida	34.533 (3.01%)	2795 (8.37%)

Fonte: Os autores

Os dados da Tabela 1 indicam uma performance aceitável da implementação desenvolvida. Os valores das soluções obtidas se aproximam muito dos ótimos globais conhecidos de cada problema, com um desvio menor que 1% nas duas instâncias. As rotas definidas pelas melhores soluções estão ilustradas na Figura 7.

Uma vez a heurística tendo apresentado bons resultados em instâncias pequenas, realizou-se o teste com as instâncias do problema de rebitagem e os resultados obtidos nessas simulações serão apresentados na próxima seção.

Figura 7 - Melhores soluções encontradas para att48.tsp (esq.) e a280.tsp (dir.)



Fonte: Os autores

4.2. Aplicação no sequenciamento de pontos de rebitagem

É importante ressaltar que nesta validação inicial aplicada ao problema industrial apenas a distância entre os rebites foi utilizada no cálculo dos custos das arestas. Desta forma, os resultados apresentados a seguir validam a trajetória da máquina automatizada, não significando que o tempo de produção seja o menor possível. Informações adicionais precisam ser consideradas, o que será feito em trabalhos futuros.

Uma vez considerada a distância entre os pontos de rebitagem para definição dos custos de cada aresta, foi extraída do programa CNC de cada produto a distância total percorrida pelo equipamento a fim de se comparar essa trajetória atual utilizada no ambiente fabril com a trajetória calculada pelo sistema de otimização.

As instâncias industriais passaram por um número menor de simulações uma vez que o ciclo de otimização é mais longo. Considerando que essas são instâncias muito maiores que as instâncias analisadas na seção 4.1, é esperado que suas simulações demorem mais tempo.

Foram realizados testes com duas instâncias e os resultados encontram-se listados na Tabela . Nota-se que a melhor solução obtida para a instância 1 se aproxima mais da solução média uma vez que mais simulações foram executadas e o tamanho desta instância é menor quando comparado à instância 2, que possui aproximadamente o dobro de vértices. O tempo de execução cresceu linearmente em relação ao número de vértices, fato importante uma vez que o crescimento no número de soluções possíveis em relação ao número de vértices é exponencial para o Problema do Caixeiro Viajante.

O ganho em relação a trajetória praticada na indústria e a melhor trajetória obtida foi de aproximadamente 30%. Apesar desse ganho representar apenas a diminuição na trajetória percorrida pelo equipamento sem considerar outros fatores, o mesmo representa uma possibilidade de ganho a ser encontrado em futuras simulações.

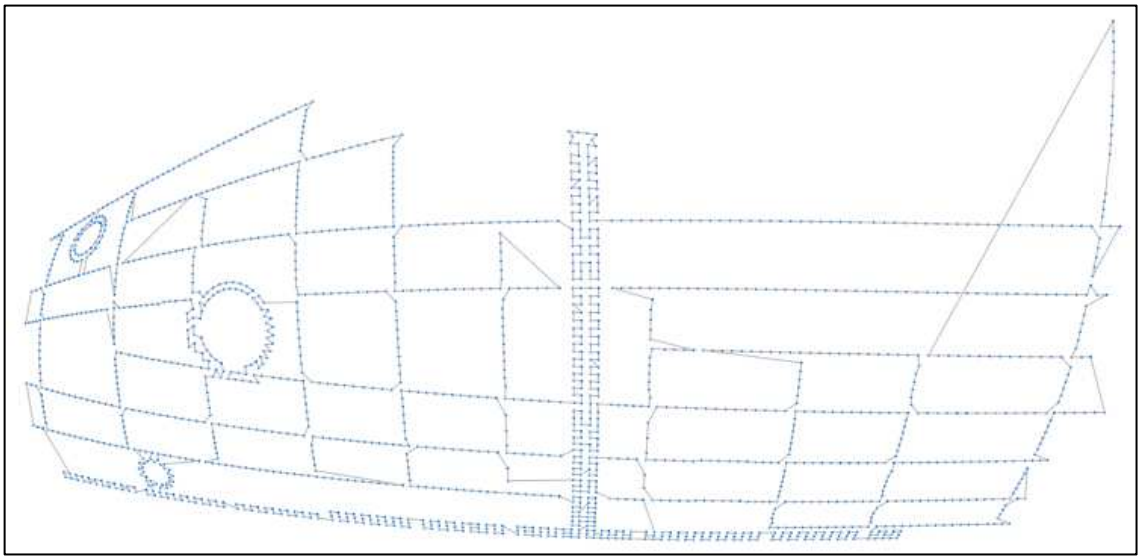
Tabela 2 - Resultados para instâncias industriais

	Instância 1	**Instância 2**
Número de vértices	1722	3730
Simulações	30	10
Tempo médio	470s / simulação	1075s / simulação
Trajetória atual	58.892	148.640
Ótimo global	-	-
Melhor solução obtida	41.448	112.690
Solução média obtida	43.135	125.980

Fonte: Os autores

Na Figura 9 é ilustrada a rota da melhor solução encontrada para a instância 1. É perceptível que se obteve um resultado melhor para a instância 1 a partir do menor número de longas arestas cruzando determinados vértices, fenômeno que ocorre com maior frequência e em maior tamanho na instância 2, gerando oportunidades de otimizações adicionais neste último problema.

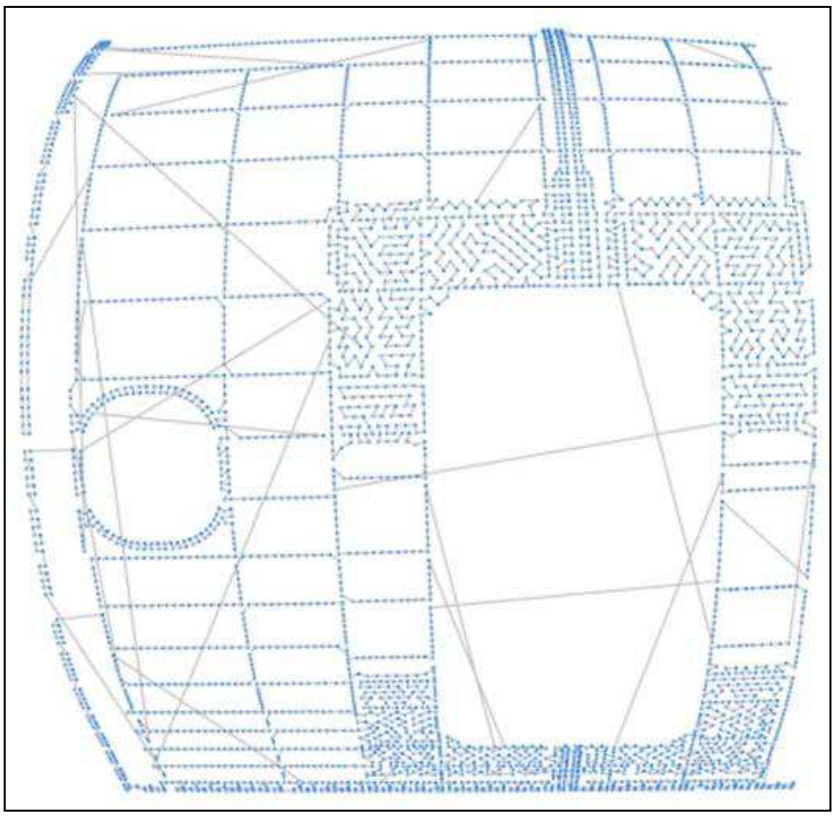
Figura 8 - Melhor solução encontrada para a instância 1



Fonte: Os autores

Na Figura 9 é ilustrada a rota da melhor solução encontrada para a instância 2. Além da maior quantidade de vértices, pode-se notar que a geometria definida por esses vértices apresenta maior complexidade quando comparada com a geometria da instância 1, uma vez que existe uma grande quantidade de vértices localizados próximos entre si em 2 regiões distintas, nas quais a maior parte das arestas longas se conectam em pelo menos uma extremidade.

Figura 9 - Melhor solução encontrada para a instância 2



Fonte: Os autores

5. Conclusão

A partir dos resultados apresentados pode-se concluir que a implementação da heurística de Lin-Kernighan na linguagem de programação *Python* foi executada com sucesso. Para as instâncias da literatura e de menor tamanho a velocidade de resolução é aceitável, levando-se em consideração que a solução implementada ainda pode ser aperfeiçoada. Já para as instâncias industriais, os resultados encontrados demonstram potencial da solução para ser utilizada como método de sequenciamento de rebites, uma vez que as trajetórias otimizadas apresentaram um ganho em distância de aproximadamente 30% quando comparadas com o sequenciamento feito na prática.

Como trabalhos futuros, propõe-se otimizar o algoritmo incluindo e testando os refinamentos sugeridos por Kernighan (1973), bem como outras melhorias sugeridas e implementadas em variações modernas da heurística. Com estas melhorias as instâncias industriais devem apresentar uma maior performance de resolução convergindo em melhores soluções. Além disso, para essas instâncias é desejado que a função de custo passe a incluir todos os parâmetros relevantes de cada rebite e não apenas a localização, com objetivo de estudar de maneira mais assertiva o tempo de trabalho do equipamento reduzido pela heurística, e não apenas a distância.

Para resolução de instâncias industriais ainda maiores propõe-se um estudo para aplicação de soluções agrupadas por atributos de rebitagem e a revisão do tipo de estrutura numérica adotada para representar as localizações dos rebites.

Por fim, espera-se que a implementação em *Python*, disponível *online* gratuitamente, colabore com entusiastas e pesquisadores da área, auxiliando no entendimento da heurística de Lin-Kernighan e incentivando novos estudiosos na área de pesquisa do Problema do Caixeiro Viajante.

6. Agradecimentos

Frederico de Castro Neto e Edilaine Martins Soler agradecem ao CNPq (Processo nº 314711/2020-1) e à FAPESP (Processo nº 2013/07375-0).

REFERÊNCIAS

Applegate, David, Robert Bixby e William Cook. 1994. "**Finding Tours in the TSP.**" *Institute for Discrete Mathematics, Universitat Bonn*: 59.

Applegate, David, William Cook e André Rohe. 2003. "**Chained Lin-Kernighan for Large Traveling Salesman Problems.**" *INFORMS Journal on Computing* 15(1): 82–92.

Bigoney, Burton e Nicholas Huddleston. 2017. "**Automated Riveting of C-130J Aft Fuselage Panels.**" *SAE International*.

Castro, Frederico. 2022. "**Lin-Kernighan Heuristic in Python.**" https://github.com/kikocastroneto/lk_heuristic.

Guariente, P. et al. 2017. "**Implementing Autonomous Maintenance in an Automotive Components Manufacturer.**" *Procedia Manufacturing* 13: 1128–34. https://doi.org/10.1016/j.promfg.2017.09.174.

Helsgaun, Keld. 1998. "**An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic**".

Johnson, David S. 1990. "**Data Structures for Traveling Salesmen.**" *Lecture Notes in Computer Science* 447 LNCS: 287.

Kernighan, S. Lin e B. W. 1973. "**An Effective Heuristic Algorithm for the Traveling-Salesman Problem.**" *Operations Research* 21(2): 498–516.

Li, Jun et al. 2014. 19 IFAC Proceedings Volumes (IFAC-PapersOnline). "**Colored Traveling Salesman Problem and Solution.**" IFAC. http://dx.doi.org/10.3182/20140824-6-ZA-1003.01403.

Li, Xueguang et al. 2009. "**Research on Application of NC Program Optimization Based on TSP.**" *2009 IEEE International Conference on Mechatronics and Automation, ICMA 2009*: 1493–98.

Nedjatia, Arman, e Béla Vizvárib. 2020. "**Robot Path Planning by Traveling Salesman Problem with Circle Neighborhood: Modeling, Algorithm, and Applications.**" http://arxiv.org/abs/2003.06712.

Reinelt, Gerhard. 1995. "**Tsplib 95.**" *Institut f¨ur Angewandte Mathematik, Universit¨at Heidelberg, Germany*: 1–16. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

Rita, Gamberini, Galloni Luca, Lolli Francesco e Rimini Bianca. 2017. "**On the Analysis of Effectiveness in a Manufacturing Cell : A Critical Implementation of Existing Approaches.**" *Procedia Manufacturing* 11(Junho): 1882–91. http://dx.doi.org/10.1016/j.promfg.2017.07.328 (Janeiro, 2019).

Suarez-Ruiz, Francisco, Teguh Santoso Lembono e Quang Cuong Pham. 2018. "**RoboTSP - A Fast Solution to the Robotic Task Sequencing Problem.**" *Proceedings - IEEE International Conference on Robotics and Automation*: 1611–16.

Wang, Run-xiao. 2012. "**Modeling and Analyzing of Variation Propagation in Aeronautical Thin-Walled Structures Automated Riveting.**" *Assembly Automation* 1: 25–37.

William J. Cook. 2011. "**In Pursuit of the Traveling Salesman.**" Princeton University Press.