O Problema do Caixeiro Viajante

Larissa Ribeiro Lopes, Fábio Henrique Viduani Martinez

28 de novembro de 2023

Resumo

Este artigo, apresenta uma visão abrangente dos elementos essenciais para a compreensão sólida do problema do caixeiro viajante (TSP). O TSP consiste em encontrar a rota mais curta possível para visitar uma série de pontos. Ele é um problema NP-difícil, o que significa que não sabe-se se é possível obter um algoritmo geral que possa encontrar a solução ótima em um tempo polinomial. Por fim, é apresentado dois algorítimos heurísticos, a técnica de Lin e Kernighan e o algorítimo de insere vértices, que apresentam soluções ótimas ou quase ótimas.

1 Introdução

Em uma viagem de volta ao ano de 2016, é impossível esquecer o furor causado por um jogo que conquistou o mundo: Pokémon Go. Desenvolvido pela Niantic em colaboração com a The Pokémon Company e a Nintendo, este jogo de realidade aumentada transformou as ruas em verdadeiros campos de caça para os fãs dos adoráveis monstros de bolso. A jogabilidade de Pokémon Go revolucionou a maneira como as pessoas interagiam com seus dispositivos móveis, incentivando-as a explorar o mundo real em busca de criaturas virtuais que pareciam habitar os locais mais inesperados. Os jogadores, munidos de seus smartphones, percorriam cidades, parques e áreas públicas, com seus olhos fixos nas telas dos aparelhos. Usando a câmera dos dispositivos e a realidade aumentada, esses jogadores tinham a incrível sensação de que os Pokémon estavam realmente à espreita nos cantos do ambiente real. Além de capturar as adoráveis criaturas, os jogadores também podiam interagir com locais especiais denominados PokeStops e Ginásios, que eram pontos de referência reais integrados ao jogo.

A exploração ativa e as visitas a esses pontos tornaram-se elementos fundamentais da experiência do Pokémon Go. No entanto, para alguns jogadores mais ávidos, surgiu um desafio intrigante: como otimizar sua jornada de caçador-coletor de Pokémon? Essa pergunta levou a uma investigação interessante e um encontro com um conceito matemático conhecido como o problema do caixeiro viajante, do inglês Travelling Salesman Problem (TSP). Felizmente, o mundo da matemática já vinha estudando o TSP por mais de seis décadas, e métodos consolidados estavam disponíveis para ajudar os jogadores a encontrar a rota mais curta possível. Se você vive em uma área repleta de centenas ou até milhares de PokeStops, a matemática

poderia revelar a melhor maneira de coletar todos eles. Foi exatamente isso o que o jornal Western Cincinnati's People's Observer (WCPO), em Cincinnati, fez ao montar um intrigante mapa contendo 551 PokeStops e ginásios. Para os entusiastas do TSP, esse mapa representou um verdadeiro desafio, uma oportunidade de aplicar suas habilidades matemáticas na otimização da rota perfeita para a captura de todos os Pokémon. Na figura 1, é apresentado o fascinante resultado desse empreendimento, revelando a forma mais eficiente de visitar todos os 551 locais a pé. Para ser preciso, esse passeio épico cobre uma distância de 358.331 metros, precisaria de um bom preparo físico para percorrer esse trajeto. O tempo total de cálculo para essa conquista notável, realizado em um Mac Mini, foi de apenas 101 segundos. Isso demonstra o poder da otimização matemática e da tecnologia, já que treinadores de Pokémon agora têm a chave para cumprir sua missão com a máxima eficiência [10].

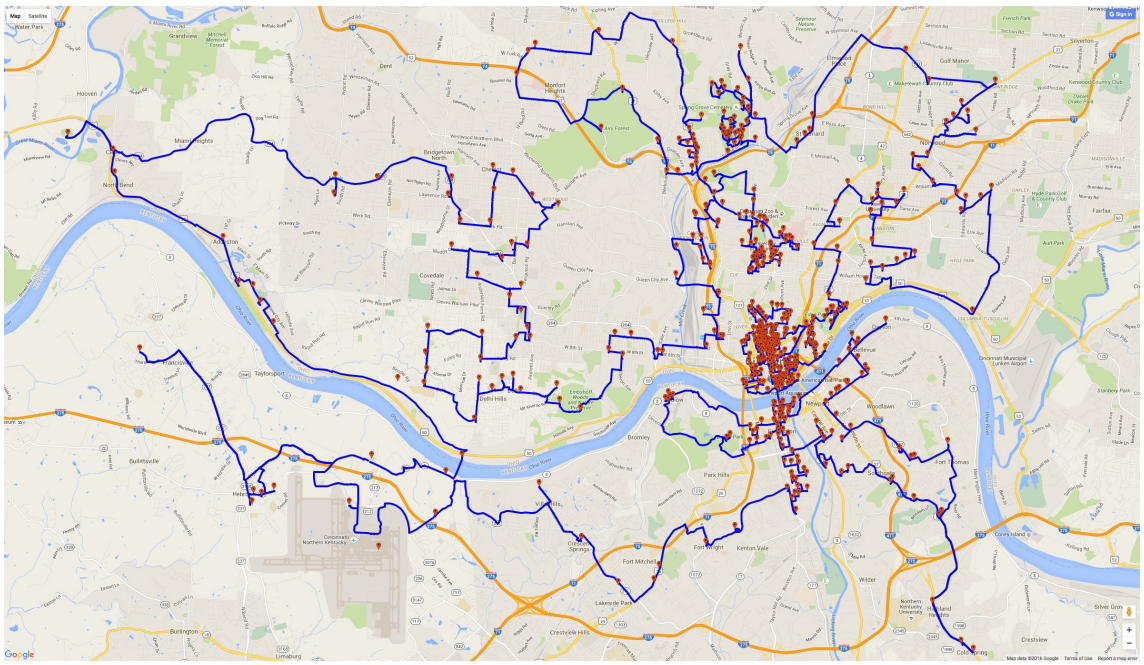


Figura 1: Passeio nos PokeStops e ginásios de Cincinnati.

O nome "problema do caixeiro viajante" não tem uma origem exata conhecida, mas sua associação com a pesquisa científica remonta às décadas de 1930 e 1940. O nome é apropriado, dada a natureza do problema de encontrar a rota mais curta entre cidades em uma região, evocando a figura icônica do caixeiro viajante na cultura americana. No início do século XX, havia uma grande presença de caixeiros viajantes nos Estados Unidos, como documentado no livro "100 Anos na Estrada: O Caixeiro Viajante na Cultura Americana" de Timothy Spears. Portanto, o nome reflete a relevância cultural e histórica desse problema de otimização de rotas [2].

Na década de 90, Gerhard Reinelt criou uma biblioteca valiosa que reúne um extenso conjunto de instâncias desafiadora do TSP. Essa biblioteca, chamada TSPLIB tem sido uma referência crucial para a pesquisa nessa área, contém mais de 100 casos de teste, abrangendo uma ampla gama de tamanhos, desde pequenos problemas com poucas cidades até desafios massivos com dezenas de milhares de cidades [2].

Este artigo apresenta uma visão abrangente dos elementos essenciais para a com-

preensão sólida do problema do caixeiro viajante (TSP). Na seção 2, são abordados os conceitos fundamentais dos grafos e complexidade computacional, destacando a relevância para o TSP. Na seção 3, aprofunda-se a análise sobre o próprio TSP, explorando seus conceitos e desafios. Na seção 4, concentra-se a atenção em algoritmos, com destaque para a técnica de Lin e Kernighan e o algoritmo de inserção de vértices. Por fim, na seção 5, encerra-se a exploração, trazendo uma conclusão que sintetiza os principais pontos e perspectivas para o estudo do TSP. Este artigo tem como objetivo proporcionar uma base sólida para aqueles que desejam mergulhar mais a fundo no fascinante mundo do problema do caixeiro viajante.

2 Definições básicas

Um **grafo** G é composto por dois conjuntos finitos fundamentais: o conjunto de vértices, representado por $V(G)$, e o conjunto de arestas, denotado como $A(G)$. Cada **aresta** em $G = (V(G), A(G)) = (V, A)$ estabelece uma relação de incidência entre um par de vértices, conectando os vértices dois a dois. É dito que dois vértices de um grafo são **adjacentes** se existe uma aresta ligando um ao outro. Assim também, duas arestas são **adjacentes** se incidem no mesmo vértice. O número de arestas que incidem em um vértice, indica o **grau do vértice**[4].

Dentro do estudo de grafos, é importante compreender alguns conceitos adicionais. Um **grafo completo**, por exemplo, é um grafo no qual cada par de vértices é conectado por uma aresta. Em outras palavras, cada par de vértices no grafo são dois a dois adjacentes. Se um grafo G é **ponderado** então existem pesos atribuídos as suas arestas ou vértices, ou seja, para cada aresta ou vértice do grafo associa-se um valor numérico[4].

Percorrer um **passeio** em um grafo G é percorrer uma sequência de vértices adjacentes, podendo ou não visitar os vértices mais de uma vez. Se o passeio visitar cada vértice uma única vez, ele é denominado **caminho**. Um grafo com **ciclo** é um grafo no qual é possível encontrar um **caminho fechado**, ou seja, uma sequência de vértices e arestas que começa e termina no mesmo vértice. Os grafos **acíclicos** são grafos nos quais não é possível encontrar ciclos. Quando um caminho passa por todos os vértices de um grafo G, ele é chamado de **caminho Hamiltoniano**. Um **ciclo Hamiltoniano** é essencialmente um caminho Hamiltoniano fechado, ou seja, começa e termina no mesmo vértice. Qualquer grafo que contém um ciclo Hamiltoniano é denominado **grafo Hamiltoniano**[4].

Aqui, é importante notar que decidir se um grafo é Hamiltoniano é uma tarefa NP-Completa. Isso significa que encontrar um ciclo Hamiltoniano em um grafo é uma tarefa que, até o momento, não possui um algoritmo eficiente conhecido que funcione para todos os casos. A classificação **NP-Completa** indica que essa tarefa pertence à classe de problemas NP (Problemas de Decisão Não-Determinísticos Polinomiais) e é **NP-difícil**, isto é, é pelo menos tão difícil quanto o problema mais difícil dentro da classe NP. A possibilidade de resolver ou de não resolver eficientemente um problema NP-Completo em tempo polinomial ainda é uma questão em aberto na teoria da complexidade computacional. A demonstração de que NP é igual ou diferente de P, classe de problemas polinomiais, vale um prêmio de um milhão de dólares, oferecido pelo instituto Clay. Problemas da **classe NP** são problemas

polinomialmente verificáveis, isto é, dada uma possível solução de uma instância do problema, é possível verificar em tempo polinomial se essa solução é válida[8].

Para demonstrar que um problema, X, pertence à classe NP-difícil, é necessário selecionar um outro problema, Y, que seja conhecido por ser NP-difícil. Posteriormente, deve-se demonstrar que Y não é mais difícil que X. A chave para estabelecer essa relação entre Y e X está na definição de redução. Uma **redução polinomial** é uma técnica utilizada na teoria da complexidade para demonstrar que um problema é, no mínimo, tão difícil quanto outro. Ela envolve a transformação de uma instância do problema Y em uma instância equivalente do problema X, de forma que essa transformação seja eficiente, ou seja, que possa ser realizada em tempo polinomial em relação ao tamanho da entrada de Y. Além disso, a transformação deve preservar a resposta da instância original, ou seja, a solução Sy de Y deve ser uma solução de X se e somente se a solução Sx de X for uma solução de Y. Se Y pode ser reduzido a X, então Y é um subproblema de X, isso significa que, se for possível resolver o problema X de maneira eficiente, também é possível resolver o problema Y da mesma forma, estabelecendo a relação de dificuldade entre eles [8].

No contexto deste trabalho, será explorado a redução do problema do ciclo Hamiltoniano para o problema do caixeiro viajante, demonstrando como a solução de um problema pode ser transformada em outra, proporcionando uma compreensão mais profunda das dificuldades intrínsecas desse problema.

3 Definição do problema

Imagine um conjunto de cidades, o objetivo do problema do caixeiro viajante é visitar cada cidade exatamente uma vez e, em seguida, retornar à cidade de origem, fazendo isso com o menor custo possível. Este problema é amplamente reconhecido e um dos mais estudados na análise combinatória. Apesar da sua formulação aparentemente simples, ele intriga a comunidade devido à sua complexidade computacional. A incerteza reside no fato de não saber se é possível encontrar uma solução eficiente em tempo polinomial para o TSP. Ele é um exemplo de problema NP-difícil, isso implica que encontrar uma solução ótima para instâncias maiores do problema pode requerer um tempo exponencial, tornando-o um dos problemas mais desafiadores da teoria da computação [8].

O problema do ciclo Hamiltoniano é outro desafio bem conhecido na teoria da complexidade e é classificado como NP-completo. Uma relação importante entre esses dois problemas é que o ciclo Hamiltoniano pode ser considerado um subproblema do TSP. Para entender essa relação, é essencial notar que o TSP envolve encontrar um ciclo Hamiltoniano que minimize o custo total desse ciclo em um grafo ponderado nas arestas [8].

Uma observação interessante é que é possível realizar uma redução polinomial do problema do ciclo Hamiltoniano para o problema do caixeiro viajante. Isso significa que uma instância do problema do ciclo Hamiltoniano pode ser transformada em uma instância equivalente do problema do caixeiro viajante em tempo polinomial. De fato, seja $G = (V, A)$ uma instância do problema do ciclo Hamiltoniano. Em seguida construa uma instância do problema do caixeiro viajante, seja $G' = (V', A')$ um grafo completo tal que V' seja igual a V. Para cada aresta em A' atribua custo

1 se essa aresta está em A e custo 2 se não pertence a A. Observe que a construção do grafo completo pode ser feita em tempo polinomial em relação ao número de vértices do grafo G. Isso ocorre porque o número de arestas que é preciso adicionar a G' é limitado superiormente pelo quadrado do número de vértices. Portanto, o tempo de construção desse grafo é $O(n^2)$, em que n é o número de vértices [8].

Se G possui um ciclo Hamiltoniano, então G' possui uma solução para o TSP de custo mínimo n. De fato, considere C um ciclo Hamiltoniano em G. Observe que o custo do ciclo C em G' é igual ao número de arestas do ciclo, que é igual ao número de vértices do ciclo, que é igual a n. E como a aresta de menor custo em G' é 1, então C é uma solução para o TSP em G' de custo mínimo n. Da mesma forma, se G' possui uma solução para o TSP de custo mínimo n, então G possui um ciclo Hamiltoniano. De fato, seja S uma solução para o TSP em G' de custo mínimo n. Como S é uma solução para o TSP, então cada vértice de V é visitado exatamente uma vez em S. Portanto, S é um ciclo Hamiltoniano em G. A partir das duas afirmações provadas acima, pode-se concluir que G possui um ciclo Hamiltoniano se e somente se G' possui uma solução para o TSP de custo mínimo n. Portanto, existe uma redução polinomial do problema do ciclo Hamiltoniano para o problema do caixeiro viajante [8].

O problema do caixeiro viajante recebe uma atenção significativa na análise combinatória. Dado que encontrar uma solução ótima para todas as instâncias do problema em tempo polinomial é uma questão em aberto, a pesquisa se concentra em desenvolver heurísticas, ou seja, métodos aproximados que possam fornecer soluções aceitáveis em tempo razoável. Essas heurísticas são estratégias inteligentes que, embora não garantam a solução ideal, muitas vezes produzem resultados próximos o suficiente para aplicações práticas.

4 Algoritmos

Uma ideia intuitiva para resolver o problema do caixeiro viajante é calcular o custo de todos os circuitos Hamiltonianos possíveis e escolher o de menor custo. No entanto, essa abordagem é inviável para grandes conjuntos de cidades devido à explosão combinatória de circuitos a serem avaliados. De fato, a complexidade computacional dessa estratégia é $O(n!)$, em que n é o número de cidades que se deseja visitar. Pois para calcular todos os circuitos possíveis, seria necessário gerar todas as permutações das cidades, o que leva a $\frac{(n-1)!}{2}$ combinações diferentes. É dividido por dois, pois os custos de viagem não dependem da direção que é tomada no passeio. Isso significa que para um número bem pequeno de cidades, o método pode ser viável, mas à medida que o número de cidades aumenta, o número de circuitos a serem avaliados cresce de forma astronômica. Por exemplo, para 10 cidades, teríaamos que calcular 181.440 circuitos, e para 20 cidades, esse número aumentaria para 60.822.550.204.416.000 [5].

Um bom algoritmo conhecido que busca encontrar a solução ótima para o TSP é o algoritmo de Bellman-Held-Karp que possui abordagem de programação dinâmica. O algoritmo de Bellman-Held-Karp é capaz de resolver o TSP para um número relativamente pequeno de cidades de maneira exata, garantindo que a solução encontrada seja ótima. Sua complexidade computacional é $O(n^2 2^n)$, onde n é o número de ci-

dades que o viajante deve visitar. Este algoritmo é melhor que o algoritmo anterior que usa uma abordagem de busca exaustiva. Isto o torna bastante eficiente para números de cidades moderados. No entanto, à medida que o número de cidades aumenta, é necessário procurar outras abordagens, como algoritmos heurísticos e aproximados [2].

A fim de solucionar diversas instâncias do TSP, David Applegate, Robert E. Bixby, Vašek Chvátal, e William J. Cook desenvolveram um software que implementa um dos algoritmos mais eficazes para resolver o problema do caixeiro viajante. O "Concorde" é escrito em C e o seu código possui mais de 130.000 linhas. O gráfico abaixo mostra algumas instâncias do TSPLIB que foram resolvidas pelo Concorde. Observe como aumentou o número de cidades das instâncias resolvidas ao longo dos anos. Todas as 110 instâncias do TSPLIB já foram resolvidas pelo software, a maior delas com 85.900 cidades. Os algoritmos implementados no Concorde são baseados em técnicas de programação linear e heurísticas para melhorar a eficiência na busca de soluções ótimas para o TSP [3].

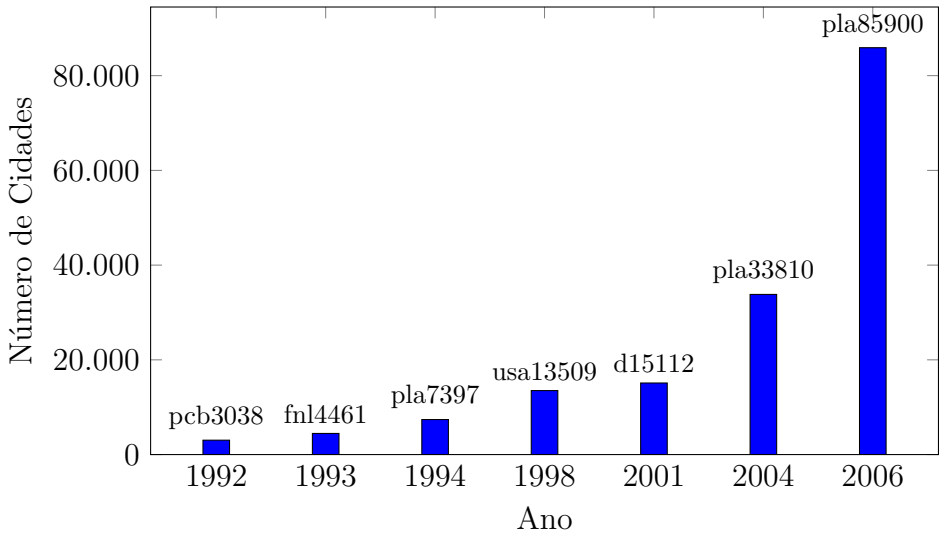


Figura 2: Soluções de instâncias do TSPLIB usando o Concorde ao longo dos anos.

4.1 Técnica de Lin e Kernighan

A principal técnica heurística implementada pelo Concorde é uma versão melhorada do algoritmo Lin e Kernighan, que é um algoritmo de otimização específico para o TSP. O algoritmo de Lin e Kernighan é um dos mais eficientes para o problema do caixeiro viajante. Ele é capaz de encontrar soluções ótimas ou quase ótimas para problemas de tamanho moderado a grande[6]. Essa heurística produz soluções ótimas com uma alta frequência e produziu soluções ótimas para todos os problemas resolvidos, ou seja problemas que a solução ótima é conhecida, inclusive o problema do TSPLIB de 85.900 cidades [7].

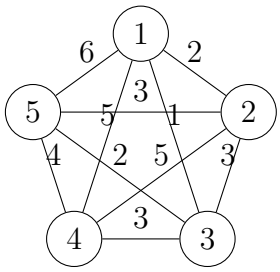
O algoritmo de Lin e Kernighan é um exemplo clássico de heurísticas de substituição, conhecidas também como heurística k-ótima. São estratégias de melhoria que se originam a partir de um ciclo Hamiltoniano inicial. A ideia consiste em

substituir k arestas em busca do menor custo. As heurísticas de substituição, especialmente as 2-Substituições e 3-Substituições, possuem relatos de bom desempenho. Vale ressaltar que a abordagem de Lin e Kernighan, apresentada abaixo, também se baseia em técnica de procedimento devido à sua natureza estruturada e aos passos bem definidos que orientam o processo de otimização [4].

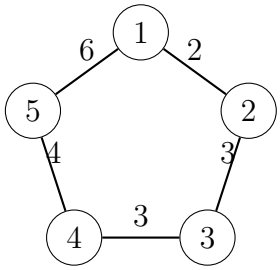
Algoritmo 1 Lin e Kernighan

1: **Entrada:** $G = (V, A)$
2: Construir e calcular o custo c de um ciclo Hamiltoniano
3: $c_{\text{melhor}} \leftarrow c$ ▷ Guarda o menor custo do ciclo
4: Selecionar uma aresta para remoção, com base em um critério específico, a fim de criar um caminho hamiltoniano, H. ▷ Uma possível estratégia seria remover a aresta mais cara do ciclo.
5: Criar duas listas vazias: Arestas removidas, L_{Sai}, e arestas inseridas, L_{Entra}
6: Incluir a última aresta excluída em L_{Sai}
7: **Se** existir uma aresta w em que um dos vértices é terminal e o outro é interno h de H, tal que $w \notin L_{\text{Sai}}$ e $w \notin L_{\text{Entra}}$ **então** $H \leftarrow H \cup \{w\}$, formando um ciclo em H e $L_{\text{Entra}} \leftarrow L_{\text{Entra}} \cup \{w\}$
8: **Senão** ir para o passo 15
9: Calcular o ganho, g, pela diferença da soma dos pesos das arestas em L_{Sai} e em L_{Entra}
10: **Se** existir uma aresta a adjacente a h tal que $a \notin L_{\text{Entra}}$ e a faz parte **então** remove a
11: **Senão** ir para o passo 15
12: Calcular o custo do ciclo Hamiltoniano induzido, c_{ind}, por H e a aresta que liga os vértices terminais
13: **Se** $c_{\text{ind}} < c_{\text{melhor}}$ **então** $c_{\text{melhor}} \leftarrow c_{\text{ind}}$ e salvar o ciclo Hamiltoniano correspondente a c_{ind}
14: **Se** $g > 0$ **então** voltar ao passo 6
15: **Se** $c_{\text{melhor}} < c$ **então** voltar ao passo 4, considerando o ciclo Hamiltoniano correspondente a c_{melhor} como a nova solução inicial e $c \leftarrow c_{\text{melhor}}$
16: **Senão saída:** H

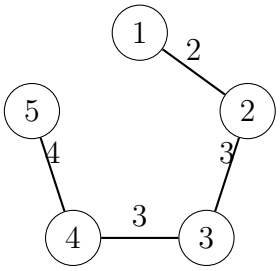
Observe que a proposta desse algoritmo apresenta uma estrutura que deixa espaço para tomadas de decisão importantes no desenvolvimento da solução. Normalmente é utilizada combinada a outro algoritmo, por exemplo, o vizinho mais próximo para construir um ciclo Hamiltoniano. O passo 4 deixa em aberto qual o critério de seleção da aresta a ser removida, permitindo uma série de adaptações e personalizações [4]. E o seu tempo cresce aproximadamente $O(n^{2.2})$, em que n é o número de vértices [9]. As Figuras 3(2)-(17) mostram um exemplo usando o algoritmo acima.



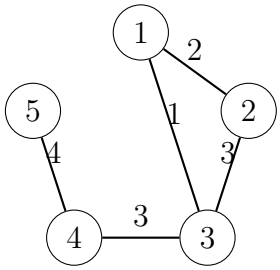
(1) Grafo G



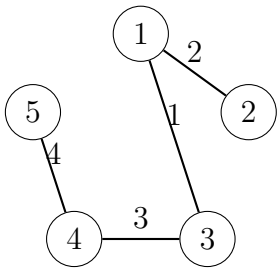
(2) c ← 18
Salva melhor ciclo
$(V_1V_2V_3V_4V_5V_1)$



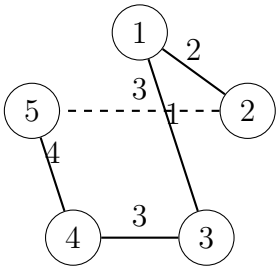
(3) $C_{melhor} \leftarrow 18$
Remove aresta V_1V_5



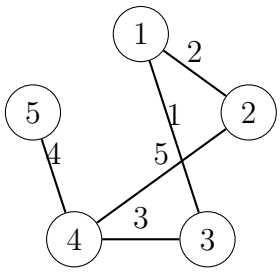
(4) $L_{sai} \leftarrow \oslash$
$L_{entra} \leftarrow \oslash$
$L_{sai} \leftarrow L_{sai} \cup \{V_1V_5\}$
$L_{entra} \leftarrow L_{entra} \cup \{V_1V_3\}$



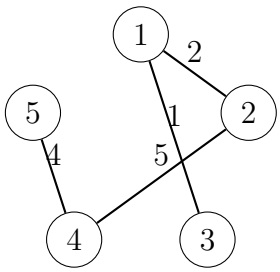
(5) g = 6 - 1 = 5
Remove aresta V_2V_3



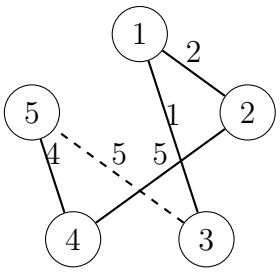
(6) c_{ind}= 13 $c_{melhor} \leftarrow 13$
Salva $(V_1V_2V_5V_4V_3V_1)$
g > 0 retorna ao passo 6
$L_{sai} \leftarrow L_{sai} \cup \{V_2V_3\}$



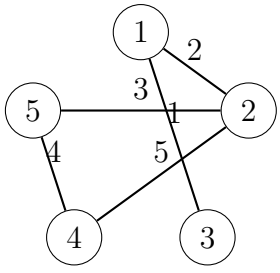
(7) $L_{entra} \leftarrow L_{entra} \cup \{V_2V_4\}$
$L_{sai} = \{V_1V_5, V_2V_3\}$
$L_{entra} = \{V_1V_3, V_2V_4\}$



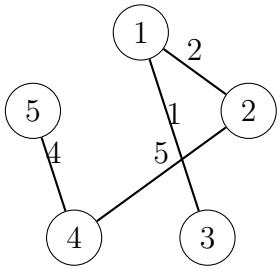
(8) g = 9 - 6 = 3
Remove aresta V_3V_4



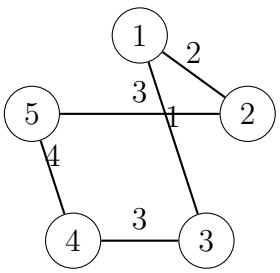
(9) c_{ind}= 17
g > 0 retorna ao passo 6
$L_{sai} \leftarrow L_{sai} \cup \{V_3V_4\}$



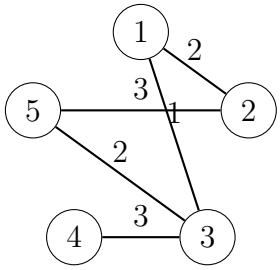
(10) $L_{entra} \leftarrow L_{entra} \cup \{V_2V_5\}$
$L_{sai} = \{V_1V_5, V_2V_3, V_3V_4\}$
$L_{entra} = \{V_1V_3, V_2V_4, V_2V_5\}$



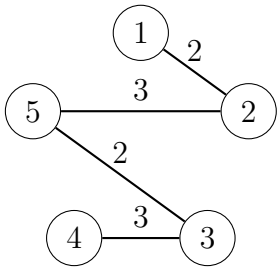
(11) g = 12 - 9 = 3
ir para o passo 15



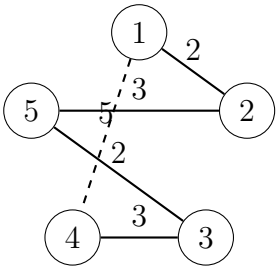
(12) Restaura melhor ciclo
ir para o passo 4
c ← 13



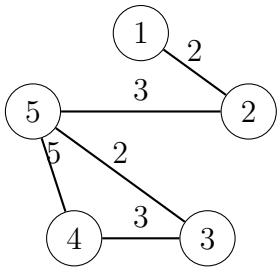
(13) $L_{sai} \leftarrow \oslash$
$L_{entra} \leftarrow \oslash$
$L_{sai} \leftarrow L_{sai} \cup \{V_4V_5\}$
$L_{entra} \leftarrow L_{entra} \cup \{V_3V_5\}$



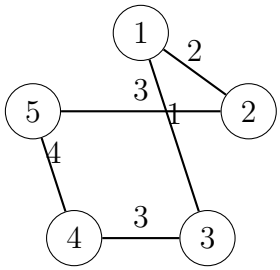
(14) g = 4 - 2 = 2
Remove a aresta V_1V_3



(15) $c_{ind} \leftarrow 15$
ir para o passo 6
$L_{sai} \leftarrow L_{sai} \cup \{V_1V_3\}$



(16) $L_{entra} \leftarrow L_{entra} \cup \{V_4V_5\}$
g = 5 - 7 = -2
ir para o passo 15



(17) Solução Final

Figura 3: Aplicação da heurística L&K.

4.2 Heurística de Inserção de Vértice

As heurísticas de inserção de vértice são uma classe de técnicas utilizadas para resolver o problema do caixeiro viajante, visando otimizar a rota. Essas heurísticas se concentram em adicionar novos vértices a uma solução parcial do TSP, com o objetivo de formar um ciclo Hamiltoniano que represente a rota de menor custo. O processo de inserção de um vértice envolve a escolha do vértice a ser inserido na solução parcial e a determinação da posição desse novo vértice em relação ao ciclo inicial. A escolha do vértice a ser inserido e a estratégia de inserção podem variar de acordo com o algoritmo utilizado e os critérios de otimização específicos em questão [4].

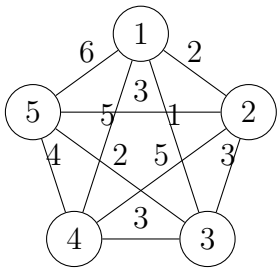
O algoritmo em destaque utiliza como critério principal para suas decisões a minimização da diferença entre o custo de entrada e saída das arestas avaliadas durante o processo de inserção. Ao inserir um novo vértice entre os vértices i e $i+1$, a ação implica na remoção da aresta que conecta esses vértices [4].

A complexidade deste algoritmo deve-se ao fato que o laço de repetição continua até que H se torne um ciclo Hamiltoniano. Em um pior caso, isso pode levar até n iterações, onde n é o número de vértices no grafo. Dentro do laço, o algoritmo procura um vértice k que não está em H. Portanto para essas duas operações tem-se uma complexidade de tempo $O(n^2)$. Contudo ao procurar um vértice $k \notin H$, examina-se as arestas incidentes em k, no pior caso pode levar $O(m)$ operações, em que m é o número do vértice de maior grau no grafo. Portanto a complexidade algo-

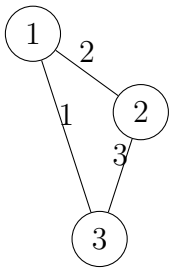
Algoritmo 2 Insere Vértice

1: **Entrada:** $G = (V, A)$
2: Começar por um ciclo de vértices H
3: **Enquanto** H não for um ciclo Hamiltoniano **fazer**
4: Selecionar um vértice $k \notin H$ para inserir entre os vértices i e $i+1$ que H minimize $c_{ik} + c_{ki+1} - c_{ii+1}$
5: Adicionar o vértice k em H entre os vértices i e $i+1$
6: **Saída:** H

ritmíca é de $O(mn^2)$. As Figuras 4(2)-(6) mostram um exemplo usando o algoritmo acima [4].



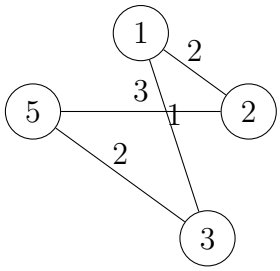
(1) Grafo G



(2) Ciclo H

$$c_{v_1v_4} + c_{v_4v_2} - c_{v_1v_2} = 8$$
$$c_{v_1v_4} + c_{v_4v_3} - c_{v_1v_3} = 7$$
$$c_{v_1v_5} + c_{v_5v_2} - c_{v_1v_2} = 7$$
$$c_{v_1v_5} + c_{v_5v_3} - c_{v_1v_3} = 7$$
$$c_{v_2v_4} + c_{v_4v_3} - c_{v_2v_3} = 5$$
$$c_{v_2v_5} + c_{v_5v_3} - c_{v_2v_3} = 2$$

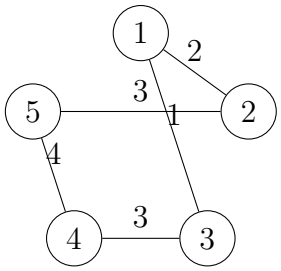
(3) Cálculos do passo 4



(4) Vértice 5 inserido

$$c_{v_1v_4} + c_{v_4v_2} - c_{v_1v_2} = 8$$
$$c_{v_1v_4} + c_{v_4v_3} - c_{v_1v_3} = 7$$
$$c_{v_2v_4} + c_{v_4v_5} - c_{v_2v_5} = 6$$
$$c_{v_3v_4} + c_{v_4v_5} - c_{v_3v_5} = 5$$

(5) Cálculos do passo 4



(6) Solução final

Figura 4: Aplicação da Heurística Insere Vértice

5 Conclusão

Em síntese, este artigo fundamentado em revisão bibliográfica, permitiu-nos adentrar no fascinante universo do problema do caixeiro viajante, um dos problemas mais importantes da teoria da computação. O TSP consiste em encontrar um ciclo Hamiltoniano de peso mínimo em um grafo com pesos nas arestas.

Ao investigar os diferentes algoritmos que podem ser usados para resolver o TSP pudemos testemunhar a evolução das técnicas de resolução ao longo do tempo. Os algoritmos exatos, como o algoritmo de Bellman-Held-Karp, têm complexidade computacional exponencial e, portanto, só podem ser aplicados a instâncias pequenas do problema. Os algoritmos heurísticos, por outro lado, têm complexidade computacional polinomial e são capazes de fornecer soluções aproximadas em tempo hábil.

Dada a complexidade computacional do TSP, nossa pesquisa enfatizou a importância dos algoritmos heurísticos, que fornecem soluções aproximadas em tempo hábil. Um exemplo notável é o algoritmo modificado de Lin e Kernighan, que, embora heurístico, muitas vezes apresenta soluções ótimas ou muito próximas a elas. Além disso, exploramos o algoritmo de inserção de vértices, que, com sua complexidade algorítmica de $O(mn^2)$, se mostrou eficaz na entrega de resultados satisfatórios.

Referências

[1] Alter, M.; Briz, A. Here are the Local Spots to Visit to Get the Most Out of Pokémon Go. Disponível em: `https://www.wcpo.com/news/local-news/hamilton-county/cincinnati/here-are-the-local-spots-to-visit-to-get-the-most-out-of-pokmon-go-`. Acesso em: 02 nov. 2023.

[2] Applegate, D. et al. The Traveling Salesman Problem: A Computational Study. New Jersey: Princeton University Press, 2006. ISBN 978-0-691-12993.

[3] Concorde TSP Solver. University of Waterloo. Disponível em: `https://www.math.uwaterloo.ca/tsp/concorde/index.html`. Acesso em: 02 nov. 2023.

[4] Goldbard, M.; Goldbarg, E. Grafos: Conceitos, algoritmos e aplicações. Rio de Janeiro: Elsevier Editora Ltda, 2012. ISBN 978-85-352-5716-8.

[5] Hillier, F. S.; Lieberman, G. J. Introduction to operations research. 10. ed. New York: McGraw-Hill Education, 2015. ISBN 978-0-07-352345-3.

[6] Helsgaun, K. An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. European Journal of Operational Research. Denmark, Vol. 126, pg 106-130, outubro, 2000.

[7] Helsgaun, K. An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems. Roskilde University, Denmark. dezembro, 2017.

[8] Kleinberg, J.; Tardos, É. Algorithm Design. 1. ed. Pearson Education, 2006. ISBN 978-0-321-29535-8.

[9] Lin, S.; Kernighan, B. W. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. Operations Research, Vol. 21, pg 498-516, abril, 1973.

[10] Pokémon Go and the Traveling Salesman Problem. University of Waterloo. Disponível em: `https://www.math.uwaterloo.ca/tsp/poke/index.html`. Acesso em: 02 nov. 2023.