57º SBPO
SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
Gramado • RS
DE 5 A 9 DE OUTUBRO DE 2025

Iterated Local Search aprimorada com Buscas em Blocos de Coordenadas para o Problema do Caixeiro Viajante Euclidiano com Vizinhanças Poligonais

Ernesto G. Birgin
Departamento de Ciência da Computação, Instituto de Matemática e Estatística, Universidade de São Paulo
Rua do Matão, 1010, Cidade Universitária, 05508-090, São Paulo, SP, Brasil
egbirgin@ime.usp.br

Paula C. R. Ertel
Departamento de Matemática Aplicada, Instituto de Matemática e Estatística, Universidade de São Paulo
Rua do Matão, 1010, Cidade Universitária, 05508-090, São Paulo, SP, Brasil
paulaertel@ime.usp.br

RESUMO

O problema do Caixeiro Viajante com Vizinhanças (TSPN) é uma generalização do famoso Problema do Caixeiro Viajante. No TSPN, dado um conjunto de regiões do espaço euclidiano, busca-se encontrar uma ordenação dessas regiões e um ponto de visita em cada uma, para minimizar o comprimento do tour que as visita uma única vez e retorna ao ponto inicial. A maior parte da literatura sobre o TSPN foca no caso em que as regiões são discos no plano euclidiano. Neste trabalho, estamos interessados em resolver o Problema do Caixeiro Viajante Euclidiano com vizinhanças poligonais arbitrárias. Para tanto, propomos uma adaptação de uma meta-heurística do tipo *Iterated Local Search* (ILS). A ILS proposta combina diferentes estruturas de vizinhança, amplamente utilizadas na literatura do TSP, para gerar as permutações com um método de busca em bloco de coordenadas para determinar os pontos de visita. Experimentos numéricos ilustram o desempenho do método proposto.

PALAVRAS CHAVE. Problema do Caixeiro Viajante com Vizinhanças, Meta-heurísticas, Busca em Bloco de Coordenadas.

Tópicos (Meta-heurística ILS para o TSPN com vizinhanças poligonais)

ABSTRACT

The Traveling Salesman Problem with Neighborhoods (TSPN) is a generalization of the famous Traveling Salesman Problem. In the TSPN, given a set of neighborhoods of the Euclidean space, the goal is to find an order of these neighborhoods and a visit point in each neighborhood to minimize the length of the tour that visits them only once and returns to the starting point. Most of the literature on TSPN focuses on the case where the regions are disks. In this work, we are interested in solving the Euclidean Traveling Salesman Problem with arbitrary polygonal neighborhoods. To this end, we propose an Iterated Local Search (ILS) metaheuristic. The proposed ILS combines different neighborhood structures, widely used in the TSP literature, to generate the permutations with a block coordinate descent method to determine the visit points. Numerical experiments illustrate the performance of the proposed method.

KEYWORDS. Traveling Salesman Problem with Neighborhoods. Metaheuristics. Block Coordinate Descent.

Paper topics (ILS metaheuristic for TSPN with polygonal neighborhoods)

DOI: 10.59254/sbpo-2025-212601
https://proceedings.science/p/212601?lang=pt-br

1. Introdução

Em sua versão clássica, o Problema do Caixeiro Viajante (TSP, do inglês *Traveling Salesman Problem*) consiste na seguinte questão: dado um conjunto de m cidades, com a distância de se viajar de uma cidade i para qualquer outra cidade j conhecida, deseja-se encontrar um tour (ordenação das cidades) de menor comprimento que passe por todas as m cidades uma única vez e retorne à cidade de partida [Applegate et al., 2006]. No caso em que cada cidade corresponde a um ponto no plano euclidiano e a distância entre cada par de cidades é dada pela distância euclidiana entre seus pontos, obtém-se o TSP Euclidiano. Apesar de simples de descrever, o TSP pertence à classe de problemas NP-difícil [Cormen et al., 2012] e está entre os problemas mais famosos da área de Otimização Combinatória e Pesquisa Operacional. Muito da popularidade deste problema se deve a sua aplicabilidade em diversas áreas, ao grande número de variantes que o problema apresenta [Gutin e Punnen, 2006] e ao interesse pelo desenvolvimento de métodos heurísticos para resolvê-lo [Laporte, 2010].

Assim, o Problema do Caixeiro Viajante foi e continua sendo um problema intensivamente investigado. No entanto, algumas variantes do TSP ainda permanecem pouco exploradas. É o caso do Problema do Caixeiro Viajante com Vizinhanças ou TSPN, do inglês *Traveling Salesman Problem with Neighborhoods*. No TSPN, dado um conjunto de m regiões arbitrárias do espaço euclidiano, busca-se encontrar uma ordenação dessas regiões e um ponto de visita em cada uma, com o objetivo de minimizar o comprimento do tour que as visita uma única vez e retorna ao ponto inicial. O TSPN também é um problema pertencente à classe NP-difícil, uma vez que ele se resume ao TSP no caso em que as regiões são dadas por um único ponto.

Em alguns artigos da literatura, o TSPN é chamado de *Close-Enough Traveling Salesman Problem* (CETSP) [Mennell, 2009]. No entanto, no CETSP as vizinhanças são definidas como discos do plano euclidiano e o problema pode ser interpretado como um TSP no qual cada cidade corresponde ao centro do círculo que representa sua região, e o objetivo é passar não exatamente por cada cidade, mas no máximo a uma distância pré-definida de cada uma delas, por isso o nome *Close-Enough*. [Behdani e Smith, 2014] formulam o CETSP como um problema de programação não-linear inteira mista e propõem métodos para gerar limites inferior e superior para o custo de um tour ótimo do CETSP. [Coutinho et al., 2016] propuseram, em seu trabalho, um algoritmo de *Branch-and-Bound* que utiliza programação cônica de segunda ordem para formular o subproblema de determinar os pontos de visita em cada círculo. Um exemplo de aplicação do CETSP é no uso de tecnologias de identificação por radiofrequência para leitura automatizada de medidores. [Shuttleworth et al., 2008] e [Dong et al., 2007] abordam esta aplicação utilizando o CETSP. Outros exemplos de aplicação do CETSP que aparecem na literatura são na localização de incêndios florestais por veículos aéreos não tripulados [Poikonen et al., 2017] e no diagnóstico de painéis solares [Placido et al., 2022].

[Arkin e Hassin, 1994] foram os primeiros a introduzir o TSPN com vizinhanças mais gerais, porém sob o nome de *Geometric Covering Salesman Problem*. Em seu trabalho, os autores fornecem algoritmos de aproximação para o caso em que as vizinhanças possuem formatos específicos, tais como segmentos de reta paralelos e de mesmo tamanho, ou regiões que são translações de um mesmo tipo de polígono convexo. Tais algoritmos consistem em escolher um ponto para representar cada vizinhança e, então, aplicar algum algoritmo de aproximação do TSP clássico. [Yang et al., 2018], por sua vez, propuseram um algoritmo para o TSPN que combina *Particle Swarm Optimization* (PSO) para calcular os pontos de visita em cada região com um algoritmo genético para gerar as permutações. Entretanto, neste artigo, apesar de o método ser proposto para o TSPN com vizinhanças arbitrárias, somente uma instância com 21 vizinhanças de diferentes formatos (segmentos de reta, discos, anéis, setores circulares e alguns polígonos) foi testada, e o

DOI: 10.59254/sbpo-2025-212601
https://proceedings.science/p/212601?lang=pt-br

restante dos experimentos computacionais foram realizados em instâncias do CETSP compostas por discos do plano. Uma aplicação do TSPN com vizinhanças poligonais, apresentada no artigo de [Dror, 1999], é o problema de cortar formas poligonais em placas de metal ou vidro. Neste problema, dado um conjunto de polígonos desenhados em uma placa bidimensional, deseja-se determinar uma sequência ótima de corte destes polígonos tal que o ponto onde o tour entra e sai de cada polígono seja o mesmo.

Portanto, uma vez que pouco da literatura do TSPN se dedica ao problema que engloba regiões mais diversas, o objetivo deste trabalho é desenvolver e testar um algoritmo meta-heurístico para o Problema do Caixeiro Viajante Euclidiano com vizinhanças poligonais arbitrárias. Nosso interesse é propor um algoritmo baseado na meta-heurística *Iterated Local Search* (ILS) que combine heurísticas já desenvolvidas para o TSP para gerar as permutações das regiões com um método de otimização contínua de Busca em Bloco de Coordenadas [Birgin e Martínez, 2021] para calcular os pontos de visita, algo ainda não experimentado na literatura do TSPN. A escolha da meta-heurística ILS se deve aos bons resultados apresentados no trabalho de [Schmidt e Irnich, 2022] para o TSP generalizado (GTSP [Fischetti et al., 1997]), uma variante do TSP muito similar ao TSPN. No GTSP, cada cidade é composta por um conjunto finito de pontos e o objetivo é determinar um tour de custo mínimo que visite exatamente um ponto de cada cidade. A escolha de um método de busca em coordenadas para calcular os pontos de visita foi inspirado no artigo de [Birgin e Martínez, 2021]. Neste trabalho, os autores utilizam o método para resolver o problema de determinar uma rota de custo mínimo que visite todos os bairros da cidade de São Paulo, Brasil, em um único ponto.

O presente trabalho está organizado da seguinte forma. A Seção 2 introduz um modelo de programação linear inteira mista com restrições quadráticas para o TSPN Euclidiano com vizinhanças poligonais. A Seção 3 descreve brevemente o método de busca em bloco de coordenadas proposto para determinar os pontos de visita. O algoritmo meta-heurístico é apresentado na Seção 4. A Seção 5 introduz instâncias para o TSPN com polígonos e a Seção 6 é dedicada aos experimentos computacionais. A Seção 7 conclui este trabalho com algumas considerações finais e propostas de projetos futuros.

2. Modelo para o TSPN

Nesta seção, introduzimos um modelo de programação linear inteira mista com restrições quadráticas para o TSPN Euclidiano com vizinhanças poligonais arbitrárias. Este modelo foi adaptado do modelo para o VRP desenvolvido no trabalho de [Matos, Em andamento]. A notação a seguir simplifica a apresentação do modelo. Vale salientar que, no caso em que o polígono P_i é não-convexo, ele é dividido em m_i polígonos convexos menores.

Parâmetros:

m: quantidade de polígonos;

m_i: quantidade de polígonos convexos $P_{i\ell}$ tais que $P_i = \bigcup_{\ell=1}^{m_i} P_{i\ell}$, $i = 1, \ldots, m$;

$q_{i\ell}$: quantidade de vértices do polígono $P_{i\ell}$;

$P_{i\ell} = \{x \in \mathbb{R}^2 | a_{i\ell k}x_1 + b_{i\ell k}x_2 + c_{i\ell k} \leq 0, k = 1, \ldots, q_{i\ell}\}$.

Variáveis de decisão:

x_{ij}: 1 se o polígono i precede o polígono j no tour e 0 caso contrário;

z_{ij}: comprimento do tour que vai do ponto visitado no polígono P_1 até o polígono j, passando pelo polígono i;

ξ_{ij}: representa a distância entre os pontos visitados nos polígonos i e j;

γ_{ij}: ξ_{ij} se $x_{ij} = 1$ e 0 se $x_{ij} = 0$;

p^i: ponto visitado no polígono P_i;

$w_{i\ell}$: 1 se $p^i \in P_{i\ell}$ e 0 caso contrário.

DOI: 10.59254/sbpo-2025-212601
https://proceedings.science/p/212601?lang=pt-br

O modelo para o TSPN é fornecido a seguir. Em (1) minimizamos o comprimento de um tour para o TSPN. As restrições (2) e (3) garantem que cada polígono é visitado uma única vez ao exigir que somente uma aresta entra e sai de cada polígono. As restrições (5)-(8) são as restrições de eliminação de subtours. Se o polígono i precede o polígono j no tour, a restrição (6), junto das restrições (5) e (7), garante que o comprimento do subtour entre o ponto visitado no polígono 1 até o ponto visitado no polígono sucessor de i, isto é, o polígono j, deve ser igual ao comprimento do subtour partindo do polígono 1 até o polígono i mais a distância de se viajar do polígono i até o polígono j. As restrições (9)-(13) são responsáveis por calcular a distância euclidiana entre os pontos de visita. As restrições (14)-(18) garantem que cada ponto de visita pertença a um único polígono. A restrição (14) é aplicada aos polígonos convexos, isto é, para os quais $m_i = 1$, e as restrições (15)-(17) para os polígonos não-convexos. Com a restrição (17) garantimos que, no caso em que o polígono é não-convexo, o ponto de visita deverá pertencer a somente um dos polígonos convexos que o integram. As constantes M_1 e M_2 das equações (7) e (11)-(12), respectivamente, podem ser definidas como a distância máxima entre todos os pares de polígonos. A constante M_3 da equação (15) pode ser definida como o comprimento da diagonal do menor quadrado que circunscreva a instância a ser resolvida.

$$\text{minimizar} \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m} \gamma_{ij} \tag{1}$$

sujeito a

$$\sum_{\substack{j=1 \\ j \neq i}}^{m} x_{ji} = 1 \qquad i = 1, \ldots, m \tag{2}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{m} x_{ij} = 1 \qquad i = 1, \ldots, m \tag{3}$$

$$x_{ij} \in \{0, 1\} \qquad i = 1, \ldots, m,\ j = 1, \ldots, m,\ i \neq j \tag{4}$$

$$z_{1j} = \gamma_{1j} \qquad j = 2, \ldots, m \tag{5}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{m} z_{ij} - \sum_{\substack{j=1 \\ j \neq i}}^{m} z_{ji} - \sum_{\substack{j=1 \\ j \neq i}}^{m} \gamma_{ij} = 0 \qquad i = 2, \ldots, m \tag{6}$$

$$z_{ij} \leq M_1 \, x_{ij} \qquad i = 1, \ldots, m,\ j = 1, \ldots, m,\ i \neq j \tag{7}$$

$$z_{ij} \geq 0 \qquad i = 1, \ldots, m,\ j = 1, \ldots, m,\ i \neq j \tag{8}$$

$$\xi_{ij}^2 \geq (p_x^i - p_x^j)^2 + (p_y^i - p_y^j)^2 \qquad i = 1, \ldots, m,\ j = 1, \ldots, m,\ i \neq j \tag{9}$$

$$\xi_{ij} \geq 0 \qquad i = 1, \ldots, m,\ j = 1, \ldots, m,\ i \neq j \tag{10}$$

$$\gamma_{ij} \leq x_{ij} M_2 \qquad i = 1, \ldots, m,\ j = 1, \ldots, m,\ i \neq j \tag{11}$$

$$\gamma_{ij} \geq \xi_{ij} - M_2(1 - x_{ij}) \qquad i = 1, \ldots, m,\ j = 1, \ldots, m,\ i \neq j \tag{12}$$

$$\gamma_{ij} \geq 0 \qquad i = 1, \ldots, m,\ j = 1, \ldots, m,\ i \neq j \tag{13}$$

$$a_{i\ell k} \, p_x^i + b_{i\ell k} \, p_y^i + c_{i\ell k} \leq 0 \qquad i = 1, \ldots, m \text{ tal que } m_i = 1,\ k = 1, \ldots, q_{i1} \tag{14}$$

$$a_{i\ell k} \, p_x^i + b_{i\ell k} \, p_y^i + c_{i\ell k} \leq M_3(1 - w_{i\ell}) \qquad i = 1, \ldots, m \text{ tal que } m_i > 1,\ \ell = 1, \ldots, m_i,\ k = 1, \ldots, q_{i\ell} \tag{15}$$

$$w_{i\ell} \in \{0, 1\} \qquad i = 1, \ldots, m \text{ tal que } m_i > 1,\ \ell = 1, \ldots, m_i \tag{16}$$

$$\sum_{\ell=1}^{m_i} w_{i\ell} = 1 \qquad i = 1, \ldots, m \text{ tal que } m_i > 1 \tag{17}$$

$$p^i \equiv (p_x^i, p_y^i)^T \in \mathbb{R}^2 \qquad i = 1, \ldots, m. \tag{18}$$

DOI: 10.59254/sbpo-2025-212601
https://proceedings.science/p/212601?lang=pt-br

3. Busca em coordenadas

Considere o seguinte problema de otimização contínua:

$$\text{minimizar} \quad f(x) \quad \text{sujeito a} \quad x \in \Omega, \tag{19}$$

com $f : \mathbb{R}^n \to \mathbb{R}$ uma função contínua e $\Omega \subseteq \mathbb{R}^n$. Métodos de Busca em Bloco de Coordenadas (BCD, do inglês *Block Coordinate Descent*) são métodos iterativos caracterizados por minimizar a função objetivo de (19) em relação a um bloco de coordenadas por vez, mantendo as demais fixadas. A vantagem deste tipo de método está na simplicidade de suas iterações, pois o subproblema resolvido a cada iteração tem dimensão menor que o problema original, sendo mais barato de resolver. Algumas aplicações relevantes e resultados teóricos de convergência podem ser encontrados no trabalho de [Wright, 2015].

Para o TSPN com vizinhanças poligonais, métodos deste tipo podem ser desenvolvidos para calcular os pontos de visita quando uma ordenação dos polígonos está fixada [Birgin e Martínez, 2021]. Assim, dado um conjunto de m polígonos possivelmente não-convexos, $P_1, \ldots, P_m$, o TSPN também pode ser formulado da seguinte forma:

$$\begin{aligned} \text{minimizar} \quad & f(x) := \|x^{i_m} - x^{i_1}\|_2 + \sum_{\nu=1}^{m-1} \|x^{i_\nu} - x^{i_{\nu+1}}\|_2 \\ \text{sujeito a} \quad & x^{i_\nu} \in P_{i_\nu} \ \text{ para } \ \nu = 1, \ldots, m, \end{aligned} \tag{20}$$

em que $\{i_1, i_2, \ldots, i_m\}$ é uma permutação de $\{1, 2, \ldots, m\}$, $x^{i_\nu} = (x_1^{i_\nu}, x_2^{i_\nu}) \in \mathbb{R}^2$ para $\nu = 1, \ldots, m$, $x^T = ((x^1)^T, \ldots, (x^m)^T) \in \mathbb{R}^{2m}$ e $\|\cdot\|_2$ corresponde à norma euclidiana. Fixada uma permutação $\{i_1, \ldots, i_m\}$ dos polígonos, propomos determinar uma solução para o problema (20) da seguinte forma: para cada $\nu = 1, \ldots, m$ é calculado um ponto $x^{i_\nu} \in P_{i_\nu}$ que minimiza a soma das distâncias até os pontos $x^{i_{\nu-1}}$ e $x^{i_{\nu+1}}$, cujos valores estão fixados. Em resumo, seguindo a ordem cíclica dada pela permutação que está fixada, a cada iteração k do método BCD escolhemos $\nu = \mathrm{mod}(k, m) + 1$ e resolvemos o seguinte problema:

$$\begin{aligned} \text{minimizar} \quad & \|p - x^{i_{\nu-1}}\|_2 + \|p - x^{i_{\nu+1}}\|_2 \\ \text{sujeito a} \quad & p \in P_{i_\nu}. \end{aligned} \tag{21}$$

A cada m iterações do método BCD dizemos que este completou um ciclo.

4. Meta-heurística

Nesta seção, propomos um algoritmo baseado na meta-heurística do tipo *Iterated Local Search* (ILS) combinada ao método de busca em bloco de coordenadas para resolver instâncias do TSPN em que as regiões são polígonos arbitrários do plano euclidiano. O algoritmo desenvolvido neste trabalho foi inspirado na ILS proposta no artigo de [Schmidt e Irnich, 2022] para o TSP generalizado (GTSP), uma variante do TSP em que cada cidade é composta por um conjunto finito de pontos e o objetivo é determinar um tour de custo mínimo que visite exatamente um ponto de cada cidade.

Meta-heurísticas integram em sua metodologia heurísticas construtivas e de melhoria de uma maneira eficiente, visando um equilíbrio entre as etapas de intensificação e diversificação na busca por soluções. Uma meta-heurística do tipo *Iterated Local Search* (ILS) [Lourenço et al., 2010] consiste em, iterativamente executar uma busca local até ela convergir para um mínimo local e então perturbar esta solução. A solução perturbada será utilizada como solução inicial para a próxima execução da busca local.

DOI: 10.59254/sbpo-2025-212601
https://proceedings.science/p/212601?lang=pt-br

O Algoritmo 1 apresenta o pseudocódigo do algoritmo baseado na meta-heurística ILS proposto para o TSPN com polígonos. Neste e nos algoritmos que seguem temos que m denota o número de polígonos, n o número de vértices de cada polígono e V_i contém as coordenadas dos vértices do polígono i. Os parâmetros η, h e $\ell_{\max}^{\text{NP}}$ indicam, respectivamente, o parâmetro de aceitação de uma nova solução, o fator pelo qual η é ajustado e o número máximo de iterações sem melhoria. O parâmetro $j_{\max}$ se refere ao número máximo de estruturas de vizinhança utilizadas pela meta-heurística *Variable Neighborhood Descent* (VND) que é empregada na etapa de busca local da ILS. Enquanto isso, $t = (t_1, \ldots, t_m)$ representa uma permutação dos m polígonos e $x \in \mathbb{R}^{2\times m}$ corresponde aos pontos visitados em cada polígono na ordem dada por t. Assim, um tour viável do TSPN é completamente representado pelo par (t, x). O comprimento do tour é dado pela soma das distâncias entre os pontos visitados, isto é,

$$f(t,x) := \|x^{t_m} - x^{t_1}\|_2 + \sum_{\nu=1}^{m-1} \|x^{t_\nu} - x^{t_{\nu+1}}\|_2, \tag{22}$$

em que $\|\cdot\|_2$ corresponde à norma euclidiana. As principais etapas do Algoritmo 1 são apresentadas a seguir.

Algoritmo 1: Algoritmo baseado na meta-heurística *Iterated Local Search* para o TSPN Euclidiano com vizinhanças poligonais.

Entrada: $m \in \mathbb{N}$, $n \in \mathbb{N}^m$, $V_i \in \mathbb{R}^{2\times n_i}$ para $i = 1, \ldots, m$, $j_{\max}$, $\ell_{\max}^{\text{NP}}$, η, $h \in (0,1)$, $t^0 \in \mathbb{N}^m$, $x^0 \in \mathbb{R}^{2\times m}$.
Saída: $t^* \in \mathbb{N}^m$, $x^* \in \mathbb{R}^{2\times m}$.

Função: ILS(m, n, V, $j_{\max}$, $\ell_{\max}^{\text{NP}}$, η, h, t^0, x^0, t^*, x^*)
  VND(m, n, V, $j_{\max}$, t^0, x^0)
  $t^* \leftarrow t^0$, $x^* \leftarrow x^0$
  $\kappa \leftarrow 0$ e $\ell \leftarrow 0$
  faça
    Perturbação(m, n, V, t^κ, x^κ, $\bar{t}$, $\bar{x}$)
    VND(m, n, V, $j_{\max}$, $\bar{t}$, $\bar{x}$)
    se $f(\bar{t},\bar{x}) < f(t^*,x^*)$ **então**
      $t^* \leftarrow \bar{t}$ e $x^* \leftarrow \bar{x}$
      $\ell \leftarrow 0$
    senão
      $\ell \leftarrow \ell + 1$
    se $f(\bar{t},\bar{x}) < f(t^\kappa,x^\kappa)$ ou $f(\bar{t},\bar{x}) \leq f(t^*,x^*) + \eta|f(t^*,x^*)|$ **então**
      $t^{\kappa+1} \leftarrow \bar{t}$ e $x^{\kappa+1} \leftarrow \bar{x}$
    senão
      $t^{\kappa+1} \leftarrow t^\kappa$ e $x^{\kappa+1} \leftarrow x^\kappa$
    $\kappa \leftarrow \kappa + 1$
    se $\kappa \mod m = 0$ **então**
      $\eta \leftarrow h\eta$
  enquanto $\ell < \ell_{\max}^{\text{NP}}$
  retorne t^* e x^*

Para construir uma solução inicial (t^0, x^0), uma heurística de inserção foi implementada.

DOI: 10.59254/sbpo-2025-212601
https://proceedings.science/p/212601?lang=pt-br

Primeiro, um vértice e seu respectivo polígono são escolhidos aleatoriamente para gerar um tour parcial. Tal tour será incrementado até gerar um tour viável, que visita os m polígonos, da seguinte forma: um novo polígono i é selecionado aleatoriamente, entre aqueles ainda não visitados, e um de seus vértices $y \in V_i$ é inserido no tour parcial atual tal que minimize o custo de inserção dado por $y = \text{argmin}_{z \in V_i, 1 \leq j \leq k}\{\|x^j - z\|_2 + \|z - x^{j+1}\|_2 - \|x^j - x^{j+1}\|_2, \}$. Após este processo, o tour gerado é submetido ao método de busca em coordenadas para ajustar os pontos visitados.

Na etapa de busca local o Algoritmo 1 opera uma meta-heurística do tipo *Variable Neighborhood Descent* (VND) [Hansen et al., 2010]. Esta meta-heurística realiza uma busca no espaço de soluções utilizando mais de uma estrutura de vizinhança. Assim, a solução final retornada por ela deve ser um mínimo local em relação a todas as vizinhanças utilizadas. O Algoritmo 2 corresponde ao algoritmo baseado na meta-heurística VND. Na linha 5 do Algoritmo 2, a busca em vizinhança realizada depende do parâmetro j de entrada, isto é, j indica qual a estrutura de vizinhança na qual será realizada uma busca por soluções melhores. Assim, dada uma solução (t, x), a VND inicia explorando a vizinhança associada a $j = 1$. Se um vizinho melhor $(\bar{t}, \bar{x})$ é encontrado, uma nova busca é realizada a partir de $(\bar{t}, \bar{x})$ se mantendo na vizinhança $j = 1$. Caso contrário, a vizinhança $j = 2$ é explorada em seguida e assim sucessivamente até $j = j_{\max}$. No entanto, sempre que um vizinho melhor é encontrado, a busca recomeça com a primeira vizinhança.

Algoritmo 2: Algoritmo baseado na meta-heurística *Variable Neighborhood Descent*.

```
   Entrada: m, n, V, j_max, t, x.
   Saída: t, x.
1  Função: VND(m, n, V, j_max, t, x)
2      Seja i_1, ..., i_{j_max} uma permutação aleatória de 1, ..., j_max
3      j ← 1
4      enquanto j ≤ j_max faça
5          BuscaVizinhança(m, n, V, i_j, t, x, t̄, x̄)
6          se f(t̄, x̄) < f(t, x) então
7              j ← 1
8              t ← t̄ e x ← x̄
9          senão
10             j ← j + 1
11     retorne t e x
```

Nove vizinhanças conhecidas da literatura do TSP foram combinadas ao método de busca em bloco de coordenadas e implementadas para operar a busca em vizinhança da VND. São elas as vizinhanças 2-Opt e 3-Opt [Lin, 1965], *Double-Bridge* [Glover, 1996], *Swap*, Inserção (ou *relocation*) e *String relocation* [Aarts e Lenstra , eds.], a vizinhança de Balas-Simonetti para $k = 2$ e $k = 3$ [Balas, 1999] e a vizinhança de Gutin (ou *Assignment neighborhood*) [Gutin e Glover, 2005]. Além disso, como é possível observar no Algoritmo 2, optamos por implementar uma ordenação aleatória com a qual a VND realiza a exploração das vizinhanças, assim como realizado em [Penna et al., 2013] e [Schmidt e Irnich, 2022]. Ou seja, sempre que a VND é chamada pela ILS, novas prioridades são selecionadas de modo aleatório para as vizinhanças e estas reordenadas de acordo.

O Algoritmo 3, corresponde à etapa de perturbação realizada pelo Algoritmo 1, que consiste na aplicação de um movimento do tipo *double-bridge* (linha 4) cujos nós das arestas a serem removidas são escolhidos de maneira aleatória. Após o movimento, o método de busca em bloco de

DOI: 10.59254/sbpo-2025-212601
https://proceedings.science/p/212601?lang=pt-br

coordenadas (BCD, linha 5) é aplicado para recalcular os pontos de visita em cada polígono.

Algoritmo 3: Realiza um movimento *double-bridge* em um tour (t, x) do TSPN em arestas escolhidas de modo aleatório.

Entrada: m, n, V, t, x.
Saída: t^*, x^*.
1 **Função** Perturbação(m, n, V, t, x, t^*, x^*)
2 Sorteie $i, j, k, \ell \in \{1, \ldots, m-1\}$ tal que $i \neq j \neq k \neq \ell$.
3 Permute i, j, k e ℓ de modo que $i < j < k < \ell$.
4 DoubleBridgeMove(t, i, j, k, ℓ, t^*)
5 BCD(m, n, V, t^*, x, x^*)
6 **retorne** t^* e x^*

O critério de aceitação de uma nova solução utilizado pelo Algoritmo 1 procura um equilíbrio entre diversificação e intensificação da busca. Uma solução $(\bar{t}, \bar{x})$ com custo menor que a anterior (t, x) é sempre aceita. Entretanto, soluções não necessariamente melhores também são aceitas, desde que a diferença entre seu custo e o custo da melhor solução (t^*, x^*) conhecida seja menor que um limite predefinido. Isto é, se

$$f(\bar{t}, \bar{x}) \leq f(t^*, x^*) + \eta |f(t^*, x^*)|. \tag{23}$$

Além disso, o parâmetro η é reduzido durante a ILS pelo fator h a cada m iterações.

5. Instâncias

Na introdução deste trabalho comentamos que não foi observado na literatura do TSPN experimentos computacionais com instâncias cujas vizinhanças correspondem a regiões poligonais arbitrárias. Com efeito, artigos que trabalham com o CETSP realizam testes com instâncias cujas vizinhanças são exclusivamente discos do plano euclidiano e artigos relacionados ao TSPN propriamente dito testam seus métodos em instâncias mais simples com regiões dadas, exclusivamente, por segmentos de reta, cópias de um mesmo polígono convexo ou nas instâncias do CETSP. Apenas no trabalho de [Yang et al., 2018] observou-se que uma única instância com 21 vizinhanças de diferentes formatos (segmentos de reta, discos, anéis, setores circulares e alguns polígonos) foi testada.

Em vista disso, um procedimento foi criado para construir instâncias para o nosso problema utilizando instâncias de outra variante do TSP, o GTSP (do inglês *Generalized Traveling Salesman Problem*). No GTSP, cada cidade é composta por um conjunto finito de pontos e o objetivo é determinar um tour de custo mínimo que visite exatamente um ponto de cada conjunto. Assim, instâncias de diferentes bibliotecas para o GTSP foram selecionadas e convertidas em instâncias para o TSPN com vizinhanças dadas por polígonos da seguinte forma: para cada conjunto de pontos da instância do GTSP, o centroide dos pontos foi calculado. Utilizando este ponto como referência, determinamos o ângulo entre cada um dos pontos do conjunto e o centroide. Em seguida, tais ângulos foram ordenados. Após essa ordenação, os pontos do conjunto passaram a exercer o papel de vértices de um polígono para a instância do TSPN. Portanto, cada conjunto do GTSP foi transformado em um polígono cujos vértices correspondem aos pontos do conjunto.

Ao todo 32 instâncias do GTSP foram convertidas em instâncias do TSPN com vizinhanças poligonais arbitrárias, com o número de polígonos variando de 10 a 89 e o total de vértices entre 51 e 1002, e estão disponíveis em `https://www.ime.usp.br/~paulaertel/ar`

DOI: 10.59254/sbpo-2025-212601
https://proceedings.science/p/212601?lang=pt-br

tigo-SBPO2025/instances/. Das 32 instâncias, 18 foram extraídas da biblioteca GTSP-Lib [Fischetti et al., 1997], 2 da biblioteca Large-Lib [Helsgaun, 2015] e 12 da biblioteca MOM-Lib [Mestria et al., 2013]. A seção a seguir apresenta os testes computacionais.

6. Experimentos numéricos

Nesta seção, apresentamos os resultados obtidos pelo Algoritmo 1 ao resolver as 32 instâncias do TSPN com vizinhanças poligonais introduzidas na seção anterior. Para a implementação dos algoritmos utilizamos a linguagem de programação Fortran 90 e como compilador GFortran. Os experimentos foram realizados em um computador com processador Intel Core i9-13900 CPU @ 5.300GHz e 128 GB de RAM. O Algoritmo 1 foi executado 10 vezes para cada instância, com um limite de tempo de 1.200 segundos para cada execução e, assim como proposto em [Schmidt e Irnich, 2022], os seguintes valores para os parâmetros foram utilizados: $\ell_{\max}^{\mathrm{NP}} = 10$, $\eta = 0{,}03$ e $h = 0{,}8$. Os resultados das 10 execuções estão disponíveis em https://www.ime.usp.br/~paulaertel/artigo-SBPO2025/. Além disso, para cada instância do TSPN construída a partir de uma instância do GTSP, uma solução para o GTSP fornece um limitante superior para a solução ótima do TSPN. Assim, as melhores soluções conhecidas para as instâncias do GTSP ([Helsgaun, 2015; Smith e Imeson, 2017; Schmidt e Irnich, 2022]) foram utilizadas para calcular o gap em relação às soluções encontradas pelo Algoritmo 1 para as respectivas instâncias do TSPN, da forma $\mathrm{gap}(\%) = 100\% \frac{(f_{\mathrm{TSPN}} - f_{\mathrm{GTSP}})}{f_{\mathrm{GTSP}}}$.

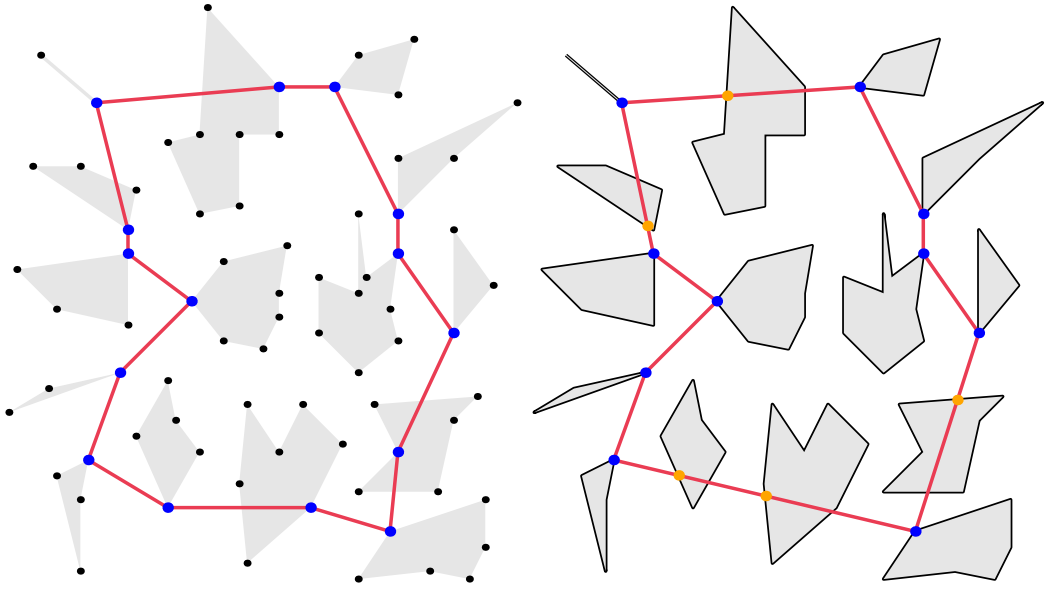
Considerando, para cada instância, a melhor solução encontrada pelo Algoritmo 1 nas 10 execuções, o gap variou de -1,46% (para a instância 10eil76) a 0,0% (para a instância 46pr226), com um gap médio de -0,32% para as 32 instâncias. O Algoritmo 1 encontrou soluções melhores que o limitante superior dado pelo GTSP em pelo menos uma execução para 31 instâncias. Apenas para a instância 46pr226 o algoritmo encontrou a mesma solução do GTSP em todas as 10 execuções. Em média, a melhor solução obtida pelo algoritmo para cada instância foi encontrada em 7,88 das 10 execuções, sendo que em 15 instâncias a melhor solução foi alcançada em todas as 10 execuções. A maior dificuldade em obter boas soluções foi observada nas instâncias com mais de 50 polígonos. Observando agora a média dos resultados obtidos nas 10 execuções para cada instância, o gap variou de -1,46% (para a instância 10eil76) a 1,67% (para a instância 88pr439), com um gap médio de -0,13% para as 32 instâncias. O desvio padrão para as 32 instâncias variou de 0 a 955,58, com uma média de 57,62 e uma mediana de 0,70. O maior desvio padrão foi observado para a instância 88pr439, cuja melhor solução obteve um gap de -0,16% e foi encontrada em somente uma execução. O alto valor do maior desvio padrão é um reflexo dos valores dos comprimentos das soluções, que são altos e variam consideravelmente entre as diferentes instâncias analisadas. Observando o desvio padrão relativo, este variou de 0,0% a 1,56%, com uma média de 0,21% para as 32 instâncias.

Por fim, apresentamos algumas figuras das soluções encontradas pelo Algoritmo 1. O tour solução é representado pela linha contínua na cor vermelha e os pontos de visita correspondem aos pontos em azul. Na Figura 1, a Figura 1a apresenta a solução reportada na literatura para a instância do GTSP, enquanto que na Figura 1b temos a solução encontrada pelo Algoritmo 1 para a instância do TSPN associada. Observe que na instância GTSP os conjuntos a serem visitados são discretos e, portanto, os pontos de visita estão localizados no que seriam os vértices dos polígonos da instância convertida para o TSPN. No TSPN, por sua vez, existe a liberdade de visitar qualquer ponto do polígono, não apenas seus vértices. Alguns destes pontos foram destacados na cor amarela para mostrar casos em que, visivelmente, os pontos de visita não estão localizados nos vértices dos polígonos, e sim em uma de suas arestas. A Figura 2 apresenta a melhor solução obtida pelo Algoritmo 1 em uma instância com 50 polígonos que totalizam 1002 vértices.

7. Conclusões

Neste trabalho, abordamos o Problema do Caixeiro Viajante com vizinhanças poligonais, uma variante do TSP ainda pouco estudada. Introduzimos um modelo de programação linear in-

DOI: 10.59254/sbpo-2025-212601
https://proceedings.science/p/212601?lang=pt-br



(a) Solução GTSP. Custo: 185,81 u.c. (b) Solução TSPN. Custo: 184,31 u.c.

Figura 1: Soluções para a instância ***15eil76***, com 15 polígonos e 76 vértices.

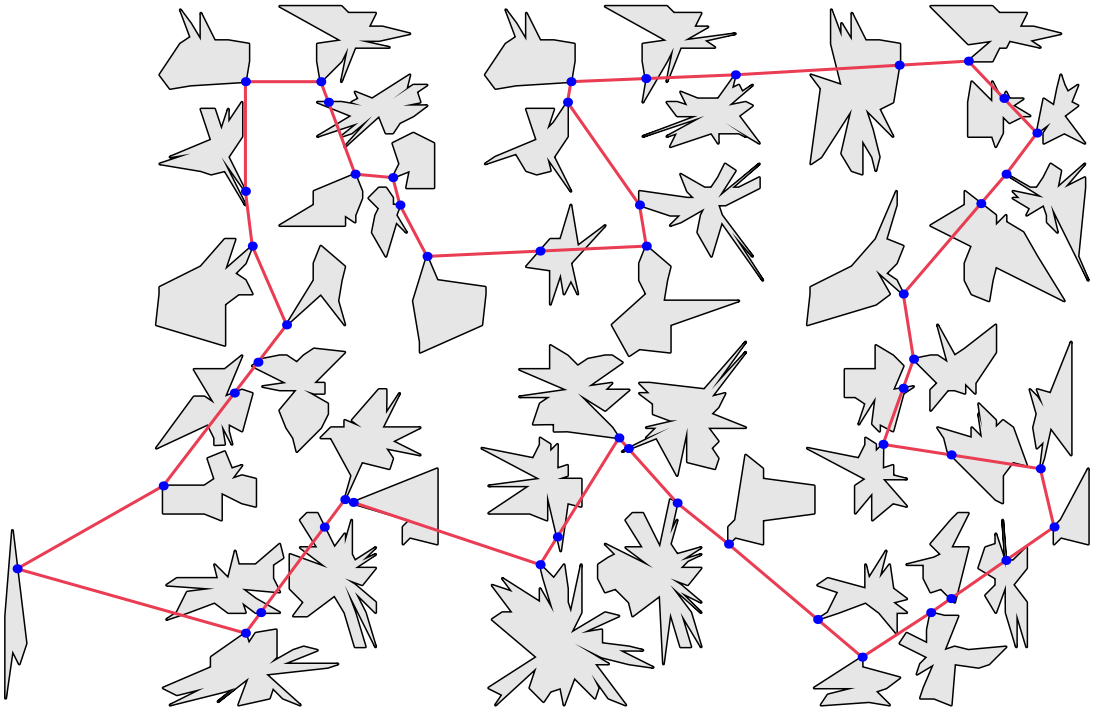


Figura 2: Solução para a instância ***50pr1002***, com 50 polígonos e 1002 vértices. Custo: 54.413,23 u.c. Gap em relação à solução do GTSP: -0,32%.

teira mista com restrições quadráticas para este problema e propomos um algoritmo baseado na meta-heurística *Iterated Local Search* (ILS) para resolvê-lo. Na etapa de busca local da ILS, nove vizinhanças do TSP foram combinadas a um método de busca em bloco de coordenadas para calcu-

DOI: 10.59254/sbpo-2025-212601
https://proceedings.science/p/212601?lang=pt-br

lar os pontos de visita e, assim, resolver instâncias do TSPN. Além disso, introduzimos um conjunto de 32 novas instâncias para o TSPN com polígonos. Verificamos na prática que o algoritmo proposto é capaz de resolver instâncias do TSPN com polígonos quaisquer, não necessariamente convexos, e obter soluções melhores do que os limitantes superiores fornecidos pelo GTSP. Como trabalho futuro, pretendemos resolver o modelo apresentado na Seção 2 para as instâncias introduzidas neste artigo e comparar com as soluções fornecidas pelo Algoritmo 1.

Referências

Aarts, E. e Lenstra (eds.), J. K. (2003). *Local Search in Combinatorial Optimization*. Princeton University Press.

Applegate, D. L., Bixby, R. E., Chvátal, V., e Cook, W. J. (2006). *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.

Arkin, E. M. e Hassin, R. (1994). Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55(3):197–218.

Balas, E. (1999). New classes of efficiently solvable generalized traveling salesman problems. *Annals of Operations Research*, 86:529–558.

Behdani, B. e Smith, J. C. (2014). An integer-programming-based approach to the close-enough traveling salesman problem. *INFORMS Journal on Computing*, 26(3):415–432.

Birgin, E. G. e Martínez, J. M. (2021). Block coordinate descent for smooth nonconvex constrained minimization. *Computational Optimization and Applications*, 83:1–27.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., e Stein, C. (2012). *Algoritmos: teoria e prática*. Campus.

Coutinho, W. P., do Nascimento, R. Q., Pessoa, A. A., e Subramanian, A. (2016). A branch-and-bound algorithm for the close-enough traveling salesman problem. *INFORMS Journal on Computing*, 28(4):752–765.

Dong, J., Yang, N., e Chen, M. (2007). Heuristic approaches for a TSP variant: The automatic meter reading shortest tour problem. In Baker, E. K., Joseph, A., Mehrotra, A., e Trick, M. A., editors, *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*, p. 145–163. Springer US, Boston, MA.

Dror, M. (1999). Polygon plate-cutting with a given order. *IIE Transactions*, 31(3):271–274.

Fischetti, M., González, J. J. S., e Toth, P. (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–394.

Glover, F. (1996). Finding a best traveling salesman 4-opt move in the same time as a best 2-opt move. *Journal of Heuristics*, 2(2):169–179.

Gutin, G. e Glover, F. (2005). Further extension of the TSP assign neighborhood. *Journal of Heuristics*, 11:501–505.

Gutin, G. e Punnen, A. (2006). *The Traveling Salesman Problem and Its Variations*. Combinatorial Optimization. Springer US.

DOI: 10.59254/sbpo-2025-212601
https://proceedings.science/p/212601?lang=pt-br

Hansen, P., Mladenovic, N., e Moreno-Pérez, J. (2010). Variable neighbourhood search: Methods and applications. *4OR*, 175:367–407.

Helsgaun, K. (2015). Solving the equality generalized traveling salesman problem using the Lin–Kernighan–Helsgaun algorithm. *Mathematical Programming Computation*, 7:269–287.

Laporte, G. (2010). A concise guide to the traveling salesman problem. *The Journal of the Operational Research Society*, 61(1):35–40.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *The Bell System Technical Journal*, 44(10):2245–2269.

Lourenço, H., Martin, O., e Stützle, T. (2010). *Iterated Local Search: Framework and Applications*, volume 146, p. 363–397. Springer New York.

Matos, R. S. (Em andamento). *Sobre o problema de roteamento de veículos com vizinhanças*. Tese de Doutorado em Matemática Aplicada, Universidade de São Paulo.

Mennell, W. K. (2009). *Heuristics for Solving Three Routing Problems: Close-Enough Traveling Salesman Problem, Close-Enough Vehicle Routing Problem, Sequence- Dependent Team Orienteering Problem*. PhD thesis, University of Maryland (College Park, Md.).

Mestria, M., Ochi, L. S., e de Lima Martins, S. (2013). Grasp with path relinking for the symmetric euclidean clustered traveling salesman problem. *Computers & Operations Research*, 40(12): 3218–3229.

Penna, P. H. V., Subramanian, A., e Ochi, L. S. (2013). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 19:201–232.

Placido, A. D., Archetti, C., e Cerrone, C. (2022). A genetic algorithm for the close-enough traveling salesman problem with application to solar panels diagnostic reconnaissance. *Computers & Operations Research*, 145:105–831.

Poikonen, S., Wang, X., e Golden, B. (2017). The vehicle routing problem with drones: Extended models and connections. *Networks*, 70(1):34–43.

Schmidt, J. e Irnich, S. (2022). New neighborhoods and an iterated local search algorithm for the generalized traveling salesman problem. *EURO Journal on Computational Optimization*, 10: 100029.

Shuttleworth, R., Golden, B. L., Smith, S., e Wasil, E. (2008). Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network. In Golden, B., Raghavan, S., e Wasil, E., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, p. 487–501. Springer US, Boston, MA.

Smith, S. L. e Imeson, F. (2017). GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem. *Computers & Operations Research*, 87:1–19.

Wright, S. J. (2015). Coordinate descent algorithms. *Mathematical Programming*, 151:3–34.

Yang, Z., Xiao, M., Ge, Y., Feng, D., Zhang, L., Song, H., e Tang, X. (2018). A double-loop hybrid algorithm for the traveling salesman problem with arbitrary neighbourhoods. *European Journal of Operational Research*, 265(1):65–80.

DOI: 10.59254/sbpo-2025-212601
https://proceedings.science/p/212601?lang=pt-br