



03 a 05 de dezembro de 2025

As Engenharias na Transformação Digital das PMEs

O Problema da Rota Multiperíodo mais Lucrativa com Prêmios de Prioridade: Modelagem e meta-heurística de Busca tabu

Pedro Di Lallo

Departamento de Engenharia de Produção - Universidade Federal de São Carlos

Vitória Pureza

Departamento de Engenharia de Produção - Universidade Federal de São Carlos

Reinaldo Morabito

Departamento de Engenharia de Produção - Universidade Federal de São Carlos

Resumo: Este estudo aborda o Problema da Rota Multiperíodo mais Lucrativa com Prêmios de Prioridade (MPTPPP) aplicado ao planejamento de itinerários turísticos com múltiplos períodos. O objetivo é definir roteiros personalizados e logisticamente viáveis que maximizem a satisfação do turista e minimizem os custos de transporte. Um modelo de otimização inteira mista é apresentado para o MPTPPP considerando paradas em atrações turísticas, restaurantes e hotéis, cada um com janelas de tempo específicas e durações de visita. Para resolver o MPTPPP, um método baseado na meta-heurística Busca Tabu foi desenvolvido e comparado com a solução do modelo. Experimentos computacionais demonstram a eficácia da meta-heurística na geração de soluções de alta qualidade para instâncias complexas, quando métodos exatos enfrentam dificuldades de resolução com o crescimento do tamanho das instâncias. Esta pesquisa contribui para a literatura ao considerar o problema em cenários multiperíodos com restrições temporais, e oferecendo soluções práticas para aplicações reais.

Palavras-chave: Otimização de roteiros turísticos, busca tabu, MPTPP, MIP, meta-heurísticas

The Multiperiod Most Profitable Route Problem with Priority Premiums: Modeling and Tabu Search Metaheuristic

Abstract: This study addresses the Multiperiod Profitable Tour Problem with Priority Prizes (MPTPPP) applied to the planning of tourist itineraries that cover multiple periods. The objective is to define personalized and logistically viable itineraries that maximize tourist satisfaction and minimize transport costs. A mixed integer optimization model is presented for the MPTPPP considering stops at tourist attractions, restaurants and hotels, each with specific time windows and visit durations. To solve the MPTPPP, a method based on the Tabu Search metaheuristic was also developed and compared with the model's solution. Computational experiments demonstrate the effectiveness of meta-heuristics in generating high-quality solutions for complex instances, especially when exact methods face difficulties in solving as the size of the instances grows. This research contributes to the literature by considering the problem in multi-period scenarios with time constraints, and offering practical solutions for real applications.

Keywords: Tourist Routing Optimization, Tabu Search, MPTPP, MIP, Metaheuristics.

1. Introdução

Em um mundo globalizado, com intenso fluxo de pessoas de e para várias partes do mundo, o turismo se estabelece como uma das atividades econômicas mais complexas e importantes. A Organização Internacional do Trabalho (OIT) estima que as atividades relacionadas ao turismo são diretamente responsáveis por 7,2% do total de empregos no mundo (INTERNATIONAL LABOUR ORGANIZATION, 2023). Essa contribuição econômica é geralmente ignorada, e este projeto tem isso como uma de suas motivações. Outro ponto importante a ser considerado é a retomada das atividades turísticas globais, um setor que foi particularmente impactado pela pandemia da COVID-19. Conforme a Organização Mundial do Turismo (UNTWO), por exemplo, em 2021 o setor perdeu US\$ 2 trilhões devido à pandemia (UN NEWS, 2022). Em 2024, a recuperação total foi confirmada no último relatório da UNWTO (UNWTO, 2024). O mesmo relatório estabeleceu uma perspectiva positiva para o crescimento contínuo do turismo. Do ponto de vista do turista e de agências de viagem, entretanto, planejar viagens turísticas costuma ser uma tarefa difícil. Escolher o que visitar, onde comer e onde dormir, bem como a ordem dessas atividades, pode ser exaustivo, especialmente ao considerar os diferentes pontos de interesse, as restrições de tempo e os trechos para visitar cada atração, hotel ou restaurante.

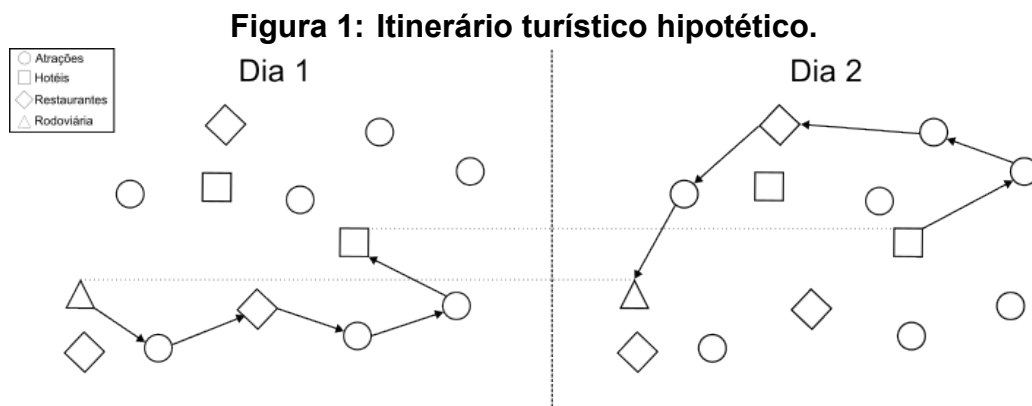
A literatura contém diversos artigos científicos sobre o desenvolvimento de itinerários turísticos, mas a maioria deles se concentra na coleta de dados, permitindo que os turistas criem seus próprios itinerários (SILVA et al., 2018). Outros artigos estudaram a eficiência logística na dinâmica do turismo. Por exemplo, ZHANG, YU et al., 2019 constatou que redes de transporte bem projetadas aumentam o turismo receptivo ao facilitar as experiências de viagem, enquanto a degradação ambiental — muitas vezes um subproduto de uma logística mal gerenciada afasta os visitantes. JANSSEN, 2022 argumentou que as abordagens logísticas orientadas por dados podem otimizar os fluxos de viagem. TUO et al., 2025 mostra que a IA pode aprimorar o design de itinerários. BAYLISS, 2021 estabelece que os algoritmos de aprendizado de máquina são eficazes na otimização do roteamento urbano, facilitando a navegação em ambientes urbanos complexos para itinerários turísticos. Tradicionalmente, dois problemas proeminentes foram usados para representar a criação de itinerários: o *Orienteering Problem with Time Windows* (OPTW) (GONÇALVES et al., 2017) e o *Profitable Tour Problem with Priority Prizes* (PTPPP). Um desenvolvimento interessante é a abordagem *Utility-Aware Trip Planning* (UATP) (GAO et al., 2018), que considera o impacto coletivo de todos os turistas que visitam uma atração, considerando as restrições de capacidade.

SILVA et al., 2018 formaliza o PTPPP e propõe um modelo de programação matemática destinado a fornecer itinerários otimizados com base em uma função objetivo que combina custos de viagem e satisfação do turista, sendo que esta última também engloba as ordens de visitação das atrações. No entanto, o modelo se limita a itinerários de um único dia e não considera os intervalos de tempo durante os quais as atrações estão disponíveis para visitas ou a duração das visitas. Em ALIANO FILHO e MORABITO, 2024, é proposto um modelo com dois objetivos para otimizar itinerários turísticos. Essa abordagem considera a roteirização em vários períodos, restrições como janelas de tempo e inclui a programação de restaurantes e hotéis, com foco em compensações entre os dois objetivos conflitantes. Esta pesquisa visa contribuir para essa área, apresentando um modelo mono-objetivo que amplia o que é proposto em SILVA et al., 2018 ao considerar vários períodos, janelas de tempo, paradas em restaurantes e pernoites em hotéis, e propondo um método heurístico baseado em Busca Tabu para resolver o problema eficazmente. Esse método também pode ser

útil para lidar com a versão com dois objetivos do problema discutido em ALIANO FILHO e MORABITO, 2024, em que são necessárias soluções de compromisso entre dois objetivos.

2. Definição do problema

O problema abordado nesta pesquisa consiste em definir, para cada período (dia) t de um horizonte de planejamento com $t = 1 \cdots T_f$ períodos, uma sequência de visitas (itinerário) de um turista a um subconjunto de possíveis locais oferecidos. Os locais oferecidos são classificados em três tipos: atrações turísticas, restaurantes e hotéis, cada um sujeito a janelas de tempo que definem o intervalo em que a visita deve começar e a duração da estadia no local. Algumas atrações abrem somente pela manhã, outras à noite. Alguns hotéis não permitem check-ins após uma determinada hora e alguns restaurantes só atendem clientes em horários predeterminados. A visita a cada atração turística, restaurante ou hotel resulta em dois *prêmios* concedidos ao turista, que representam, respectivamente, uma medida de satisfação por visitar o local e uma medida de satisfação por visitar o local em uma *ordem* específica na sequência. Ou seja, o turista pode ter uma preferência maior por visitar algumas atrações ou restaurantes em uma determinada ordem do que em outras. Além desses prêmios, visitar um local específico envolve um *custo de transporte* de onde o turista se encontra até o local em questão. Além do *custo de transporte*, há também um *tempo de viagem* relacionado ao deslocamento. No primeiro período ($t=1$), o itinerário começa no local no qual o turista chega à região de interesse (por exemplo, o aeroporto ou o terminal de ônibus de uma cidade), a partir do qual são visitadas as atrações turísticas e um restaurante. No final de $t = 1$, o itinerário inclui um hotel para pernoite, que se torna o ponto de partida do itinerário do segundo período ($t=2$), quando novas atrações e o mesmo ou outro restaurante e hotel são visitados. Embora seja necessário que exatamente um restaurante seja visitado em cada período, não há pernoite em um hotel no último período ($t=T_f$); em vez disso, o itinerário termina no local em que o turista chegou à região. O objetivo do problema é obter a sequência de visitas com o lucro máximo possível, definido como a diferença entre os prêmios recebidos pelo turista e o custo total de transporte. A Figura 1 ilustra um itinerário hipotético de 2 dias com 6 atrações, 2 restaurantes e 1 hotel.



O problema pode ser representado por um grafo $G=(N,A)$ onde o local de chegada do turista, as atrações turísticas, os restaurantes e os hotéis correspondem a um subconjunto de nós em N , e o caminho entre cada par desses locais corresponde a um subconjunto de arcos em A . Observa-se também que a sequência de visitas forma um ciclo que começa e retorna ao nó de chegada do turista. É importante observar que não é obrigatório visitar todos os nós em N , ou seja, na região turística. O problema é, portanto, uma extensão multiperíodo do PTPPP, discutido na Introdução. Nessa extensão, o turista parte de um aeroporto ou terminal de ônibus, visita atrações, um restaurante e pernoita em um hotel (cada um com sua própria

programação e duração de visita), recebendo prêmios de satisfação e pagando os custos de transporte. Nos dias seguintes, o turista visitará outras atrações, um restaurante e um hotel. No final do último dia da viagem, o turista não pernoita, mas retorna ao local de chegada, e o itinerário é concluído. Chamamos essa extensão de *Multiperiod Profitable Tour Problem with Priority Prizes (MPTPPP)*.

3. Modelagem

O modelo matemático a seguir apresenta o problema descrito na seção anterior. São exibidos os índices, conjuntos, parâmetros, variáveis, objetivos e restrições. Os índices são definidos como segue: i, j denotam nós; k representa a ordem de visita para os nós; t indica o índice dos períodos do modelo. Os conjuntos são: $N = 1, 2, \dots, n$ compreende os nós do grafo G , onde $i = 1$ é o ponto de chegada do turista na região (por exemplo, um aeroporto ou rodoviária) e serve como ponto de início e término do itinerário, e $i = 2, \dots, n$ são as atrações, restaurantes e hotéis oferecidos; $A(i, j)$ inclui os arcos do grafo G que conectam cada par de nós i e j ($\forall i, j \in N$); $T = 1, 2, \dots, T_f$ representa os dias de planejamento do itinerário; $R \subset N$ consiste nos nós que são restaurantes; $H \subset N$ consiste nos nós que são hotéis; $R_t \subset R$ inclui os nós restaurantes abertos no dia $t \in T$; $H_t \subset H$ inclui os nós hotéis abertos no dia t . Os parâmetros são: c_{ij} é o custo de percorrer o arco $(i, j) \in A$; p_i é o prêmio de visita do nó $i \in N$; p_{ki} é o prêmio de visita do nó $i \in N$ na ordem $k \in N$; w_0 é o horário de início do itinerário; d_i é a duração da visita ao nó $i \in N$; e_{ij} é o tempo de viagem ao percorrer o arco $(i, j) \in A$; $[a_{it}, b_{it}]$ são os limites inferior e superior da janela de tempo para o nó $i \in N$ e $t \in T$. As variáveis de decisão são: $x_{ij} \in 0, 1$ igual a 1 se o arco $(i, j) \in A$ for percorrido, e 0 caso contrário; $y_{ki} \in 0, 1$ igual a 1 se o nó $i \in N$ for visitado na ordem $k \in N$, e 0 caso contrário; ϕ_{it} igual a 1 se o nó $i \in N$ for visitado no dia $t \in T$, e 0 caso contrário; w_i é o horário de chegada ao nó $i \in N$; w_{it} é o horário de chegada ao nó $i \in N$ no dia $t \in T$. Deve-se notar que a mesma função objetivo é utilizada neste artigo e em SILVA et al., 2018.

$$\begin{aligned}
\text{Max} \quad & \sum_{k \in N} \sum_{i \in N} (p_{ki} + p_i) \cdot y_{ki} - \sum_{i=1}^n \sum_{j=1}^n c_{ij} \cdot x_{ij} \quad (1) & w_j &\geq w_0 + e_{1j} \cdot x_{1j} - M \cdot (1 - x_{1j}); \forall (1, j) \in A \quad (10) \\
& \sum_{j: (1, j) \in A} x_{1, j} = 1 \quad (2) & w_j &\geq w_i + d_i \cdot x_{ij} + e_{ij} \cdot x_{ij} - M \cdot (1 - x_{ij}); \\
& \sum_{i: (i, 1) \in A} x_{i, 1} = 1 \quad (3) & & \forall (i, j) \in A; i \neq 1 \quad (11) \\
& \sum_{i: (i, j) \in A} x_{ij} = \sum_{k \in N} y_{kj} \quad \forall j \in N \quad (4) & w_{it} &\leq M \phi_{it}; \forall i \in N; \forall t \in T \quad (12) \\
& \sum_{j: (i, j) \in A} x_{ij} = \sum_{k \in N} y_{ki} \quad \forall i \in N \quad (5) & w_i &= \sum_{t \in T} w_{it}; \quad \forall i \in N \quad (13) \\
& x_{1, j} = y_{1j} \quad \forall (i, j) \in A \quad (6) & a_{it} - a_{it} \cdot (1 - \phi_{it}) &\leq w_{it} \leq b_{it} - b_{it} \cdot (1 - \phi_{it}); \quad (14) \\
& x_{ij} \geq y_{k-1, i} + y_{k, j} - 1 \quad \forall (i, j) \in N; \forall k \in N; k \neq 1 \quad (7) & & \forall i \in N; \forall t \in T \\
& \sum_{i \in N} y_{ki} \geq \sum_{j \in N} y_{k+1, j} \quad \forall k \in N; k \neq n \quad (8) & \sum_{k \in N} \sum_{i \in H_t} y_{ki} &= 1; \quad \forall t \in T, t \neq T_f \quad (15) \\
& \sum_{t \in T} \phi_{it} = \sum_{k \in N} y_{ki}; \forall i \in N \quad (9) & \sum_{k \in N} \sum_{i \in R_t} y_{ki} &= 1; \quad \forall t \in T \quad (16) \\
& & w_{it} &> 0; w_i > 0; \phi_{it} \in \{0, 1\}; x_{ij} \in \{0, 1\}; \\
& & y_{ki} &\in \{0, 1\}; (i, j) \in A; k \in N; i \in N, t \in T \quad (17)
\end{aligned}$$

A função objetivo (1) a ser maximizada consiste na soma dos prêmios totais coletados pela visita aos nós e pela ordem em que são visitados, subtraída dos custos totais de transporte incorridos. Estes dois termos representam, na verdade, dois objetivos conflitantes: o

primeiro incentiva o turista a viajar mais (para coletar mais prêmios), enquanto o segundo motiva a redução da quilometragem percorrida (para diminuir os custos). As restrições (2) e (3) garantem que o roteiro seja um ciclo que começa e termina no nó 1 (o ponto de chegada do turista na região, ou o depósito). Além disso, garantem que somente um nó suceda imediatamente o nó 1 e que apenas um nó preceda imediatamente o nó 1. As restrições (4) e (5) impõem, respectivamente, que cada nó j ofertado e efetivamente visitado na k -ésima ordem do roteiro tenha exatamente um nó predecessor imediato i , e que cada nó i ofertado e efetivamente visitado na k -ésima ordem do roteiro tenha exatamente um nó sucessor imediato j . As restrições (6) garantem que o nó que sucede o nó 1 tenha ordem 1. As restrições (7) prescrevem que, se o nó i é o predecessor imediato do nó j no roteiro, então a ordem de i precede imediatamente a ordem de j . As restrições (8) evitam soluções simétricas (equivalentes), impondo que os nós visitados tenham o menor valor de ordem possível (ou seja, a ordem $k + 1$ não pode ser atribuída a um nó a menos que a ordem k tenha sido atribuída ao nó imediatamente anterior). As restrições (9) atribuem o dia de visita t a cada nó i , se i for visitado. Se o nó j é o sucessor imediato de $i = 1$, as restrições (10) prescrevem que o tempo de chegada em j seja maior ou igual à soma do tempo de início do roteiro no nó 1 (w_0) e do tempo de viagem do nó 1 para o nó j . As restrições (11) também prescrevem o tempo de chegada em j , mas quando seu predecessor imediato i não é o nó 1; neste caso, o lado direito das restrições inclui a duração da visita em i (d_i). As restrições (12) e (13) impõem, respectivamente, que o tempo de chegada no nó i no dia t (w_{it}) seja atribuído ao nó i somente se ele for visitado no dia t , e que o tempo de chegada no nó i (w_i) seja igual a w_{it} se i for visitado no dia t . As restrições (14) limitam o tempo de chegada em cada nó i visitado ao intervalo definido pelas janelas de tempo de i no dia t em que i é visitado. A garantia de visitar exatamente um restaurante em cada dia $t \in [1, \dots, T_f]$ e exatamente um hotel em cada dia $t \in [1, \dots, T_f - 1]$ é fornecida pelas restrições (16) e (15), respectivamente. Por fim, as restrições (17) fornecem o domínio das variáveis de decisão.

4. Heurística inspirada na Busca Tabu

Um método heurístico baseado na metaheurística de Busca Tabu (GLOVER e LAGUNA, 1997) foi desenvolvido e implementado para resolver o problema. Ele utiliza uma heurística construtiva e a busca local, que serão descritos a seguir. A heurística construtiva é baseada em uma proposta de SOLOMON, 1987, chamada *Best Insertion* (iii). Inicialmente, há um procedimento para inicializar a rota adicionando apenas os nós obrigatórios: nó 1, hotéis (em $t = 1, \dots, T_f - 1$) e restaurantes (em $t = 1, \dots, T_f$). No primeiro dia, o hotel mais distante do nó 1 é selecionado e adicionado à rota (que se torna $1 \rightarrow H_1 \rightarrow 1$). Em seguida, o hotel mais distante do anterior é selecionado e os restaurantes são escolhidos aleatoriamente, um para cada dia. A rota final inicializada é $1 \rightarrow R_1 \rightarrow H_1 \rightarrow R_2 \rightarrow H_2 \rightarrow \dots \rightarrow R_n \rightarrow 1$. O objetivo dessa inicialização é fornecer um bom “contorno” para a rota inicial, permitindo a inserção de mais atrações na rota, e ela é inspirada no algoritmo de inserção mais distante, definido em ROSENKRANTZ et al., 1977. O algoritmo de inicialização é então representado no algoritmo 1.

Algoritmo 1 Algoritmo de Inicialização

1. Seja $S = \langle 1^c, 1^p \rangle$ a rota multiperíodo corrente, na qual 1^0 representa o aeroporto/rodoviária no dia da chegada do turista e 1^f representa o aeroporto/rodoviária no dia de sua partida.
 2. Para cada dia $t = 1 \dots T - 1$:
 - (a) Dentre os hotéis $\in H_t$ e com disponibilidade para acomodar o turista no dia t , selecione h^* como aquele que apresenta a maior distância do penúltimo nó de S e que mantém S factível quando adicionado à posição k que precede à do 1^p . Adicione h^* a S na posição k .
 - (b) Dentre os restaurantes $\in R_t$, selecione aleatoriamente r^* que mantenha S factível quando adicionado à posição que precede à de h^* escolhido no passo anterior.
 3. Compute o instante de início de visita em cada hotel e restaurante selecionado no Passo 2.
 4. Retorne a rota resultante S e sua programação.
-

A partir desse ponto, a heurística tenta inserir nós de atração ($u \in U$, o conjunto de nós não visitados) em suas melhores inserções possíveis, na ordem k , até que alguma restrição seja violada ou não haja mais nós para inserir. A melhor inserção é calculada com base no custo, no tempo e no prêmio adicionado à rota, todos com pesos iguais. O algoritmo também considera todos os dias simultaneamente para as inserções. Essa parte do procedimento é representada no algoritmo 2.

Algoritmo 2 Algoritmo da Melhor Inserção

1. Seja $S = \langle 1^c, i2, i3, i4, \dots, 1^p \rangle$ a rota multiperíodo corrente e U o conjunto de atrações não visitadas.
 2. Repita:
 - (a) Para cada atração $u \in U$:
 - i. Compute $c_1(u, k) = \text{prêmio adicionado} - \text{distância adicionada} - \text{tempo adicionado}$ resultante da inserção de u em cada posição factível de inserção k da rota S .
 - ii. Caso haja pelo menos uma posição factível de inserção de u na rota, faça $c_1^*(u, k^*) = \max_k c_1(u, k)$.
 - (b) Caso haja pelo menos uma posição factível de inserção de alguma atração $u \in U$ na rota com $c_1^*(u, k^*) > 0$:
 - i. Selecione $u^* \in U$ tal que $c_2(u^*) = \max_u c_1^*(u, k^*)$ e insira-o na posição k^* de S . Faça $U = U - \{u^*\}$.
 - ii. Atualize o prêmio e a distância total de S com a inserção de u^* .
 - iii. Compute o tempo de início de visita em u^* e atualize a programação dos nós subsequentes.
 - Até que não haja posições factíveis de inserção com $c_1^*(u, k^*) > 0$.
 3. Retorne a rota resultante S e sua programação.
-

Na Busca Local implementada, foram desenvolvidos quatro métodos de geração de vizinhança que juntos formam a vizinhança $N(S)$: Remoções, Adições, Inserções e Trocas. Observa-se que algumas considerações foram necessárias para manter a viabilidade das rotas na heurística de busca local, conforme definido matematicamente na seção anterior e no algoritmo 3. Não se pode remover hotéis ou restaurantes, nem adicionar hotéis ou restaurantes. Hotéis só podem fazer SWAP com outros hotéis, enquanto restaurantes podem fazer SWAP com outros restaurantes ou com atrações (nós comuns), desde que ambos estejam sendo visitados no mesmo dia. Além disso, restaurantes podem fazer inserções para qualquer posição k , desde que isso não implique alocá-los para um dia diferente do

que já estão alocados. Assim, o algoritmo de busca local implementado é representado no algoritmo 3.

Algoritmo 3 Busca Local

1. Seja $S = \langle 1^c, i2, i3, i4, \dots, 1^p \rangle$ a rota multiperíodo corrente obtida com o Algoritmo da Melhor Inserção.
 2. Repita:
 - (a) Compute a vizinhança Adição de S , resultado da inclusão de uma atração i não visitada em S , com $i \notin H \cup R$.
 - (b) Compute a vizinhança Remoção de S , resultado da remoção de um nó $i \in S$ tal que $i \notin H \cup R$, $i \neq 1^c$ e $i \neq 1^p$.
 - (c) Compute a vizinhança Troca de S , resultado da troca de posições de dois nós distintos i e j em S , considerando a condição $(i, j \in S \wedge (i \neq 1^c) \wedge (i \neq 1^p) \wedge (j \neq 1^c) \wedge (j \neq 1^p))$ e, adicionalmente: $((i \notin H \cup R) \rightarrow (j \notin H \cup R)) \wedge ((i \in R) \rightarrow (j \in R)) \wedge ((i \in H) \rightarrow (j \in H))$.
 - (d) Compute a vizinhança Inserção de S , resultado da mudança de posição k do nó i para a posição k' em S , com a restrição de que $((i \in R) \vee (i \in H)) \rightarrow (\text{dia}(k') = \text{dia}(k))$. Isto significa que restaurantes e hotéis não podem trocar de dia.
 - (e) Caso haja pelo menos uma rota vizinha factível dentre as obtidas nos Passos 2.1 a 2.4 com lucro superior a $p(S)$:
 - i. Selecione a rota vizinha S' como aquela de maior lucro, e faça $S \leftarrow S'$ e $p(S) \leftarrow p(S')$.

Até que não haja soluções vizinhas factíveis com lucro superior a $p(S)$.
 3. Retorne a rota resultante S e sua programação.
-

A aplicação do algoritmo inspirado na meta-heurística de Busca Tabu (GLOVER e LAGUNA, 1997) justifica-se pela sua habilidade em ampliar a diversificação na procura por novas soluções. O método inicia a partir de uma solução factível e investiga movimentos que conduzam a soluções não exploradas anteriormente, favorecendo uma busca mais extensa pelo espaço de soluções. A versão do algoritmo *Busca Tabu* adotada neste trabalho (4) adota uma estrutura clássica, na qual o algoritmo percorre soluções viáveis, evitando retornar a áreas já examinadas do espaço de soluções via uma matriz tabu. O processo começa com a aplicação da heurística construtiva e posterior Busca Local para gerar a solução inicial. Depois, examina a vizinhança da solução vigente, escolhendo o movimento de maior lucro que não se encontre tabu-ativo. Neste contexto, um movimento tabu-ativo refere-se àquele que anula a inclusão ou exclusão de ao menos a metade dos arcos alterados por um movimento prévio, contanto que a iteração presente esteja aquém do período tabu itx dessas alterações. Caso ocorra aprimoramento, a solução é renovada. Tal abordagem fundamenta-se em PUREZA et al., 2012 e foi ajustada ao problema tratado aqui. O ciclo prossegue por um número fixo de iterações, com a matriz tabu anotando os movimentos efetuados para assegurar a diversificação, inibindo o retorno a soluções próximas no tempo. Decorrido certo número de iterações, o movimento perde a classificação de “tabu-ativo”. Sejam $pt = \frac{|N|}{2}$ o período tabu e it a iteração atual. Define-se itx por escolha equiprovável de um inteiro em um dos três intervalos abaixo:

$$itx \in \begin{cases} [\lfloor (pt + it)/2 \rfloor - 2, \lfloor (pt + it)/2 \rfloor + 2], \\ [pt + it, pt + it + 2], \\ [\lfloor 2(pt + it) \rfloor - 2, \lfloor 2(pt + it) \rfloor + 2]. \end{cases}$$

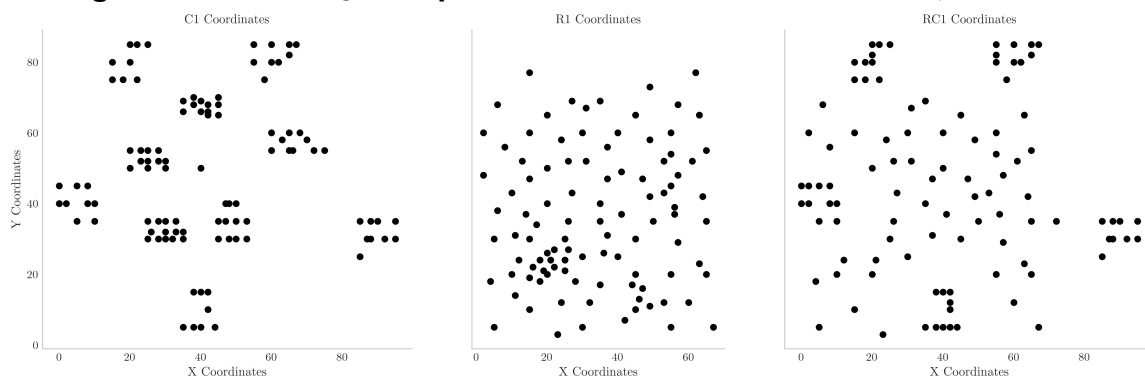
Algoritmo 4 Busca Tabu

1. Faça $t = 0$.
 2. Gere a rota multiperíodo S com o Algoritmo de Inicialização seguido do Algoritmo da Melhor Inserção. Faça $S^* \leftarrow S$ e $p(S^*) \leftarrow p(S)$.
 3. Enquanto $t \leq \max T_{BT}$:
 - (a) Compute as vizinhanças utilizadas no Algoritmo de Busca Local a S .
 - (b) Armazene as rotas vizinhas factíveis obtidas no passo anterior em ordem não crescente de lucro na lista L . Faça $C \leftarrow L$.
 - (c) Enquanto um movimento não tiver sido realizado ou $L \neq \emptyset$:
 - i. Selecione a rota vizinha S' na primeira posição de L .
 - ii. Se $p(S') > p(S^*)$ ou se os atributos do movimento m que resulta em $S \xrightarrow{m} S'$ não o classificam como tabu-ativo:
 - A. Faça $S \leftarrow S'$, $p(S) \leftarrow p(S')$ e classifique como tabu por it_x iterações os atributos do movimento que reverte m .
 - B. Se $p(S) < p(S^*)$, faça $S^* \leftarrow S$ e $p(S^*) \leftarrow p(S)$.Caso contrário, elimine S' da lista L .
 - iii. Se $L \neq \emptyset$ e nenhum movimento foi realizado, remova a classificação tabu do movimento mais antigo realizado e faça $L \leftarrow C$.
 4. Retorne a rota resultante S^* e sua programação.
-

5. Experimentos computacionais

As instâncias usadas foram geradas com base nos exemplos de 100 nós propostos por SOLOMON, 1987 para o problema de roteamento de veículos com janelas de tempo (VRPTW). O interesse nesses exemplos decorre de sua classificação em classes distintas, caracterizadas pela maneira como os nós são distribuídos no espaço bidimensional. Especificamente, a classe C (agrupada: subconjuntos de nós geograficamente próximos uns dos outros), a classe R (aleatória: distribuição aleatória de nós) e a classe RC (agrupada aleatoriamente: uma mistura das distribuições anteriores). A Figura 2 mostra a distribuição espacial de três instâncias C1, R1 e RC1 originais do Solomon.

Figura 2: Distribuição espacial dos nós das instâncias C1, R1 e RC1.



Dessas instâncias, utilizaram-se exclusivamente as coordenadas, com janelas de tempo geradas aleatoriamente usando duas categorias diferentes: normalmente e uniformemente distribuídas. Os horários de atendimento de cada atração foram escolhidos seguindo uma distribuição uniforme e são iguais para cada instância. Para simplificar, os hotéis são os mesmos para todas as instâncias, assim como os restaurantes. O depósito é sempre o nó 1.

A partir desse ponto, havia 3 classes $\times$ 2 distribuições = 6 instâncias, já que as coordenadas são as mesmas dentro de cada grupo nas instâncias geradas por Solomon SOLOMON, 1987. Cada uma delas, contendo 100 nós, originou mais cinco instâncias, uma com 10 e as outras com 20, 30, 50 e 80 atrações, por haver 10 restaurantes e 10 hotéis. Essas, portanto, originaram mais três instâncias: cada uma contendo 2, 4 e 7 dias. Assim, resultando em um total de 90 instâncias. Um segundo lote de instâncias também foi criado em um esquema semelhante ao do primeiro, mas dessa vez foi gerado um total de 96 instâncias, dividindo as instâncias originais de Solomon em grupos de dois e quatro dias e número de atrações variando de 3, 5, 7, 10, 15, 30, 50 e 80, com mais instâncias menores em tamanho total de nós. Portanto, foi gerado um total de 186 instâncias, combinando os dois lotes.

Os resultados foram obtidos com a execução de cada instância uma vez por 300 segundos com a meta-heurística. O modelo (1) - (17) foi implementado usando a biblioteca Pyomo do Python e os parâmetros passados para o solver foram um limite de tempo de 3600 segundos e um limite de memória de 10 Gb, que foi usado exclusivamente pelo Gurobi, o solver utilizado. Para esse software, foi instalada a versão “11.0.3 build v11.0.3rc0”. O ambiente em que os experimentos foram realizados consiste em um Lenovo ThinkPad com 16 Gb de RAM, processador i7 de 11ª geração e sistema operacional openSUSE Tumbleweed. O primeiro resultado é que o solver não foi capaz de resolver a maioria das instâncias. Isso se deve às limitações de tempo e memória nos experimentos, com algumas instâncias não podendo ser carregadas na memória e a maioria impedindo o solver de encontrar uma solução em menos de uma hora. Isso está representado na Tabela 1, que mostra a quantidade resolvida e o número total de instâncias para cada classe (C, R e RC) e tipos de distribuição (normal e uniforme) de probabilidade. Os cabeçalhos da tabela indicam o limite superior dos compartimentos, aumentados em 25 nós em cada coluna, começando de 1 a 25 nós (coluna 25), seguido de 26 a 50 nós (coluna 50), e assim por diante, até 201 a 225 nós (coluna 225). Observe que nenhuma instância de sete dias pôde ser resolvida pelo Gurobi em menos de uma hora e com 10 Gb de RAM, e todas as instâncias com até 25 nós foram resolvidas, como mostrado na Tabela 2. Foram encontradas soluções viáveis para instâncias com até 125 nós, mas somente para aquelas com apenas 2 dias. Isso destaca o impacto que o número de dias tem sobre a dificuldade do problema.

Tabela 1: Distribuição de frequência das instâncias solucionadas pelo Gurobi por tipo de instância, classe e distribuição das janelas de tempo.

Tipo	Classe	Distribuição	[0,25]	(25,50]	(50,75]	(75,100]	(100,125]	(125,150]	(150,175]	(175,200]	(200,225]
1	C1	normal	1/1	1/3	1/2	1/3	0/3	0/0	0/2	0/0	0/1
		uniforme	1/1	2/3	1/2	1/3	1/3	0/0	0/2	0/0	0/1
	R1	normal	1/1	1/3	1/2	1/3	0/3	0/0	0/2	0/0	0/1
		uniforme	1/1	2/3	1/2	1/3	1/3	0/0	0/2	0/0	0/1
	RC1	normal	1/1	1/3	1/2	1/3	0/3	0/0	0/2	0/0	0/1
		uniforme	1/1	2/3	1/2	1/3	1/3	0/0	0/2	0/0	0/1
2	C1	normal	3/3	2/3	2/4	0/2	0/2	0/1	0/1	0/0	0/0
		uniforme	3/3	2/3	2/4	0/2	0/2	0/1	0/1	0/0	0/0
	R1	normal	3/3	2/3	2/4	0/2	0/2	0/1	0/1	0/0	0/0
		uniforme	3/3	2/3	2/4	0/2	0/2	0/1	0/1	0/0	0/0
	RC1	normal	3/3	2/3	2/4	0/2	0/2	0/1	0/1	0/0	0/0
		uniforme	3/3	2/3	2/4	0/2	0/2	0/1	0/1	0/0	0/0

Tabela 2: Resumo da execução das instâncias pelo Gurobi

Tipo de Instância	Não Resolvidas	Resolvidas com Sucesso	Total
1	60	30	90
2	54	42	96
Total	114	72	186

O algoritmo baseado na Busca Tabu, por outro lado, cumpriu seu objetivo e retornou boas soluções para todas as instâncias em um curto espaço de tempo. Foram realizados testes estatísticos para avaliar duas hipóteses sobre o desempenho do algoritmo: que o Gurobi supera a meta-heurística implementada quando existe uma solução e o contrário quando o tamanho das instâncias aumentam. Essas hipóteses foram testadas usando comparações estatísticas gerais por método de solução e análises detalhadas por tamanho de instância. Vale ressaltar que essas análises consideraram apenas as instâncias para as quais o Gurobi retornou alguma solução. Os resultados da Tabela 3 mostraram diferenças estatisticamente significantes no GAP (%) — calculado através da Equação (18) - entre os métodos. Dessa forma, a primeira hipótese foi confirmada, pois o Gurobi superou o algoritmo baseado na Busca Tabu quando encontrou solução, segundo os testes estatísticos.

$$GAP = \frac{|\text{Limitante Superior} - \text{Valor da solução}|}{|\text{Limitante Superior}|} \quad (18)$$

Tabela 3: Teste Kruskal-Wallis da mediana do gap (%)

Método de Solução	Mediana do Gap (%)	H_STAT	Valor-p
Gurobi	70.24	4.57139	0.03251
Tabu	78.50		

No entanto, conforme mostrado na Figura 3, o boxplot da diferença no valor da função objetivo entre os métodos em função do número de nós, é possível observar que, para instâncias menores, essa diferença tende a ser negativa, indicando a ligeira superioridade do Gurobi. Entretanto, à medida que o número de nós aumenta, surgem valores positivos, revelando que a Busca Tabu não só encontra rapidamente soluções viáveis, mas também supera o *solver* exato em instâncias maiores. Esse comportamento está perfeitamente alinhado com a discussão no parágrafo anterior, no qual foi observado que o Gurobi não conseguiu resolver a maioria das instâncias maiores dentro do tempo e da memória disponíveis, enquanto a heurística geralmente produziu soluções de alta qualidade em um curto espaço de tempo. Consequentemente, a figura ressalta o sucesso da meta-heurística em casos mais complexos, destacando sua relevância prática quando os recursos computacionais são limitados. A Tabela 4 mostra um resumo dos casos em que a Busca Tabu obteve soluções melhores que o Gurobi.

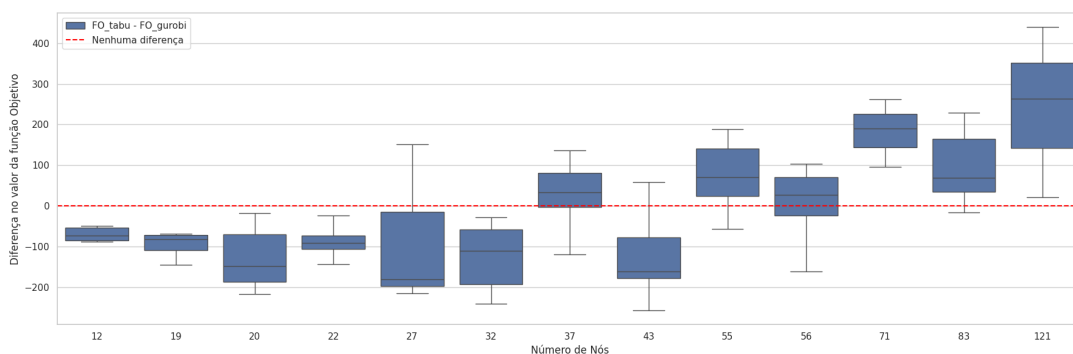
Figura 3: Diferenças nas funções objetivos com o crescimento do número de Nós.

Tabela 4: Comparação de desempenho: Busca Tabu vs. Gurobi por classe

Tipo	Total de Instâncias	Instâncias (Busca Tabu Melhor)	Proporção (%)
Total	72	29	40.28
C1	24	6	25.00
R1	24	13	54.17
RC1	24	10	41.67

O segundo conjunto de testes consistiu em comparações dos métodos de solução conforme diferentes faixas de tamanho, para verificar a segunda hipótese proposta no início desta seção e os indicativos de que ela seja verdadeira de acordo com a Tabela 4 e a Figura 3. A Tabela 5 mostra o resultado de testes de comparação de medianas dos valores da função objetivo obtidos pelo Gurobi e pela Busca Tabu. Os Resultados comprovam que a Busca Tabu continua oferecendo boas soluções com o crescimento do problema e que essas soluções são melhores que as obtidas pelo método exato. Essa diferença fica mais evidente quando o tamanho das instâncias é igual ou maior do que 100 nós.

Tabela 5: Resultados do teste de Mann-Whitney e valores medianos da função objetivo

Tamanho	Método	Valor Mediano da F.O.	U_STAT	Valor-p	Comparado o Gurobi
1-25	Busca Tabu	63.70	193.5	0.05259	Sem diferença
	Gurobi	131.07			
26-50	Busca Tabu	127.74	165.0	0.00044	Gurobi melhor
	Gurobi	272.59			
51-75	Busca Tabu	307.58	326.0	0.97804	Sem diferença
	Gurobi	291.94			
76-100	Busca Tabu	374.85	119.0	0.23298	Sem diferença
	Gurobi	308.34			
101-125	Busca Tabu	550.69	88.0	0.00147	Busca Tabu melhor
	Gurobi	41.07			
126-150	Busca Tabu	759.84	N/A	N/A	Busca Tabu melhor (s/ solução para o Gurobi)
	Gurobi	N/A			
151-175	Busca Tabu	915.12	N/A	N/A	Busca Tabu melhor (s/ solução para o Gurobi)
	Gurobi	N/A			
201-225	Busca Tabu	997.60	N/A	N/A	Busca Tabu melhor (s/ solução para o Gurobi)
	Gurobi	N/A			

6. Conclusões

Este estudo apresentou um modelo de otimização baseado em programação inteira mista e um método heurístico de solução baseado na metaheurística de Busca Tabu para auxiliar no planejamento de itinerários turísticos. Tanto o modelo MIP quanto a meta-heurística podem ser abordagens efetivas para resolver o MPTPPP e otimizar itinerários de viagem, dependendo do tamanho do problema, considerando restrições de janela de tempo e vários períodos. A avaliação experimental realizada em um conjunto diversificado de instâncias forneceu hipóteses sobre o desempenho dos métodos de solução propostos, ressaltando aspectos relacionados à complexidade do problema, às características das instâncias e à eficiência de diferentes técnicas de solução. Essas instâncias podem servir de referência para problemas relacionados em pesquisas futuras e estão disponíveis no repositório: instances_MPTPPP. A Busca Tabu teve seu desenvolvimento justificado dado que que o solver Gurobi não conseguiu resolver o modelo MIP para nem a metade das instâncias e também não forneceu resultados para algumas instâncias relativamente pequenas com 25 a 50 nós, diferentemente da meta-heurística. Verificou-se que tanto a classe das instâncias quanto o número de dias desempenham um papel importante na dificuldade de solução do problema. A classe RC foi a mais difícil de resolver e, quanto maior o número de dias nestas instâncias, maior foi a dificuldade para resolvê-las.

Agradecimentos: Agradecemos ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro fornecido por meio de uma bolsa de iniciação científica do Projeto Universal (405702/2021-3) e à FAPESP pelo apoio parcial ao desenvolvimento deste projeto (processos 2022/05803-3 and 2025/00494-0).

Referências

- Aliano Filho, Angelo e Reinaldo Morabito (2024). «An effective approach for bi-objective multi-period touristic itinerary planning». Em: *Expert Systems with Applications* 240, p. 122437. ISSN: 0957-4174.
- Bayliss, Christopher (jul. de 2021). «Machine learning based simulation optimisation for urban routing problems». en. Em: *Applied Soft Computing* 105, p. 107269. (Acedido em 11/04/2025).
- Gao, F., K. Han e K. Xu (2018). «Utility-Aware Trip Planning». Em: *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pp. 865–872.
- Glover, Fred e Manuel Laguna (1997). *Tabu Search*. en. Boston, MA: Springer US. DOI: 10.1007/978-1-4615-6089-0. (Acedido em 20/08/2025).
- Gonçalves, V. A. et al. (2017). «ALGORITMO GENÉTICO BASEADO EM CHAVES ALEATÓRIAS PARA O ORIENTEERING PROBLEM WITH TIME WINDOWS». Em: *Revista Interdisciplinar de Pesquisa em Engenharia* 2.10, pp. 90–104.
- International Labour Organization (2023). *How data can bolster decent work in the tourism sector*. (Acedido em 14/04/2025).
- Jansson, André (2022). «Guided by data: A logistical approach to tourism in the platform economy». en. Em: *Digital Geography and Society* 3, p. 100040. ISSN: 26663783.
- Pureza, Vitória, Reinaldo Morabito e Marc Reimann (mai. de 2012). «Vehicle routing with multiple deliverymen: Modeling and heuristic approaches for the VRPTW». en. Em: *European Journal of Operational Research* 218.3, pp. 636–647.
- Rosenkrantz, Daniel, Richard Stearns e Philip II (set. de 1977). «An Analysis of Several Heuristics for the Traveling Salesman Problem». Em: *SIAM J. Comput.* 6, pp. 563–581.
- Silva, Admilson Alcântara, Reinaldo Morabito e Vitória Pureza (2018). «Optimization approaches to support the planning and analysis of travel itineraries». Em: *Expert System with Applications* 112.
- Solomon, Marius M. (1987). «Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints». Em: *Operations Research* 35.2, pp. 254–265.
- Tuo, Yanzheng et al. (mar. de 2025). «Artificial intelligence in tourism: insights and future research agenda». en. Em: *Tourism Review* 80.4, pp. 793–812.
- UN News (2022). *Interview: Time to rethink tourism, an economic lifeline for millions*.
- UNWTO (2024). *International tourism recovers pre-pandemic levels in 2024*.
- Zhang, Yu et al. (2019). «Is tourism really affected by logistical operations and environmental degradation? An empirical study from the perspective of Thailand». en. Em: *Journal of Cleaner Production* 227, pp. 158–166.