Geração de Roteiros Turísticos Personalizados em Tempo Real através de Heurística GRASP Adaptada

Lucas G. S. Félix[1,3], Washington Cunha[1], Carolina Ribeiro Xavier[2], Pedro Olmo Vaz de Melo [1], Marcos André Gonçalves[1], Jussara Marques Almeida [1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)

[2]Departamento de Ciência da Computação – Universidade Federal de São João del Rei (UFSJ)

[3]INRIA-Saclay - France

{lucas.felix, washingtoncunha, olmo, mgoncalv, jussara}@dcc.ufmg.br

carolinaxavier@ufsj.edu.br

Abstract. *Planning a tourist trip is a challenging task as it involves reconciling the user's personal interests with financial, logistical, and categorical constraints. This problem is formally known as the Tourist Problem and is often addressed as an optimization problem. Exact methods often fail to find optimal solutions within a reasonable time for this problem, making metaheuristics a viable alternative. In this study, we propose a GRASP-based approach to solve the Tourist Problem. Our technique stands out by considering not only the utility of the visited locations but also their geographic proximity, encouraging the exploration of neighboring areas. Besides, our approach was evaluated using real data from the city of Rio de Janeiro, achieving superior performance compared to other techniques in the literature, with up to a 60% improvement in utility and low computational cost, generating routes in less than 1 second.*

Resumo. *Planejar uma viagem turística é uma tarefa desafiadora, pois envolve conciliar os interesses pessoais do usuário com restrições financeiras, logísticas e categóricas. Esse problema é formalmente conhecido como o Problema do Turista e é frequentemente tratado como um problema de otimização. Métodos exatos frequentemente falham em encontrar soluções ótimas em tempo hábil para este problema, o que torna meta-heurísticas uma alternativa viável. Neste trabalho, propomos uma abordagem baseada em GRASP para solucionar o Problema do Turista. Nossa técnica se diferencia por considerar não apenas a utilidade dos locais visitados, mas também a proximidade geográfica, promovendo a exploração de áreas vizinhas. Além disso, nossa abordagem foi avaliada com dados reais da cidade do Rio de Janeiro, alcançando um desempenho superior a outras técnicas da literatura, com melhoria de até 60% em utilidade, e um baixo custo computacional, gerando rotas em menos de 1 segundo.*

1. Introdução

Com a expansão da internet, a indústria do turismo passou por uma revitalização, permitindo que os usuários se conectassem por meio de comunidades online, como blogs de viagens e redes sociais, facilitando tanto a busca por pontos de interesse (POIs) quanto a troca de informações sobre locais e experiências [Kotiloglu et al. 2017]. Embora essas plataformas concentrem um volume significativo de informações relevantes para o planejamento de viagens, é comum que usuários encontrem dificuldades decorrentes da vasta quantidade de dados disponíveis, o que dificulta a seleção da opção mais adequada, um

fenômeno conhecido como sobrecarga de informação [Pang et al. 2008]. Uma das maneiras de lidar com esse problema é via sistemas de recomendação (SRs), os quais oferecem sugestões personalizadas baseadas no histórico de visitas dos usuários. Contudo, esses sistemas geralmente não abrangem aspectos fundamentais de uma viagem turística, como restrições de preço, horário de funcionamento e distância, que são igualmente cruciais para uma experiência satisfatória do usuário [Kotiloglu et al. 2017].

Para simplificar o planejamento de viagens, muitos estudos buscam automatizar esse processo, abordando-o como um problema de otimização. No geral, este problema turístico é mapeado como uma instância do Problema de Orientação (PO), o qual é descrito como uma combinação do problema do caixeiro-viajante e o problema da mochila [Gavalas et al. 2014]. De maneira geral, o PO busca encontrar uma rota em um grafo que maximize a utilidade, respeitando uma restrição de distância [Gavalas et al. 2014]. Neste cenário, a utilidade é dada pelo benefício do usuário visitar um certo local, e.g. a nota que um usuário deu ao local[1]. Entretanto, ao considerar cenários reais de turismo, apenas essas restrições não são capazes de satisfazer os requisitos de usuários visto a simplicidade de sua modelagem, considerando apenas uma restrição de distância. Assim, diversos trabalhos na literatura expandem essa modelagem considerando restrições de tempo [Vansteenwegen et al. 2011], disponibilidade (ou não) de múltiplos dias de visitas [Vansteenwegen et al. 2011], e categorias de atrações [Sarkar and Majumder 2021].

Outro fator que torna mais difícil a construção de soluções para turistas é que o PO é um problema NP-difícil [Ruiz-Meza and Montoya-Torres 2022], o que faz encontrar boas soluções em tempo polinomial uma tarefa não-trivial. Além disso, estudos mostram que usuários estão dispostos a aguardar no máximo 1 segundo para terem respostas em aplicações em tempo real [Headspin 2023] (e.g. sistemas de recomendação veiculares). Assim, técnicas que possuem um bom desempenho computacional são essenciais, mesmo que isso signifique prescindir (até certo ponto) da qualidade da solução gerada.

Na literatura, Heurísticas e Meta-heurísticas (HMs) têm sido bastante utilizadas para resolver o PO visto que HMs otimizam o custo-benefício entre o tempo de execução e qualidade das soluções. Neste cenário, diferentes técnicas foram aplicadas para resolver o problema como Algoritmos Genéticos (AG), Colônia de Formigas (CF), e Busca Tabu [Ruiz-Meza and Montoya-Torres 2022]. Contudo, as soluções e técnicas existentes falham em (i) considerar aspectos importantes de uma viagem turística como a seleção de hotéis e múltiplos dias de viagem em suas modelagens [Sarkar and Majumder 2021], tendo pouco uso prático, visto que, no geral, viagens turísticas duram múltiplos dias [Friggstad et al. 2018], (ii) avaliar suas propostas em cenários realistas, com base de dados que refletem a experiências turísticas reais.

Nesse contexto, este trabalho propõe uma meta-heurística voltada para a resolução do problema do turista, com ênfase em cenários realistas e priorizando um baixo custo computacional. Assim, propomos uma modelagem condizente com as demandas dos usuários, que no geral, são negligenciadas na literatura, considerando restrições de múltiplos dias de viagens, seleção de hotéis, restrições monetárias, de distância diária, categorias, e múltiplas visitas. Para tratar o problema propomos uma heurística de solução única baseada no *GRASP* (*Greedy Randomized Adaptive Search Procedure*), que, ao contrário de técnicas populacionais, como Algoritmos Genéticos (AGs) e Colônia de Formigas (CF), que exploram múltiplas soluções simultaneamente, apresenta um custo

[1]Mais formalmente, a **utilidade** de um roteiro é medida pelo somatório das utilidades dos locais visitados, onde a utilidade de um local é dado pela nota que um usuário atribui àquele local.

computacional menor, tornando-a mais eficiente em comparação com outras abordagens exploradas na literatura. Além disso, o *GRASP* se caracteriza por sua simplicidade, o qual funciona por meio de dois passos principais: (i) uma seleção gulosa aleatorizada de locais para compor o roteiro, (ii) busca local a qual tenta substituir partes da trajetória gerada, por outros POIs mais interessantes ao usuário. A seleção gulosa aleatorizada, é no geral guiada por um critério baseado na utilidade dos locais visitados, que por sua vez é pouco viável em cenários reais, visto que os locais de utilidade mais alta podem estar muito longe. Nesse cenário, introduzimos uma seleção baseada no custo-benefício dos locais ponderando a distância e a utilidade desses locais. Ao fazer tal, a seleção gulosa aleatorizada é guiada para locais geograficamente próximos que possuem utilidade altas, simulando turistas que visitam os melhores locais de regiões de uma cidade. Por fim, avaliamos nossa proposta com dados reais coletados da plataforma *TripAdvisor* para a cidade do Rio de Janeiro. A escolha do *TripAdvisor* é motivada por esta ser uma das maiores plataformas de turismo da atualidade, contendo uma vasta gama de informações contextuais dos locais visitados (e.g. preço, categoria, serviços oferecidos), o que permite avaliar melhor nossa proposta em cenários realistas. Por fim, nós comparamos a nossa proposta com diferentes técnicas propostas na literatura, incluindo a proposta de [Kotiloglu et al. 2017].

Nossos resultados experimentais mostram que nossa proposta permite gerar bons roteiros, superando em 60% em termos de utilidade outras técnicas consideradas em nossa avaliação experimental. Além disso, nossa proposta apresenta, no pior cenário, um tempo de execução de 0.84 segundos, indicando, portanto, um baixo custo computacional. Por fim, disponibilizamos tanto o conjunto de dados utilizados quanto o código das técnicas implementadas [2] para garantir a reprodutibilidade dos resultados aqui apresentados.

2. Revisão da Literatura

Conforme discutido anteriormente, este trabalho propõe uma técnica para a resolução do problema do turista. Nesta seção, apresentamos os trabalhos relacionados, organizando-os em três partes: (i) Problema do turista, mostrando como é feita a modelagem deste problema na literatura, (ii) fundamentos em heurísticas e meta-heurísticas, as quais justificam as técnicas que utilizamos neste trabalho, e por último (iii) estudos que abordam o mesmo problema analisado neste trabalho.

2.1. Problema do Turista

Neste trabalho, nós focamos na resolução de uma instância do Problema de Orientação (PO), o qual visa sugerir roteiros que maximizem a utilidade, respeitando as restrições dos usuários e locais.

Formalmente, PO pode ser modelado como a seguir: seja $G = (V, E)$ um grafo no qual cada vértice $v \in V$ possui associado um peso $w_v \in \mathbb{R}_+$ de utilidade, onde $W = \{w_0, w_1, ..., w_v\}$. Dado um nó inicial s, um nó terminal f, onde $s, f \in V$, e um orçamento de tempo positivo B, o objetivo é encontrar um caminho de s para f com comprimento máximo B, de modo que a utilidade total dos nós visitados seja maximizado.

Como mostrado no trabalho de [Ruiz-Meza and Montoya-Torres 2022], o PO é um problema NP-difícil, o que torna a resolução deste problema em um tempo hábil não trivial. Neste cenário, na literatura, este problema foi tratado utilizando heurísticas e meta-heurísticas, as quais nós discutimos a seguir.

[2] Código Github

2.2. Heurísticas e Meta-heurísticas

Em cenários em que a identificação de soluções ótimas em tempo hábil é impossível, heurísticas e meta-heurísticas viabilizam alcançar resultados aceitáveis minimizando o tempo de espera para alcançar esses. Essas técnicas, por sua vez, podem ser categorizadas em: **Meta-heurísticas de solução única**, e **meta-heurísticas populacionais**.

Meta-heurísticas de **solução única** operam com apenas uma solução, que é iterativamente refinada pelo algoritmo ao explorar uma parte limitada do espaço de busca. No entanto, essas técnicas podem sofrer com a estagnação em ótimos locais. Exemplos incluem GRASP, Busca Tabu (BT) e Busca Local Iteradas (BLI).

Em contraste, as **soluções baseadas em população** geram diferentes soluções candidatas iniciais e realizam operações combinatórias sobre essas soluções, visando criar uma solução final que seja melhor que as soluções iniciais [Talbi 2009]. Exemplos incluem, Algoritmo Genético (AG) e Colônia de Formigas. Por iniciarem a partir de múltiplas soluções, as técnicas baseadas em população apresentam maior diversidade e podem alcançar resultados promissores. Contudo, algumas das soluções geradas podem ser inviáveis, resultando em avaliações desnecessárias e um aumento do custo computacional. Além disso, o alto custo computacional dessas técnicas dificulta sua aplicação em cenários de tempo real. Dado que este trabalho busca soluções de baixo custo computacional, focamos em técnicas de **solução única**, cuja aplicação ao problema do turista será discutida na próxima seção.

2.3. Trabalhos relacionados

Na literatura, diferentes propostas atacaram o problema de PO utilizando de **técnicas baseadas em solução única**. A preferência por esse tipo de técnica se dá principalmente pelo baixo custo computacional. Neste cenário, técnicas como GRASP, BT, e BLI foram utilizadas. Podemos categorizar esses trabalhos em 2 tipos: aqueles que realizam avaliações com usuários reais através de uma aplicação *web* e aqueles avaliam sua metodologia apenas por meio da utilidade dos roteiros recomendados.

Os trabalhos [Ruiz-Meza et al. 2021, Vansteenwegen et al. 2011], propõem uma abordagem baseada no *GRASP* para o problema do turista, tendo como principal foco a avaliação com usuários reais. O primeiro trabalho foca na recomendação de roteiros para grupos de turistas, enquanto o segundo avalia sua solução através de diferentes usuários reais por meio de uma aplicação web. Em ambos cenários, os autores confirmam a qualidade do *GRASP* para a resolução do problema, atingindo soluções com boa utilidade e bom desempenho computacional, conforme avaliação dos usuários.

Os trabalhos de [Gavalas et al. 2017, Ghobadi et al. 2023, Aliano Filho and Morabito 2024] utilizam técnicas baseadas em BLI para resolver o problema do turista. Em [Gavalas et al. 2017], os autores focam em aspectos da interação do usuário com aplicações para análise. Já em [Ghobadi et al. 2023, Aliano Filho and Morabito 2024] focam em comparar as rotas recomendadas automaticamente para os usuários com rotas feitas previamente por esses turistas. Vale ressaltar que, em geral, os usuários não realizam escolhas ótimas, o que faz com que a rotas recomendadas sejam melhores que as rotas feitas pelos usuários sem auxílio de ferramentas [Kotiloglu et al. 2017].

Por último, listamos a proposta de [Kotiloglu et al. 2017], a qual utilizamos como *baseline* comparativo. Neste trabalho, os autores usam técnicas de BT com um foco semelhante ao nosso: maximizar concomitantemente a utilidade para o usuário e o desempenho

computacional. O trabalho oferece porém uma modelagem aprofundada dos requisitos dos turistas, considerando aspectos como horários de funcionamento de locais, seleção de hotéis e viagens de múltiplos dias, avaliando a metodologia proposta com dados do *Foursquare*. Ressaltamos, porém, que embora abrangentes, esses dados não apresentam informações contextuais o suficiente, o que levou os autores a usarem parcialmente dados sintéticos em suas avaliações. A seleção deste trabalho como *baseline* é motivada por esse ser um dos poucos trabalhos que possui uma modelagem abrangente dos requisitos de um usuário. Neste cenário, para realizar uma comparação justa, evitando avaliações que não são consideradas em nossa modelagem, removemos restrições que não estão presentes nas nossa, mas estão presentes na modelagem original de [Kotiloglu et al. 2017], incluindo a restrição de horário de funcionamento de locais, e restrição de locais obrigatórios.

Em suma, diferentemente das propostas anteriores, nosso trabalho foca na recomendação de roteiros turísticos preenchendo três principais lacunas deixadas na literatura: (i) a geração de soluções em tempo real, (ii) restrições necessárias para turistas em uma viagem real, e (iii) avaliação em cenários realistas. Nos endereçamos cada um desses problemas da seguinte forma: (i) proposta de uma técnica GRASP, a qual possui menor custo computacional que outras técnicas propostas na literatura, visto que foca em uma única solução, e (ii) modelagem do problema baseado em demandas reais de turistas, por fim, (iii) avaliação da proposta com um conjunto de dados da cidade do Rio de Janeiro, coletado da plataforma *TripAdvisor*, a qual oferece um conjunto único de atributos, e permite uma avaliação realista da nossa proposta .

3. Metodologia

3.1. Modelagem e Restrições do problema

Considere que um turista (usuário) está viajando, por uma cidade, por p dias, visando visitar o máximo de m atrações turísticas todos os dias. O objetivo desse usuário é criar passeios diários em locais que podem ser visitados durante os p dias. Cada local i está associado a uma utilidade $R_i \in \mathbb{R}_+$. Em nosso cenário a utilidade pode ser vista como a nota que aquele usuário deu ao local. Assim, o objetivo do modelo é encontrar o subconjunto de locais disponíveis que **maximize** a utilidade dos locais visitados, enquanto respeitando as restrições do usuário. Sendo assim, a função objetivo é maximizar a utilidade do turista por meio da seleção de locais ótimos para visita. Neste trabalho seguimos restrições similares às consideradas no trabalho de [Kotiloglu et al. 2017]. Assim, consideramos restrições **Monetária** a qual define o quanto o usuário pode gastar ao longo de todos os dias, **Distância Diária** a qual define o quanto o usuário pode andar, em quilômetros, por dia, **Quantidade de Atividades Diárias** a qual define a quantidade máxima de atividades a serem realizadas em um dia, **Categórica** a qual define a quantidade máxima de locais de uma mesma categoria podem ser visitados sequencialmente por dia, e **Múltiplas Visitas:** a qual define que nenhum local, com exceção do hotel, pode ser visitado mais de uma vez ao dia.

3.2. Solução proposta

O *GRASP* é uma meta-heurística [Feo and Resende 1995], que combina elementos de construção gulosa com busca local para resolver problemas de otimização combinatória [Talbi 2009], gerando uma solução viável ao final de seu processo. O *GRASP* se destaca entre outras técnicas por ter uma natureza aleatória que permite a geração de soluções mais diversificadas que outras técnicas de solução única que não possuem nenhum princípio estocástico. De maneira geral, o *GRASP* opera da seguinte forma:

Construção Gulosa Aleatorizada: A partir de solução vazia, é adicionado de maneira gulosa aleatorizada, elementos ao conjunto de soluções. Neste cenário, esses elementos são locais que irão compor a rota. Para que sejam selecionados estes elementos, é feita uma seleção em duas partes: (i) Etapa Gulosa: a qual pré-seleciona elementos de acordo com um critério guloso (usualmente a utilidade), criando uma Lista de Candidatos Restritos (LCR). A seleção destes elementos é controlada por um parâmetro α, que quanto menor mais guloso é algoritmo, e quanto maior mais aleatório. Atuando de forma diferenciada, valores menores de α restringem o tamanho da LCR, favorecendo soluções de maior utilidade, mas com menor diversidade. Por outro lado, valores maiores de α promovem maior diversidade, visto que permitem uma LCR menos restritiva, às custas de uma possível redução na qualidade das soluções. Portanto, ajustar adequadamente esse parâmetro é crucial para alcançar um equilíbrio que permita a geração de soluções ótimas. Por fim, (ii) Etapa Aleatória: A qual seleciona um elemento aleatoriamente dentre a LCR.
Busca Local: Após a construção de uma solução inicial, aplica-se uma busca local para melhorar a solução encontrada na etapa anterior. A busca local explora soluções vizinhas em busca de melhorias. Em nosso cenário, essa busca visa alterar locais que não possuam a melhor utilidade possível e ainda tornam a solução viável. Para isso, selecionamos para substituir locais que possuem a mesma categoria e que são próximos ao local que será substituído, e que tenham uma alta utilidade. Neste caso, ordenamos as melhores soluções pela utilidade do local, testando novos locais que podem ser atribuídos. Este processo é repetido até que um novo local, que melhora a utilidade da solução, seja encontrado. Caso nenhum dos candidatos selecionados melhore a solução e ainda a deixa viável, então passamos para o próximo local em ordem de utilidade.
Iterações: Repete as etapas de construção e busca local por um número definido de iterações, mantendo a melhor solução encontrada.

Propostas prévias presentes na literatura que tratam o problema do turista utilizando a meta-heurística *GRASP* [Brito et al. 2017, Expósito et al. 2019, Ruiz-Meza et al. 2022], no geral baseiam sua política de construção da LCR em sua função objetivo, ou seja, as soluções são selecionadas apenas com base na utilidade dos locais visitados. Entretanto, em alguns cenários, essa pode não ser a melhor solução, visto que além de maximizar a utilidade da solução, é necessário respeitar as restrições. Em particular, em cenários reais de aplicação, a distância entre os pontos de interesse pode ser significativa, o que frequentemente resulta na geração de soluções candidatas inviáveis pelas heurísticas, visto que o usuário possui restrições de tempo e limite de quilômetros deslocados em um dia. Para resolver esse problema, propomos neste trabalho uma abordagem que seleciona os elementos da LCR considerando não apenas a utilidade, denotada como R, mas também a distância percorrida. Essa distância é calculada a partir de uma matriz quadrática de distância D, que contém as distâncias reais entre todos os pontos do conjunto de dados. Neste caso, dado a distância percorrida ($D_{i,j}$) e a utilidade (R_j) que o local possui, o local é selecionado para compor a LCR baseado no custo de deslocamento pela utilidade (c) calculado na Equação[1], onde i é o local imediatamente anterior visitado a j. Neste cenário, os locais que são mais próximos e possuem maior utilidade possuem maior chance de serem escolhidos dentro da solução. Quanto menor o valor de c, melhor é o custo de se visitar aquele local. Por fim, os elementos que compõe a LCR são selecionados caso eles pertençam ao intervalo [c^{min}, β], sendo β o custo máximo aceito para um elemento ser selecionado, sendo este calculado pela Equação[2].

Note que a Equação [2], possui como parâmetro o valor α, o qual, como discutido, possui um grande impacto na qualidade das soluções geradas. Como definir o valor de

α pode ser uma tarefa árdua, realizamos a seleção de α de maneira adaptativa, seguinte a proposta de [Talbi 2009]. Neste cenário, nas m primeiras iterações, o α é selecionado de conjunto de valores possíveis $A = \{\alpha_1, ..., \alpha_m\}$. Inicialmente, a probabilidade de associar com cada α_i é de $p_i = 1/m$, $i \in [1, ..., m]$. Depois das m primeiras iterações a probabilidade de cada α é atualizada de acordo com a qualidade das soluções obtidas. Considere que r^* seja a utilidade de solução encontrada e $AVG(\alpha_i)$ a média de todas as soluções encontradas usando $\alpha = \alpha_i$. Então, a probabilidade p_i para cada valor de α é atualizado da seguinte maneira:

$$c_j^i = \frac{D_{i,j}}{R_j} \tag{1}$$

$$\beta = c^{min} + \alpha(c^{max} - c^{min}) \tag{2}$$

$$p_i = \frac{q_i}{\sum_{j=1}^{m} q_j}, i \in [1, ..., m] \tag{3}$$

Onde $q_j = r^*/AVG(\alpha_i)$. Assim, valores maiores de p_i correspondem a valores melhores para o parâmetro α_i.

Estrutura da solução: A solução gerada pelo GRASP é representada por um vetor $S = [h, p_0, ..., p_n]$, onde h corresponde ao hotel, fixo na posição inicial do vetor, e p_i representa os pontos de interesse. Os locais visitados pelo usuário devem ser percorridos na ordem especificada em S. Caso algum local não possa ser visitado em um dia devido à violação de alguma restrição, o dia atual é finalizado com uma visita ao hotel, e um novo dia é iniciado, sendo o ponto de partida de cada dia novamente o hotel.

Busca Local: Na etapa de busca local, utilizamos operações baseadas no problema da mochila, visando substituir locais que possuem utilidade mais baixas. Neste caso, o critério para substituição é baseado na utilidade do local. Para locais que possuem uma utilidade menor que a maior utilidade possível, verificamos locais de mesma categoria (respeitando a restrição de categoria), que possui uma nota melhor e que são próximos ao último local visitado (respeitando a restrição de distância), garantindo assim a viabilidade da nova solução gerada.

4. Procedimentos Experimentais

4.1. Coleta de dados

Categoria	# de Pontos	Preço	Nota
Hotéis	648	339.63 ($\pm$ 59.88)	3.89 ($\pm$ 0.10)
Atrações	954	0 ($\pm$ 0.0)	3.28 ($\pm$ 0.22)
Restaurantes	564	124.47 ($\pm$ 24.26)	4.19 ($\pm$ 0.07)

Tabela 1. Distribuição dos valores nas colunas

Para realização da avaliação experimental escolhemos a cidade do Rio de Janeiro, uma metrópole com uma vasta gama de pontos turísticos. Para aquisição dos dados foi desenvolvido um *crawler* para coletar dados do TripAdvisor, incluindo informações de nota média e nota dada pelo usuário (sendo a nota média a utilidade dada a cada local - os valores de notas variam entre 1-5), preço mínimo e máximo de cada local dado em reais (utilizamos neste trabalho sempre o preço máximo, considerando o pior cenário), categoria, e geo-coordenadas (latitude e longitude). Vale ressaltar que outros atributos como avaliações escritas, tempo recomendado no local, atributos de restaurantes (e.g. tipo

de comida oferecida, tipo de serviço) também foram coletados, contudo, esses atributos não foram explorados neste trabalho, porém pretendemos utilizá-los em trabalhos futuros. Os foram coletados para hotéis, restaurantes e atrações (e.g. praias, teatros e pontos de interesse). A Tabela 1, mostra o valor médio dos atributos coletados juntamente com o Intervalo de Confiança (IC) de 95 %. Um ponto que é possível perceber é que não há valores para as atrações, isso ocorre devido a uma limitação da plataforma, o qual não permite aos usuários colocarem preços de atrações. Além disso, dada a grande quantidade de atrações públicas no Rio de Janeiro como praias (34), igrejas (32), museus (71), e pontos de interesse (e.g. Morro da Urca, Praça Mauá) (73), muitas dessas atrações teriam um preço pequeno ou irrisório ao considerar o valor da viagem.

A coleta resultou num total de 2166 pontos de interesse distintos. Esses dados, inicialmente, não-estruturados foram convertidos em um formato semi-estruturado (e.g. CSV) após o *parsing* e pré-processamento das páginas coletadas. Nossa coleta de dados possui dados de fevereiro de 2004 à agosto de 2020, porém, neste trabalho, não utilizamos os dados no nível de usuário, mas sim no nível de local, o que significa que, ao invés de utilizarmos a nota individual do usuário para definir R_i (utilidade de cada local), utilizamos a média das notas dadas aos locais.

Por fim, foi calculada de maneira *offline* a distâncias entre esses locais, afim de construir a matriz de distâncias D. Para cada par de pontos de interesse foi calculada a distância entre eles utilizando a biblioteca *OpenStreet Maps*, que considera rotas reais em vez de distâncias geodésicas, trazendo assim maior realismo para os cenários avaliados. Em média, a distância entre os pontos é de cerca de 27 quilômetros.

4.2. Baselines

A seguir, apresentamos os *baselines* utilizados para comparação com a técnica proposta. Além dessas abordagens, incluímos também um limite superior teórico para as instâncias avaliadas. Os *baselines* testados são:

- **Aleatório (*Random*):** O algoritmo aleatório gera soluções de forma estocástica até encontrar uma solução viável. O algoritmo gera uma solução aleatória considerando todos os hotéis e locais disponíveis. Desses hotéis, 1 é selecionado. Dos locais, são selecionados de maneira aleatória a quantidade máxima de locais que podem ser visitados pelo usuário a cada dia. Em seguida a nova solução é avaliada, caso ela não seja viável, são removidos para cada um dos dias inviáveis os últimos locais a serem visitados. Esse processo se repete enquanto a solução não for viável. Embora essa abordagem seja eficiente, é considerada ingênua, pois não garante a geração de boas soluções, e estabelece um limite inferior para a qualidade das soluções produzidas.
- **Guloso (Greedy):** O algoritmo guloso segue a lógica de um turista se perguntando *"Qual é o melhor lugar viável que posso visitar agora?"*. Sendo assim, o algoritmo guloso é uma abordagem construtiva que seleciona a melhor solução candidata viável disponível no momento. No momento em que ao adicionar um local a solução deixa de ser viável, então o turista volta para o hotel. Assim como o algoritmo aleatório, essa abordagem também é considerada ingênua e oferece um limite comparativo para a qualidade da solução gerada.
- **Busca Tabu (BT):** Por fim, implementamos a busca tabu iterada proposta no trabalho [Kotiloglu et al. 2017]. Neste trabalho implementamos uma versão que remove as restrições de janela de tempo, e locais obrigatórios proposta no BT, visto que não consideramos tais restrições em nosso problema.

Ressaltamos também que modelamos o nosso problema também no solver comercial CPLEX [3], contudo, este não foi apto de encontrar soluções dentro de um limite de tempo computacional de 30 minutos. Em nossa avaliação foi possível identificar que com uma quantidade de locais máxima de 400, ainda é possível gerar roteiros, contudo, ao passar disso o tempo de execução se torna inviável, motivando ainda mais o uso de meta-heurísticas.

Por fim, apresentamos um *baseline* teórico, o qual permite calcular um limite superior para qualidade de soluções, dado pela Função (4). Nela temos que o limite teórico para a rota gerada é dada se o usuário visita a quantidade máxima de locais (q) em um dia ao longo de todos seus dias de viagem (d), e que todos os locais que o mesmo visitou tivessem utilidade máxima (nota 5). Esse valor é acrescido de 5, sendo esta a utilidade do hotel que é contado apenas uma vez.

$$t = (5 \times d \times q) + 5 \tag{4}$$

4.3. Perfis de usuários

Dado que neste trabalho não são realizadas avaliações com usuários reais, para realização da avaliação experimental foram criados cenários fictícios representando possíveis perfis de usuários definidos a partir de parâmetros usados na solução. Os parâmetros podem ser categorizados em 2: fixos e variáveis. Os fixos incluem a duração da viagem (2 dias), número máximo de locais diários (5 locais) e visitas consecutivas por categoria (3 visitas). Já as variáveis são os quilômetros por dia e o orçamento (em reais), criando diferentes perfis de usuário, os quais são apresentados na Tabela 2. O valor de orçamento monetário foi baseado no custo médio de hotéis de restaurantes, o qual somados dão um valor aproximado de 450 reais, neste caso, para o perfil mais econômico (A), o usuário teria que optar por locais mais baratos para que consiga fazer uma viagem dentro do orçamento planejado. Além disso, o usuário teria que encontrar restaurantes e atrações que estejam próximos a seu hotel visto que não pode caminhar muito para evitar extrapolar a restrição de distância.

Perfil	Orçamento Monetário	# KMs diário	Descrição
A	R$ 500	10	Gasto pouco, anda pouco
B	R$ 500	20	Gasto pouco, anda muito
C	R$ 1000	10	Gasto médio, anda pouco
D	R$ 1000	20	Gasto médio, anda muito
E	R$ 2500	10	Gasto muito, anda pouco
F	R$ 2500	20	Gasto muito, anda muito

Tabela 2. Diferentes perfis testados

4.4. Avaliação experimental

Nós avaliamos as técnicas nos orientando por duas métricas principais: a qualidade das soluções geradas por cada uma das técnicas em termos de utilidade da solução, e pelo custo computacional dado pelo tempo de execução para gerar cada uma das técnicas. Para realizar a avaliação, cada um dos algoritmos foi executados 100 vezes, e são comparados utilizando um teste-t pareado com 95% de confiança, além da correção de Bonferroni devido à comparação de múltiplos algoritmos [Cunha et al. 2021, Cunha et al. 2025]. Ao realizarmos as comparações, todos os resultados foram avaliados dentro de uma máquina com as especificações apresentadas na Tabela 3, sendo a máquina isolada ao realizar a

[3]https://www.ibm.com/products/ilog-cplex-optimization-studio

avaliação. Estes testes foram realizados para cada um dos perfis descritos acima em todas as técnicas avaliadas. Por fim, ressaltamos que algoritmo *Greedy*, por ser um algoritmo determinístico, não temos variação na utilidade, apenas no tempo de execução.

Sistema Operacional	Memória RAM	CPU	Threads
Ubuntu 18	94 GB	Intel(R) Xeon(R) CPU E5620 2.40GHz	16

Tabela 3. Configuração de computador utilizado

5. Parametrização das técnicas

Nesta seção, discutimos o processo de parametrização das técnicas avaliadas. Conforme mencionado anteriormente, este trabalho busca resolver o problema do turista por meio da aplicação de diferentes abordagens. Entretanto, destacamos que, entre as técnicas analisadas, apenas o *GRASP* e a Busca Tabu possuem parâmetros que influenciam diretamente seu funcionamento. Assim, apresentamos, a seguir, os resultados obtidos na parametrização dessas duas técnicas.

5.1. GRASP

A técnica *GRASP* possui como parâmetros o α, e a quantidade de iterações. Visto que o α é calculado de maneira adaptativa, a quantidade de iterações é o único parâmetro que impacta em nossa proposta. Para quantidade de iterações foram testados com 10, 50, e 100 iterações, porém, caso tenhamos 10 iterações seguidas onde não há melhoria na qualidade da melhor solução, o algoritmo é finalizado.

A Tabela 4, apresenta o impacto da quantidade de iterações nos resultados alcançados no perfil A e F, visto que são os dois perfis mais distintos dentre os avaliados. É possível perceber que ao aumentar a quantidade de iterações temos um aumento na utilidade, mas também no tempo de execução do algoritmo. O aumento da utilidade é justificada pela partida múltipla do *GRASP*, que, devido à maior quantidade de iterações, consegue avaliar melhor o espaço de busca e encontrar melhores soluções. Consequentemente, o tempo de execução aumenta visto que o processo é repetido uma maior quantidade de vezes. Como os resultados obtidos possuem tempos de execução menores que o estipulado como ideal (1 segundo), em todos os cenários, utilizamos como parâmetro em nossas próximas avaliações a quantidade de iterações igual a 100.

Perfil	A			F		
# Iterações	10	50	100	10	50	100
Utilidade	39.97 ($\pm$ 0.80)	42.015 ($\pm$ 0.35)	42.57 ($\pm$ 0.3)	42.42 ($\pm$ 0.34)	43.45 ($\pm$ 0.22)	43.67 ($\pm$ 0.16)
Tempo (s)	0.06 ($\pm$ 0.0)	0.31 ($\pm$ 0.001)	0.63 ($\pm$ 0.01)	0.08 ($\pm$ 0.0)	0.41 ($\pm$ 0.0)	0.83 ($\pm$ 0.0)

Tabela 4. Utilidade e tempo de execução para diferentes parâmetros do *GRASP*

5.2. Busca Tabu

No método proposto por [Kotiloglu et al. 2017], a busca tabu utiliza dois parâmetros principais: o tamanho da lista tabu e o mecanismo de perturbação. O tamanho da lista tabu evita revisitar locais já visitados por um número fixo de iterações, enquanto o mecanismo de perturbação remove aleatoriamente um subconjunto de locais com base em uma probabilidade. O estudo de [Kotiloglu et al. 2017] sugere valores padrão de tamanho da lista tabu igual a 40 e probabilidade de remoção igual a 0.5. Para este trabalho, além desses valores padrão, foram utilizados um tamanho de lista tabu de 20 e probabilidade de remoção de 0.25. Estes valores, em teoria, dão a BT vantagem em tempo computacional, dado que as alterações nessa busca são menores.

Parâmetros		Perfil A		Perfil F	
Tamanho da Lista	**Prob. Remoção**	**Utilidade**	**Tempo (s)**	**Utilidade**	**Tempo (s)**
20	0.25	33.23 (± 1.25)	36.06 (± 1.73)	37.43 (± 0.78)	41.11 (± 0.25)
40	0.25	32.85 (± 2.09)	40.07 (± 0.26)	37.77 (± 0.74)	41.1 (± 0.18)
20	0.50	33.42 (± 2.10)	39.72 (± 0.22)	37.52 (± 0.78)	40.51 (± 0.22)
40	0.50	32.56 (± 2.16)	39.72 (± 0.22)	37.7 (± 0.70)	40.45 (± 0.20)

Tabela 5. Parâmetros avaliados na técnica de Busca Tabu

A Tabela 5 apresenta a comparação dos resultados da parametrização para a técnica BT. Neste cenário, temos que com relação à utilidade os resultados são estatisticamente equivalentes com 95% de confiança. De maneira similar, é possível ver pouca variação com relação ao tempo de execução, apesar da diferença estatística entre esses. Logo, não podemos afirmar que os parâmetros possuem poucos resultados da Busca Tabu. Acreditamos que este baixo impacto se deve ao fato das operações de busca local (e.g. *intra e inter 2-opt, 1-0 Relocate e 1-1 Exchange*) terem um pouco efeito dentro da instância que testamos. Isso ocorre porque técnicas como o *2-opt* apenas realizam a redução no custo de distância da rota, o que não é um objetivo de nossa proposta. A técnica de *1-0 Relocate* remove um local da rota, o que apenas reduz a mesma em custo, distância e utilidade. Por último, a técnica de *1-1 Exchange* adiciona um ponto qualquer a rota, sendo a única abordagem em pode haver um ganho em utilidade. Contudo, em um cenário como o do Rio de Janeiro, essa pode não ser a melhor estratégia, visto que a maior distância que consideramos é de 20 quilômetros por dia (Perfil F), enquanto a distância média entre todos os pontos disponíveis é de 27 quilômetros, o que faz com que na grande maioria dos casos os resultados sejam inviáveis. Por último, vale ressaltar que, dentro do contexto do trabalho de [Kotiloglu et al. 2017] as técnicas de busca local fazem sentido, uma vez que restrições de janela de tempo também são tratadas dentro da proposta. Visto que os resultados são estatisticamente equivalentes, selecionamos como parâmetros padrão em nossas próximas avaliações a lista tabu com tamanho 20 e a probabilidade de remoção igual a 0.25, dado que este cenário apresenta resultados melhores ou estaticamente equivalentes em tempo computacional nos cenários avaliados.

6. Avaliação Experimental

Nesta seção apresentamos os resultados alcançados pela metodologia proposta. Passamos primeiro pela etapa de avaliação das técnicas que utilizamos para resolver o problema do turista, apresentando então resultados com relação ao tempo e à utilidade. Por último realizamos uma breve discussão sobre os resultados alcançados.

	Utilidade				**Tempo (s)**			
Perfil	**GRASP**	**Greedy**	**Busca Tabu**	**Random**	**GRASP**	**Greedy**	**Busca Tabu**	**Random**
A	42.57 (± 0.30) ▲	18.50 (± 0.00)	33.23 (± 1.25)	12.40 (± 1.23)	0.63 (± 0.01)	0.05 (± 0.00)	36.06 (± 1.73)	0.02 (± 0.00) ▲
B	43.02 (± 0.25) ▲	19.00 (± 0.00)	34.43 (± 1.47)	14.47 (± 1.92)	0.68 (± 0.01)	0.05 (± 0.00)	40.30 (± 0.26)	0.01 (± 0.00) ▲
C	43.17 (± 0.20) ●	43.0 (± 0.00)	36.10 (± 1.11)	12.21 (± 1.06)	0.74 (± 0.01)	0.05 (± 0.00)	40.59 (± 0.27)	0.02 (± 0.01) ▲
D	43.54 (± 0.19)	44.5 (± 0.00) ▲	36.65 (± 1.16)	16.12 (± 2.37)	0.77 (± 0.01)	0.05 (± 0.00)	40.75 (± 0.24)	0.00 (± 0.00) ▲
E	43.38 (± 0.21) ▲	43.00 (± 0.00)	37.15 (± 0.98)	12.60 (± 1.37)	0.83 (± 0.01)	0.05 (± 0.00)	40.88 (± 0.21)	0.01 (± 0.00) ▲
F	43.67 (± 0.16)	44.5 (± 0.00) ▲	37.43 (± 0.78)	17.66 (± 2.83)	0.84 (± 0.00)	0.05 (± 0.00)	41.11 (± 0.25)	0.00 (± 0.00) ▲

Tabela 6. Utilidade e tempo de execução das técnicas avaliadas por perfil.

A Tabela 6 mostra os IC em relação à utilidade, e tempo de execução, respectivamente, para cada uma das técnicas avaliadas em nosso trabalho. São indicados nas células da Tabela aqueles valores que são estatisticamente superiores (▲), e os que são estatisticamente empatados (●).

6.1. Avaliação de Desempenho Computacional

Ao avaliarmos na Tabela 6 o tempo de execução gasto por cada um dos algoritmos, podemos ver que, com exceção da técnica Busca Tabu, todos os algoritmos possuem tempo

de execução menor que 1 segundo, podendo portanto serem utilizadas em uma aplicação em tempo real. Dentre os fatores que levam a técnica Busca Tabu a ter um tempo de execução superior estão o fato de ser uma técnica de inicialização múltipla, como proposto em [Kotiloglu et al. 2017], mas também, o alto custo para as técnicas de busca local, e o fato de termos que avaliar todas as restrições após feita a operação de busca local. Isso faz com que o custo sob esta execução esteja mais associado a reavaliação da solução. Isso poderia ser mitigado caso fosse feita a atualização dos atributos da rota (e.g. custos, distância, quantidade de locais por categoria) ao longo da execução. Contudo, dentro de nosso entendimento, isso não é apontado explicitamente pelos autores como algo feito ao longo da busca local e, portanto, acreditamos ser um potencial trabalho futuro.

As técnicas ingênuas (*Greedy, Random*) possuem tempo de execução baixo em comparação às outras técnicas. Isso se dá pelo fato de realizarem a exploração sobre apenas uma solução. Já a nossa técnica *GRASP*, apesar de ter um custo um pouco mais elevado que técnicas ingênuas, ainda possui tempo de execução dentro do esperado. Um dos fatores que impactam em um maior tempo de execução do *GRASP* é o fato de ser uma técnica de inicialização múltipla, que neste cenário faz com que 100 soluções sejam avaliadas. Além disso, podemos perceber que a medida que a quantidade de quilômetros deslocados por dia aumenta, o tempo de execução de nossa proposta também tem um acréscimo. Isso ocorre, pois, a medida que aumentamos essa característica do perfil, consequentemente elevamos a lista de soluções candidatas, o que impacta no custo computacional da técnica. Para evitarmos problemas de tempo de execução em cenários em que a quantidade de quilômetros deslocados por dia aumenta, seria possível realizar a avaliação dentro do critério de seleção de locais para a LRC ser baseada também em custo, ou algum fator de preferência do usuário. Vislumbramos essa estensão como um trabalho futuro.

6.2. Avaliação da Função Objetivo (Utilidade)

Na Tabela 6 também é possível avaliar o IC da utilidade de cada técnica avaliada em nosso trabalho. Neste cenário, apresentamos também como valor de referência para um ótimo teórico o resultado da Equação 4, o qual é igual a 55. Vale ressaltar que este o valor para uma solução ótima teórica, visto que não se sabe se há uma solução que satisfaça todas as restrições e seja composta apenas de locais com nota igual a 5. Neste cenário, quanto mais próxima é a utilidade dos algoritmos desse ótimo teórico, melhor é a técnica.

Ao avaliarmos os resultados é possível ver que a técnica *Random* possui um baixo valor de utilidade. Isto se deve ao fato da escolha aleatória de locais levar a distâncias percorridas muito altas, dado a alta média de distância entre locais (27 quilômetros). O mesmo ocorre com as buscas locais realizadas pela técnica de Busca Tabu, como discutido na Subseção 5. A diferença é que a inicialização da meta-heurística de Busca Tabu é baseada em uma abordagem aleatorizada-gulosa, assim como o *GRASP*. Logo, a solução inicial gerada já possui um resultado melhor que uma solução inicial totalmente aleatória.

A diferença entre os resultados da Busca Tabu e do *GRASP* se dá justamente pelo fato da busca local aliada à seleção automática de α conseguirem gerar soluções viáveis, as quais focam nos itens que possuem uma nota baixa, substituindo-os por candidatos viáveis que possuem nota maior. Já a busca local do BT realiza seleções aleatorizadas, as quais não levam necessariamente a substituição de locais de menor utilidade.

Por fim, o algoritmo *Greedy*, apesar de ser uma heurística simples, apresenta bons resultados para cenários em que o custo monetário não é um problema. Os resultados ruins nos perfis A e B, que possuem um orçamento de 500 reais, estão relacionados a

seleção do hotel realizada pelo algoritmo. Nesse cenário, uma parcela significativa do orçamento é destinada à seleção do hotel, com um gasto de 214 reais em duas diárias, representando 85.6% do total. Isso deixa recursos mínimos para serem alocados em atrações e restaurantes. Contudo, em cenários menos restritivos monetariamente (*budget* ≥ 1000), o algoritmo guloso consegue ter resultados um pouco superiores ao algoritmo *GRASP* para os perfis E e F, considerando um teste-t pareado com 95% de confiança.

Por fim, ao compararmos com nossa proposta, observamos que, embora o método *Greedy* apresente um bom desempenho em cenários menos restritivos e excelente eficiência computacional, nossa abordagem também se destaca por sua eficiência computacional e por alcançar resultados muito próximos aos melhores encontrados, além de se aproximar consistentemente do ótimo. Além disso, por ser uma abordagem que possui princípio aleatorizado, o GRASP traz ao usuário uma diversificação das soluções, gerando diversas soluções que possuem qualidade, e não ficando restritas a apenas uma solução, como acontece em técnicas determinístico (e.g. *Greedy*). Finalmente, ressaltamos que a técnica proposta pode ser facilmente modificada para executar em paralelo, o que pode nos resultar em tempos de execução ainda menores.

7. Conclusões

Neste trabalho, propomos uma técnica baseada em meta-heurísticas para o Problema de Orientação, o qual é uma combinação do problema da mochila e do problema do caixeiro-viajante, tendo sido amplamente estudado no contexto de Turismo. Neste cenário, o objetivo é encontrar uma rota que maximize as preferências e respeite as restrições dos turistas. Neste estudo propomos uma meta-heurística GRASP e a comparamos com outras heurísticas (*Greedy, Random, Busca Tabu*).

Neste cenário, avaliamos a técnica proposta, juntamente com as abordagens da literatura utilizando dados reais da cidade do Rio de Janeiro do TripAdvisor. Nos resultados obtidos com uma base de dados contendo mais de 2000 locais, ficou clara a falta de escalabilidade de métodos determinísticos baseados em *solvers* comerciais para situações com mais de 400 locais, motivando a necessidade de abordagens meta-heurísticas para resolver eficientemente esse problema. As heurísticas testadas se destacaram com um tempo de processamento viável – com exceção do *baseline* de Busca Tabu, todas as outras três meta-heurísticas avaliadas produziram um desempenho computacional que respeita o tempo de execução imposto para aplicações reais.

Além disso, os resultados mostram que a técnica GRASP proposta apresenta os melhores resultados em 3 dos 5 cenários avaliados, mostrando grande potencial para utilização em cenários reais por seu bom desempenho computacional, e resultados superiores em cenários onde o orçamento é restrito. Outra técnica que se destaca é o algoritmo *Greedy*, o qual possui melhores resultados em cenários com abundância de recursos. No entanto, em situações com maior restrição financeira, o *Greedy* tende a não gerar rotas eficientes, pois gasta grande parte do orçamento em hospedagem.

Como direções para trabalhos futuros iremos aprofundar os testes da heurística *Greedy* para identificar cenários onde essa técnica não é eficaz. Explorar novas restrições, como restrições de janelas de tempo, também se mostra relevante. Por fim, avaliar nossa proposta em diferentes cenários e com usuários reais.

Agradecimentos Este trabalho foi financiado pelo CNPq, FAPEMIG, CAPES, INCT-TILD-IAR, CAPES-STIC-AMSUD 22-STIC-07 projeto LINT, FAPESP, e AWS.

Referências

Aliano Filho, A. and Morabito, R. (2024). An effective approach for bi-objective multi-period touristic itinerary planning. *Expert Systems with Applications*, 240:122437.

Brito, J., Expíósito-Márquez, A., and Moreno, J. A. (2017). A fuzzy grasp algorithm for solving a tourist trip design problem. In *2017 IEEE Int Conf on Fuzzy Systems*.

Cunha, W., Mangaravite, V., Gomes, C., Canuto, S., Resende, E., et al. (2021). On the cost-effectiveness of neural and non-neural approaches and representations for text classification: A comprehensive comparative study. *IP&M*, 58(3):102481.

Cunha, W., Moreo, A., Esuli, A., Sebastiani, F., Rocha, L., and Gonçalves, M. A. (2025). A noise-oriented and redundancy-aware instance selection framework. *ACM TOIS*.

Expósito, A., Mancini, S., Brito, J., and Moreno, J. A. (2019). A fuzzy grasp for the tourist trip design with clustered pois. *Expert Systems with Applications*, 127:210–227.

Feo, T. A. and Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6:109–133.

Friggstad, Z., Gollapudi, S., Kollias, K., Sarlos, T., Swamy, C., and Tomkins, A. (2018). Orienteering algorithms for generating travel itineraries. In *CIKM*.

Gavalas, D., Kasapakis, V., Konstantopoulos, C., Pantziou, G., and Vathis, o. (2017). Scenic route planning for tourists. *Personal and Ubiquitous Computing*.

Gavalas, D., Konstantopoulos, C., Mastakas, K., and Pantziou, G. (2014). A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics*.

Ghobadi, F., Divsalar, A., Jandaghi, H., and Nozari, R. B. (2023). An integrated recommender system for multi-day tourist itinerary. *Applied Soft Computing*, 149:110942.

Headspin (2023). Why measuring and optimizing response time is critical for applications success. `https://shorturl.at/aTX56`. Accessed: 2023-10-17.

Kotiloglu, S., Lappas, T., Pelechrinis, K., and Repoussis, P. (2017). Personalized multi-period tour recommendations. *Tourism Management*, 62:76–88.

Pang, B., Lee, L., et al. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in information retrieval*, 2(1–2):1–135.

Ruiz-Meza, J., Brito, J., and Montoya-Torres, J. R. (2021). A grasp to solve the multi-constraints multi-modal team orienteering problem with time windows for groups with heterogeneous preferences. *Computers & Industrial Engineering*, 162:107776.

Ruiz-Meza, J., Brito, J., and Montoya-Torres, J. R. (2022). A grasp-vnd algorithm to solve the multi-objective fuzzy and sustainable tourist trip design problem for groups. *Applied Soft Computing*, 131:109716.

Ruiz-Meza, J. and Montoya-Torres, J. R. (2022). A systematic literature review for the tourist trip design problem: Extensions, solution techniques and future research lines. *Operations Research Perspectives*, 9:100228.

Sarkar, J. L. and Majumder, A. (2021). A new point-of-interest approach based on multi-itinerary recommendation engine. *Expert Systems with Applications*, 181:115026.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*. John Wiley & Sons.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., and Van Oudheusden, D. (2011). The city trip planner: an expert system for tourists. *Expert Systems with Applications*.