CiRi-Engine: POI Recommender System for Diverse and Balanced Walking Tours

Joanna Zamiechowska[1], Julia Neidhardt[1] and Wolfgang Wörndl[2]

[1]Technische Universität Wien (TU Wien), CDL RecSys, Karlsplatz 13, 1040, Vienna, Austria
[2]Technical University of Munich (TUM), Arcisstraße 21, 80333, Munich, Germany

Abstract

We present CiRi-Engine (CityRiddler Recommendation Engine), an interactive city-walking-tour recommender system. This demonstration paper showcases a novel approach to generating personalized and balanced itineraries for urban exploration. By combining user-specified constraints, such as start and end locations, tour duration, interest categories, and challenge preferences, with an efficient dual-stage routing algorithm, CiRi-Engine dynamically constructs diverse routes featuring curated Points of Interest (POIs). The engine leverages a novel hybrid of A* and Beam Search for path planning, and incorporates preference-aware POI selection to ensure both relevance and diversity. We demonstrate firsthand how the system balances route diversity, thematic coherence, and user-specified constraints, demonstrating its effectiveness for handling multiple objectives and generating engaging walking tours.

Keywords

tourist trip design problem, point-of-interest recommendation, hybrid search algorithms, heuristic algorithm, multi-objective optimization

1. Introduction

Using recommender systems for the creation of tourist itineraries is an active and long-studied problem in the tourism domain [1]. These systems provide personalized suggestions for destinations, activities, or Points of Interest (POIs), aiming to enhance the user's travel experience, while alleviating overwhelming choice among the many options that travelers face in new, unfamiliar environments. This is known as the Tourist Trip Design Problem (TTDP), which aims to generate a travel itinerary comprised of POIs in a way that takes into account the user's preferences, as well as other contexts and constraints [2].

One relevant application is the generation of walking tours within an urban environment that visits locations of interest to the user. In such urban environments, there are typically numerous POIs that can satisfy one or more aspects of a user's interests; therefore, we aim to maximize the coverage of their diverse interests. Additionally, in order to retain user engagement and improve the user experience along the tour, the route should consider balanced travel distances, avoiding large gaps between POIs, as well as avoiding many POIs clustered together. The motivation for this research is to investigate POI selection and route finding algorithms that optimize for the following requirements:

- The route should maintain balanced walking distances between POIs.
- The route should optimize for POI diversity.
- The entire walking tour must fit within a specified time constraint.

The core problem addressed, known as the Tourist Trip Design Problem (TTDP), is formulated as sequencing POIs that fit the user's stated interests and time constraints. The CiRi-Engine approach is a multi-objective optimization system that selects POIs to maximize category coverage according to user

Workshop on Recommenders in Tourism (RecTour 2025), September 22, 2025, co-located with the 19th ACM Conference on Recommender Systems, Prague, Czech Republic.

joanna.zamiechowska@tuwien.ac.at (J. Zamiechowska); julia.neidhardt@tuwien.ac.at (J. Neidhardt); woerndl@in.tum.de (W. Wörndl)

0000-0002-5101-8097 (J. Zamiechowska); 0000-0001-7184-1841 (J. Neidhardt); 0000-0003-2972-5817 (W. Wörndl)

interests, align the total tour time (the sum of walking and POI stop times) with the user's preferred duration, and ensure balanced segment lengths between consecutive stops.

The system serves as the recommendation engine for CityRiddler, a mobile app that creates interactive, gamified walking tours[1]. Users begin by specifying their preferences, and the app generates a personalized tour where each point of interest (POI) includes a riddle to solve. This design encourages engagement with the surroundings while offering an educational and entertaining experience. The POI dataset incorporates game-specific features, such as riddle difficulty, to further tailor the tour.

Unlike traditional recommender systems, CityRiddler does not rely on personal user history or external data. Instead, it constructs routes solely from elicited preferences, eliminating the need for privacy-invasive tracking. This constraint rules out conventional approaches like user-user or tour-tour similarity metrics. The engine (CiRi-Engine) addresses this challenge by combining heuristic methods with practical constraints (e.g., time, distance) to balance computational efficiency with route quality, moving beyond popularity-based or shortest-path prioritization.

No current solution concurrently achieves real-time mobile performance, minimization of collected user data, route segment balance, and explicit multi-objective optimization. To address current challenges and research gaps, we present an interactive tour recommender system that combines personalized itinerary planning, constraint management, and responsive performance on mobile devices. Our system, the CiRi-Engine, uses a fast, two-stage search method, merging A* and Beam Search, to create diverse and balanced city walking tours.

2. Related Work

Recent work on personalized tourism recommendation intersect two lines of research: Point-of-Interest (POI) recommender systems and the Tourist Trip Design Problem (TTDP). Research gaps remain around real-time, privacy-preserving, multi-objective itinerary construction for mobile devices.

The TTDP extends the Orienteering Problem by adding additional constraints such as time windows, budgets, category quotas, and other logical rules. Exact algorithms, such as Vu et al.'s branch-and-check approach[3], demonstrate optimality but suffer from exponential runtimes that are not suited for real-time use on smartphones. Heuristic families such as adaptive large-neighbourhood search, memetic strategies, and recent branch-and-check hybrids have improved scalability but still require minutes to hours of server-side processing[4]. Parallel work on time-dependent or multi-day TTDP variants deepen the constraint set but further inflates computation time.

Several studies formulate itinerary planning as a multi-objective optimization problem, balancing metrics such as reward, cost, novelty, and diversity. Techniques range from super-MOEAs[5] and weak-correlation evolutionary search to archival ant-colony systems[6]. Although these methods can find high-quality trade-offs offline, they typically require hundreds of generations and substantial GPU/CPU resources, limiting their viability for real-time deployment. Complementary research leverages clustering plus genetic operators to stitch multi-day routes, but again produces runtimes unsuited to instant mobile feedback.

Hybrid approaches combine search strategies to balance efficiency and quality. Herzog's user-centered approach to the TTDP demonstrates the effectiveness of combining classical shortest-path algorithms with context-aware heuristics, using an extended Dijkstra's algorithm that incorporates dynamic profit adjustments based on user preferences, contextual factors, and route attractiveness attributes. His work also integrates GRASP (Greedy Randomised Adaptive Search Procedure) with context-aware recommendations, balancing greedy selection with randomization to enhance solution diversity[7][8].

In the tourism domain specifically, hybrid approaches often combine multiple recommendation techniques with routing optimization. Tenemaza et al. propose combining k-means clustering with genetic algorithms for multi-day itinerary optimization[9], while other researchers integrate collaborative filtering with metaheuristic algorithms like tabu search for personalized route recommendations[10]. These hybrid methodologies address the inherent complexity of the TTDP by blending the strengths

[1] https://www.cityriddler.com/

of different algorithmic paradigms: exact methods provide optimality guarantees for smaller subproblems, heuristic approaches offer computational efficiency for larger instances, and machine learning components enable personalization and context-awareness.

3. Solution Design Description

We use a hybrid of Beam Search and A* in the selection of POIs. The algorithm begins at the start location and evaluates all possible next POIs using a hybrid scoring approach, keeping only the most promising partial routes for further expansion. This process continues iteratively, with A* providing the feasibility check at each step, until the algorithm finds the best complete tour that fits within the user's time and preference constraints.

Beam Search maintains a fixed number (beam width) of the most promising partial tour solutions at each iteration, and prunes less favorable routes to keep the search space manageable. The A* algorithm is used within the beam search's candidate evaluation process.

3.1. Algorithm Description

POI data consists of geolocation, as well as attributes that the user can select for: thematic categories (e.g., history, art, cuisine, local culture), riddle difficulty (easy, medium, hard), prominence (main attraction, standard, hidden gems). The set of POIs is filtered to an initial candidate set of POIs based on the user's preferences, and a score attribute is attached to each POI. We compare the user's requested categories to the distribution of categories available from the candidate set, and apply inverse frequency weighting (rarer category = higher weight). This ensures that if the user chooses a category that has very few candidates, those POIs will be prioritized in order to maximize the coverage.

The closest POI is chosen as the starting point for the tour. Using the current location and end location, the system's fixed walking speed and minimum POI visit time, the algorithm estimates the maximum number of POIs that will fit within the target duration. This is the maximum number of iterations (POI selections) allowed.

At each iteration, when the beam search needs to determine which POIs to consider next, it calls a candidate generation function that implements the classical A* cost evaluation:

$$\begin{aligned} \text{g-cost:} &\quad g(n) = \text{accumulated duration (actual)} \\ \text{h-cost:} &\quad h(n) = \text{heuristic to goal (estimated)} \\ \text{f-cost:} &\quad f(n) = g(n) + h(n) \end{aligned}$$

The A* f-cost calculation is embedded directly into the multi-objective scoring function that combines user preferences with spatial efficiency. These are expressed as *preference gain* and *proximity bonus*.

Preference gain quantifies how well a candidate Point of Interest (POI) matches the user's selected preferences (categories, difficulty, and prominence). The idea is to reward the inclusion of POIs that help diversify or increase coverage of the user's stated interests. It is normalized between 0 and 1, where 1 indicates maximum alignment with all user preferences and 0 means the POI does not improve preference coverage, and can be expressed as:

$$\text{Pref} = \frac{\text{Preferences covered after addition}}{\text{Preferences user-selected}} \tag{1}$$

Proximity bonus is a scoring component that encourages the route to favor POIs that are spatially close to the current location. This helps build routes that are more compact and efficient, avoiding unnecessary detours. It is a normalized value between 0 and 1, and can be expressed as:

$$\text{Prox} = 1 - \frac{\text{distance to candidate (m)}}{1000} \tag{2}$$

This means the bonus decreases linearly with distance, reaching zero for candidate POIs that are 1,000 meters or farther from the current location.

Each candidate POI finally receives a composite score that weighs preference gain (40%) and proximity bonus (20%), minus the A* f-cost (60%).

$$\text{score:} \quad \text{Score} = 0.4 \cdot \text{Pref} + 0.2 \cdot \text{Prox} - 0.6 \cdot \frac{f(n)}{100}$$

Each candidate is attached to the existing partial tour, creating new tour possible tour variations. The composite score is calculated for each partial-tour and ranked. The beam search has a width of 5, so it prunes all but the top 5 possible partial-tours. This is now the current beam state.

The next iteration will repeat these steps for each of the existing 5 partial-tours in the current beam state. New candidates are found in the proximity of each partial-tour's last visited POI, and the process repeats. Each iteration carries forward only the top 5 partial tours. For each new iteration the algorithm checks if adding another POI would exceed the time budget, or if maximum number of POIs is reached. If terminating, it finalizes the tour by adding a path to the end location.

After the algorithm completes and returns a feasible tour, two post-processing steps refine the final itinerary. The algorithm applies a 2-opt local search routine that examines all possible edge swaps in the tour. For each potential swap, it removes two non-adjacent edges and reconnects the path segments, accepting improvements that reduce total walking distance and eliminate backtracking. During the final output phase, the system calculates walking segment statistics including mean distance, standard deviation, and variance to assess route quality.

3.2. Heuristics and Decision Process

An outline of the heuristic components and decision logic implemented in the system is provided below.

1. **Prevention of Backtracking and Repeats**

   No Repeats: The route planner maintains a record of already selected POIs within the current solution path. When evaluating candidate POIs for the next segment, any POI present in the current route is excluded from subsequent selection passes. This prevents the same location from appearing more than once in a single generated tour.

   Backtracking Avoidance: Transition rules ensure that the search expands only to POIs not already visited and that move away from the starting point. POI transition edges are constructed such that cycles and loops are not permitted, adhering to a path-constraint principle common in TTDP formulations.

2. **Walking Segment Distance and Time Calculation**

   Route Partitioning: The total available time (input by user) is allocated as the sum of estimated walking time and predefined POI stop times (based on riddle difficulty at each POI).

   Balanced Segment Goal: The algorithm partitions the walking segments by dividing the total walking time evenly among the anticipated number of stops. Each segment is then targeted to have a similar travel time and distance, supporting a comfortable pacing and route coherence.

   Distance/Time Computation: Segment distances are available from a pre-existing list on the server containing the walking distances between every pair of POIs in the city. After candidate POIs are chosen, the variance of the resulting segment distances is calculated. This value is considered as a penalty in scoring. Tours with lower segment variance are preferred.

3. **Category and Feature Coverage Heuristics**

   Preference Matching: Each POI is scored based on user preferences, such as desired categories (e.g., "Art", "History", "Cuisine") and riddle difficulty. The initial pre-filter narrows the candidate pool to POIs matching at least one user-stated interest.

Non-Repetition of Category: For each new candidate POI, the algorithm checks its category (e.g., "History"). If the previous POI in the route has the same category the candidate's score is decreased. This mechanism promotes diversity along the route, reducing consecutive repeated types of POIs.

Running Coverage Calculation: At every step, the algorithm maintains a record of all categories included so far. When candidates are evaluated, those representing under-covered or not-yet-seen categories are boosted in score.

Coverage as Objective: The algorithm uses a root mean square diversity error (RMSDE) metric to measure and aim for maximum possible coverage.

4. **Tie-Breaking and Segment Variance**

   Tie on POI Scores: When two candidates have similar top scores:

   Preference is given to the POI whose type or category has not yet appeared in the route.

   If this still results in a tie, the algorithm selects the POI that, when incorporated, minimizes the variance in walking segment distances.

   As a final tie-breaker, random selection provides nondeterministic diversity.

5. **Real-Time Adjustment and Cascading Effects**

   Dynamic Re-Scoring: After each POI is added, the scores of remaining candidates are dynamically updated, reflecting the current state of category coverage and segment evenness. This ensures that the path is adaptive and responsive to prior selections.

   Constraint Enforcement: At each state expansion, constraints are checked: total projected time must not exceed user-defined limits, all POIs must be within the search area buffer, and categories must diversify sufficiently within available options.

4. Implementation

The CiRi-Engine technical architecture is designed for responsive and privacy-preserving mobile tour generation. All core systems are deployed on AWS cloud infrastructure. The algorithm itself is implemented in Python and served via a FastAPI backend. User preferences are solicited directly from the user's mobile device, and upon receiving a request, the backend algorithm processes the input and constructs a tour in near real time. The generated itinerary is returned to the user's smartphone. The only data transmitted or stored is GPS location and time, ensuring strong privacy standards.

4.1. System Overview

The raw data consists of POI locations in Vienna, as well as the riddle category, prominence, and difficulty associated with them. A POI can have multiple riddles assigned to it, therefore also multiple categories and so forth. The walking distances between each POI is calculated using GraphHopper API[2] , and stored in a file for later lookup.

The user provides their preferred start location, categories, difficulty, prominence, and tour duration on the mobile app front-end. The system has hard-coded values for walking speed, and the expected time it takes to solve a riddle of each difficulty level. These times are considered the POI's visit time, and are added to the walking times to comprise the entire tour duration.

[2]https://www.graphhopper.com/

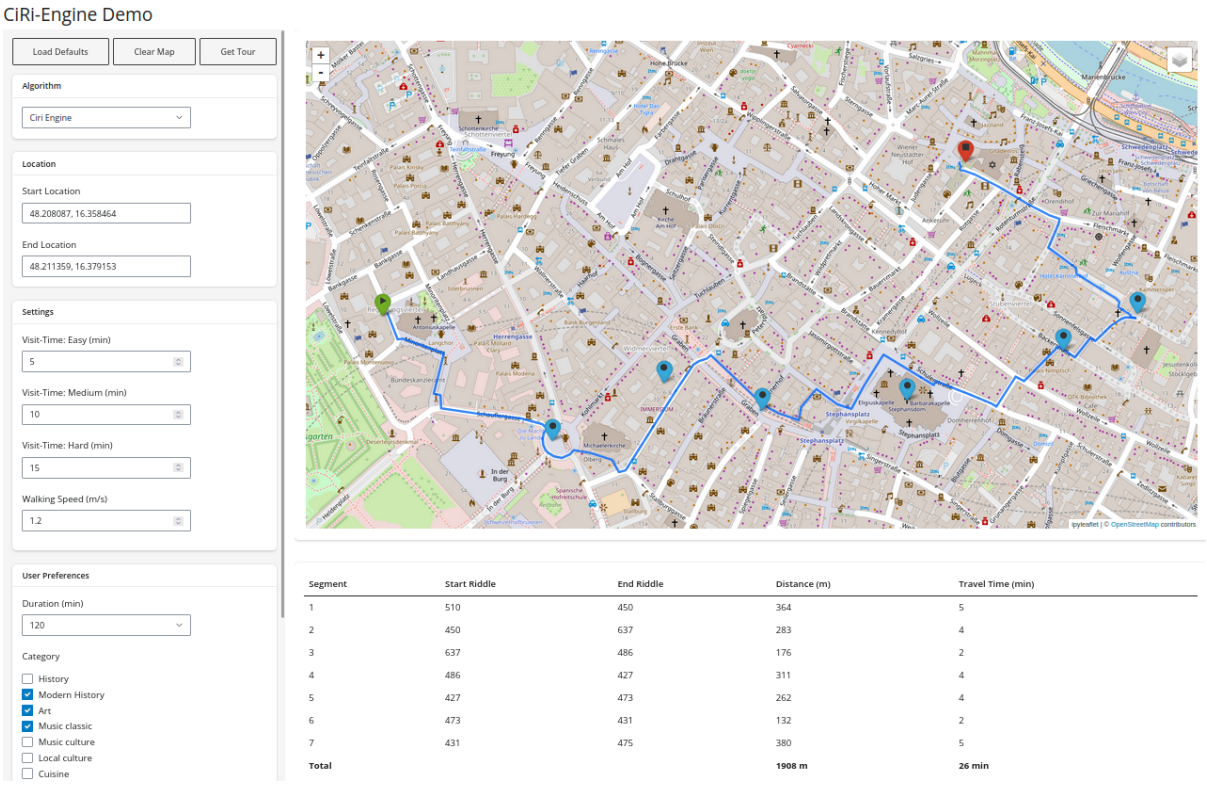

Figure 1: The interactive demo site with a tour.

4.2. Proof of Concept

This system will be demonstrated live, and participants are invited to interact with the demo page. Users will be able to input their own preferences and generate custom walking tours in real time, exploring the system's features and seeing first-hand how routes, statistics, and map visualizations adapt to their selections. This interactive demonstration offers direct experience of CiRi-Engine's capabilities and encourages feedback on its performance and usability.

The system is designed primarily for use as a mobile application; however, to support demonstration and testing, we have also developed a web-based platform, illustrated in Figure 1. This demo site features an interactive interface where users can configure tour preferences and modify key system settings. Generated routes and walking paths are displayed on an interactive map, accompanied by a statistical summary of the tour. The demonstration platform is implemented as a Shiny[3] web application and communicates with a FastAPI[4] backend via API endpoints.

A sample tour was generated with the following user preferences: start (48.208087, 16.358464) and end (48.211359, 16.379153); target tour duration 90 minutes; categories 2, 3, and 4; difficulty levels 0 and 1; and prominence levels 1 and 2. The visit time and walking speed settings remain the same as the existing system's default, but could be modified for testing. The tabulated results of the tour are shown in Table 1.

5. Conclusion

By combining Beam Search and A* search strategies, the CiRi-Engine dynamically tracks user preferences and spatial efficiency during route planning. This enables it to craft tour itineraries that align closely with user preferences, such as desired difficulty, categories, and time constraints. It also optimizes for practical considerations that enhance the user experience like walking distance and overall tour

[3]https://shiny.posit.co/py/
[4]https://fastapi.tiangolo.com/

Table 1
Enhanced Segment Details

Segment	Start POI	End POI	Distance (m)	Walking Time (min)	End POI Visit Time (min)	Total Segment Time (min)
1	Start	492	340	4.0	5	9.0
2	492	491	477	7.1	5	12.1
3	491	501	521	6.2	10	16.2
4	501	717	333	4.2	10	14.2
5	717	522	392	5.8	10	15.8
6	522	531	8	0.1	5	5.1
7	531	533	271	3.2	5	8.2
8	533	End	561	6.7	0	6.7

=== COMPREHENSIVE TOUR ANALYSIS ===
Tour POIs: [492, 491, 501, 717, 522, 531, 533]
Number of route segments: 8
Total walking distance: 2.91 km
Total walking time: 37.3 minutes
Total visit time: 50.0 minutes
Total tour time: 87.3 minutes
Distance coefficient of variation: 48.2

compactness. The engine's data-driven approach ensures each proposed tour is both thematically relevant and logistically efficient.

At the time of this paper's writing, the CiRi-Engine is being integrated to the City Riddler's live system for A/B testing to conduct a user study. The evaluation will focus on how well the generated tours match user preferences, as well as how quickly and efficiently routes are computed compared to the previous solution. Early feedback confirms that the CiRi-Engine generates tours significantly faster than the current implementation. This optimization has the potential to enhance user satisfaction and to support scalability as the app is rolled out in other major cities in the future. The ongoing user study will provide further insights into the practical impact of these advancements and inform future refinements to both the algorithm and the overall tour planning experience.

6. Future Work

The current CiRi-Engine prototype shows promising performance, but several avenues can strengthen its robustness, scalability, and user appeal.

The current algorithm allows for a system to expand the features of a POI and incorporate them into the scoring system. The data could be refined with such attributes as sustainability score, indoor-outdoor flag, or crowding level. Opening times for parks or businesses could also be included. The route itself could be selected based on its features, such as aesthetic or cultural qualities. Pedestrian or low-traffic areas could be favored, and accessibility or ease of walking (avoiding steep gradients or stairs).

The routing algorithm could be enhanced to allow for generating the tour dynamically after each POI visit, recalibrating remaining time and accommodating user feedback or user deviations from the expected visit-time. Different strategies for supporting the case where the start and end locations are the same should be compared and implemented.

In addition, it would be interesting to evaluate the system against state-of-the-art large language model (LLM) techniques, assessing whether such models can handle the multi-objective optimization and adapt to dynamic recalibration constraints.

Future work also includes implementing and evaluating against state of the art Branch-and-Bound methods, which guarantee global optimality but face severe scalability challenges. The algorithm will also be tested using a dataset from different cities to test scalability and performance with very dense or sparse data.

Acknowledgments

We would like to express our sincere gratitude to CityRiddler for their collaboration, especially Lukas Baronyai, Co-Founder & CTO, for his support. This work was supported by the Austrian Research Promotion Agency (FFG). Furthermore, the authors gratefully acknowledge the financial support provided by the Austrian Federal Ministry of Labour and Economy, the National Foundation for Research, Technology and Development, and the Christian Doppler Research Association.

Declaration on Generative AI

During the preparation of this work, the authors used Perplexity AI[5] and Overleaf's TeXGPT[6] in order to: Grammar and spelling check, Citation management, and Formatting assistance. After using this tool, the authors reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] J. Sarkar, A. Majumder, C. Panigrahi, S. Roy, B. Pati, Tourism recommendation system: a survey and future research directions, Multimedia Tools and Applications 82 (2022) 1–45. doi:10.1007/s11042-022-12167-w.

[2] J. Ruiz-Meza, J. R. Montoya-Torres, A systematic literature review for the tourist trip design problem: Extensions, solution techniques and future research lines, Operations Research Perspectives 9 (2022) 100228. URL: https://www.sciencedirect.com/science/article/pii/S2214716022000069. doi:https://doi.org/10.1016/j.orp.2022.100228.

[3] D. M. Vu, Y. Kergosien, J. E. Mendoza, P. Desport, Branch-and-check approaches for the tourist trip design problem with rich constraints, Computers & Operations Research (2022) 105566. doi:10.1016/j.cor.2021.105566.

[4] A. Santini, An Adaptive Large-Neighbourhood Search Algorithm the Orienteering Problem, Technical Report 29 October, Universitat Pompeu Fabra, Barcelona, Spain, 2018. URL: https://doi.org/10.1016/j.eswa.2018.12.050. doi:10.1016/j.eswa.2018.12.050, reports run-times ranging from 5 minutes to 5 hours for larger instances.

[5] X.-R. Zhang, X.-Y. Wang, T. Ebara, Personalised multi-objective travel route recommendation based on a super-MOEA, Journal of Network Intelligence 9 (2024) 535–554.

[6] J. Zhang, Y. Li, Application of improved multi-objective evolution algorithm in intelligent tourism interest point recommendation and itinerary planning, Informatica 49 (2025) 1–16. URL: https://doi.org/10.31449/inf.v49i7.6529. doi:10.31449/inf.v49i7.6529.

[7] D. Herzog, A User-Centered Approach to Solving the Tourist Trip Design Problem for Individuals and Groups, Ph.D. thesis, Technical University of Munich, Germany, 2020. URL: https://mediatum.ub.tum.de/node?id=1540474.

[8] D. Herzog, C. Laß, W. Wörndl, Tourrec — a tourist trip recommender system for individuals and groups, in: Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18), ACM, Vancouver, Canada, 2018, pp. 507–510. doi:10.1145/3240323.3241612.

[9] M. Tenemaza, S. Luján-Mora, A. de Antonio, J. Ramírez, Improving itinerary recommendations for tourists through metaheuristic algorithms: An optimization proposal, IEEE Access 8 (2020) 82820–82837. doi:10.1109/ACCESS.2020.2990348.

[10] B. Han, X. Zheng, M. Guan, L. Sun, Z. Yue, Personalized route recommendation with hybrid tabu search algorithm based on crowdsensing, International Journal of Intelligent Systems 2023 (2023) 1–15. URL: https://doi.org/10.1155/2023/3054888. doi:10.1155/2023/3054888.

[5] https://www.perplexity.ai

[6] https://www.overleaf.com/about/ai-features