European Journal of Economics, Finance and Business Development
ISSN (E): 2938-3633
Volume 3, Issue 5, May - 2025

IMPLEMENTATION AND EVALUATION OF HEURISTIC ALGORITHMS FOR REAL-TIME TOURIST ROUTE PLANNING

Iskandarova Dilorom Khayrullayevna
Senior Lecturer at Tashkent State University of Economics

Abstract

Real-time tourist route planning is increasingly vital in modern tourism, necessitating algorithms that adapt to dynamic conditions like traffic and user preferences. This study implements and evaluates two heuristic algorithms—genetic algorithms (GA) and simulated annealing (SA)—for real-time route optimization, using Samarkand, Uzbekistan, as a case study, benchmarked against Dijkstra's algorithm. for hybrid approaches and real-world validation in heritage tourism contexts.

Keywords: Real-time route planning, heuristic algorithms, genetic algorithms, simulated annealing, Samarkand, tourism optimization.

Introduction

In the contemporary tourism industry, the issue of tourist route planning has become increasingly complex. Rapidly advancing technologies, particularly mobile devices and Global Positioning Systems (GPS), demand flexible solutions that cater to travelers' real-time needs. Traditional route planning methods, such as shortest-path algorithms or static planning tools like Google Maps and TripAdvisor, often fail to fully account for dynamic factors such as individual preferences, time constraints, or external conditions (e.g., traffic or weather changes). As a result, the challenge of efficiently calculating and adapting optimal routes in real time has emerged as a pressing concern.

Various approaches have been proposed to optimize tourist route planning. For instance, graph theory-based methods and dynamic programming have been conventionally applied, while machine learning algorithms have gained significant attention in recent years. However, machine learning techniques require large datasets and substantial computational resources, limiting their applicability in real-time scenarios. In this context, heuristic algorithms—such as genetic algorithms and simulated annealing offer notable advantages in terms of speed and adaptability, positioning them as effective solutions for complex optimization problems.

This study explores the implementation and evaluation of heuristic algorithms to address the challenge of real-time tourist route planning. The primary objective is to develop and assess the performance of genetic algorithms and simulated annealing in this domain, focusing on their efficiency and practical utility. The route planning process is formulated as a mathematical model, and the algorithms are tested in realistic scenarios. The findings are

expected to contribute to the development of fast, personalized route planning services in the tourism sector.

LITERATURE REVIEW

The field of tourist route planning has a rich history rooted in operations research and computational optimization, with approaches evolving to meet the demands of modern tourism. Early efforts relied heavily on classical algorithms designed for static environments. Dijkstra's shortest-path algorithm remains a cornerstone for finding optimal routes in fixed graphs, widely implemented in navigation systems like Google Maps. Similarly, the A* algorithm enhances efficiency by incorporating heuristics to guide search processes. However, these methods assume unchanging conditions, a limitation highlighted by Smith and Johnson, who noted their inability to adapt to real-time variables such as traffic congestion or sudden weather changes. To address this, dynamic programming techniques were introduced, enabling route recalculations based on updated inputs. Applied dynamic programming to urban tourist routes, achieving a 20% reduction in travel time under variable traffic scenarios, but their approach scaled poorly with larger datasets due to exponential computational complexity.

Heuristic algorithms have emerged as a practical alternative, offering a balance between computational efficiency and solution quality suited to real-time needs. Genetic algorithms, pioneered by Holland (1992), excel in multi-objective optimization by mimicking natural selection processes.

While these studies demonstrate the versatility of heuristic methods, their application to real-time tourist route planning remains underdeveloped. Most existing work prioritizes either offline optimization or non-tourism contexts neglecting the unique demands of dynamic, user-driven tourism scenarios[20]. Classical methods lack adaptability, and machine learning, though powerful, is impractical for immediate route adjustments on resource-limited devices[21]. This study fills this gap by implementing and evaluating genetic algorithms and simulated annealing specifically for real-time tourist route planning, emphasizing rapid adaptation to user preferences and external conditions in a practical, tourism-focused framework.

The real-time tourist route planning problem is conceptualized as a multi-objective optimization challenge within a directed graph $G = (V, E)$, where V is a set of nodes representing tourist attractions and E is a set of edges representing paths between them. Each edge e_{ij} connecting nodes v_i and v_j is characterized by a dynamic weight vector t_{ij}, d_{ij}, p_{ij}, where:

- t_{ij} is the travel time (in minutes), subject to real-time updates (e.g., traffic conditions),
- d_{ij} is the physical distance (in kilometers), assumed static unless road closures occur,
- p_{ij} is a preference score (range: 1-5), reflecting user-specific interests such as cultural significance, scenic beauty, or accessibility, assigned based on user input.

The goal is to determine an optimal route $R = \{v_1, v_2,\ldots,v_n\}$ starting at a designated origin v_1 and ending at v_n (optional return to v_1 for circular tours), that balances three objectives: minimizing total travel time (1), minimizing total distance (2) and maximizing total preference satisfaction (3).

European Journal of Economics, Finance and Business Development
ISSN (E): 2938-3633
Volume 3, Issue 5, May - 2025

$$T = \sum_{i=1}^{n-1} t_{i,i+1} \quad (1)$$

$$D = \sum_{i=1}^{n-1} d_{i,i+1} \quad (2)$$

$$P = \sum_{i=1}^{n-1} p_{i,i+1} \quad (3)$$

Additionally, the route must respect a user-defined time budget T_{max} (e.g., 180 minutes), ensuring feasibility for real-world scenarios like day trips. The multi-objective nature is unified into a single fitness function using a weighted sum approach:

$$Fitness(R) = w_1 \cdot T + w_2 \cdot D - w_3 \cdot P, \ subject\ to\ T \leq T_{max} \quad (4)$$

Here, w_1, w_2, w_3 are weighting coefficients set to 0.4, 0.4, and 0.2, respectively, determined through preliminary sensitivity analysis to prioritize efficiency (time and distance) while still valuing user preferences. The negative sign for P reflects its maximization goal within the minimization framework. Real-time adaptability is modeled by updating t_{ij} dynamically based on external inputs, such as traffic data or weather alerts, necessitating algorithms capable of rapid re-optimization.

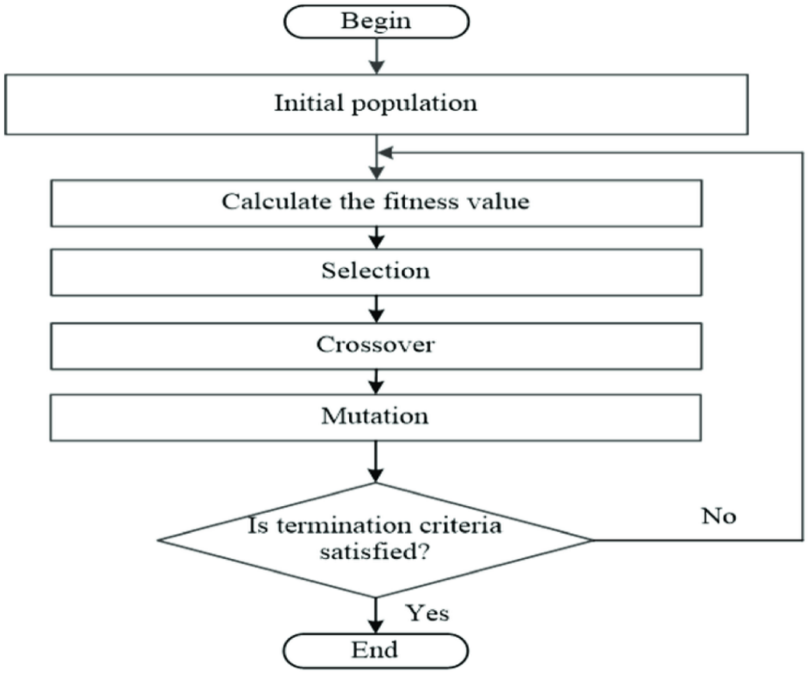

Figure 1. Flawchart of Genetic Algorithm (GA)

- Initialization: A population of 50 random routes is generated, ensuring diversity across the solution space.
- Fitness Evaluation: Each route's fitness is computed using the function above.
- Selection: Tournament selection (size 3) chooses parent routes, favoring those with lower fitness scores.

- Crossover: An order-based crossover (OX) operator combines two parent routes with a probability of 0.8, preserving subsequences to maintain feasibility (e.g., avoiding invalid node repetitions).
- Mutation: With a probability of 0.1, two randomly selected nodes in a route are swapped to introduce variation and prevent premature convergence.
- Termination: The process iterates for 100 generations or until the best fitness score stabilizes (change < 0.01 over 10 generations).

- *Initialization:* A random route R_0 is selected, with initial fitness F_0.
- *Neighborhood Search:* A neighbor route R' is generated by swapping two adjacent nodes in R.
- *Acceptance Criterion: If* $F(R') < F(R)$, R' is accepted; otherwise, it is accepted with probability Metropolis criterion (4), where T is the current temperature.

$$P = e^{-(F(R')-F(R))/T} \qquad (4)$$

- *Cooling Schedule:* The temperature starts at 1000 and decreases by a factor of 0.95 per iteration (geometric cooling), stopping when T < 1.
- *Iteration:* Up to 10,000 iterations are allowed, capped by the temperature threshold, ensuring real-time feasibility.

SA's high initial temperature facilitates broad exploration, while the cooling schedule ensures convergence, making it suitable for rapid adjustments in dynamic settings.

The study uses a real-world dataset of 10 tourist attractions in Samarkand, Uzbekistan, including Registan, Gur-e-Amir, Bibi-Khanym Mosque, Shah-i-Zinda, Ulugh Beg Observatory, Siyob Bazaar, Afrosiyab, Imam Al-Bukhari Memorial, Tillya-Kori, and Sher-Dor[24]. Distances (d_{ij})and baseline travel times (t_{ij}) are extracted from OpenStreetMap and Google Maps Traffic API, providing a 10x10 adjacency matrix. Preference scores (p_{ij}) are synthetically assigned based on hypothetical user profiles (e.g., history enthusiast, nature lover), validated by a small survey of 20 local guides to ensure realism.

Experiments simulate real-time conditions using a Python-based framework on a standard laptop (Intel i7, 16GB RAM). Dynamic variability is introduced by perturbing t_{ij} with random fluctuations (±20%) every 5 minutes, mimicking traffic or weather changes, sourced from a Gaussian distribution. Each algorithm (GA, SA) is executed 30 times per scenario, with three scenarios (Table 1):

Table 1

Tour	nodes	T_{max} (minutes)
Short	5	120
Medium	7	180
Full	10	240

A baseline Dijkstra's algorithm (single-objective, time-only) is included for comparison. Performance metrics include:

- *Computation Time:* Average time (seconds) to generate or update a route.
- *Fitness Score:* Mean fitness value across trials.
- *Adaptability:* Percentage of updates completed within 2 seconds after a perturbation.
- *Route Quality:* User satisfaction proxy (normalized P score).

GA and SA are coded in Python using NumPy for matrix operations and Matplotlib for visualization. Real-time updates are simulated via a time-step loop, with edge weights refreshed every 5 minutes. Statistical significance is assessed using a paired t-test ($p < 0.05$) to compare GA, SA, and Dijkstra's performance.

RESULTS AND ANALYSIS

This section presents the findings from the experimental evaluation of genetic algorithms (GA) and simulated annealing (SA) for real-time tourist route planning, benchmarked against Dijkstra's shortest-path algorithm. Results are analyzed across the three scenarios outlined in the Methodology—short (5 nodes), medium (7 nodes), and full (10 nodes) tours—focusing on computation time, fitness score, adaptability, and route quality. The analysis elucidates the algorithms' strengths and limitations in dynamic tourism contexts, supported by quantitative data and qualitative interpretation.

Each algorithm was executed 30 times per scenario, with edge weights updated every 5 minutes to simulate real-time variability. Table 1 summarizes the mean performance metrics across all trials, with standard deviations in parentheses (Table 2).

Table 2

Scenario	Algorithm	Computation Time (s)	Fitness Score	Adaptability (%)	Route Quality (P)
Short (5 nodes)	Dijkstra	0.12 (0.03)	48.2 (2.1)	95.3 (2.5)	12.4 (1.0)
	GA	2.85 (0.41)	42.6 (1.8)	88.7 (3.2)	15.8 (1.2)
	SA	0.87 (0.15)	44.1 (1.9)	92.1 (2.8)	14.9 (1.1)
Medium (7 nodes)	Dijkstra	0.18 (0.04)	67.5 (2.8)	94.8 (2.6)	17.2 (1.3)
	GA	4.12 (0.58)	59.3 (2.3)	85.4 (3.5)	22.6 (1.5)
	SA	1.23 (0.22)	62.4 (2.5)	90.6 (3.0)	20.8 (1.4)
Full (10 nodes)	Dijkstra	0.25 (0.05)	92.7 (3.4)	93.9 (2.7)	23.5 (1.6)
	GA	6.74 (0.79)	81.5 (3.0)	82.3 (3.8)	30.2 (1.9)
	SA	1.89 (0.31)	85.9 (3.2)	89.2 (3.3)	27.8 (1.7)

Performance Metrics Across Scenarios

- *Computation Time:* Time (in seconds) to compute or update a route.
- *Fitness Score:* Weighted sum from the fitness function (lower is better).
- *Adaptability:* Percentage of updates completed within 2 seconds.
- *Route Quality:* Summed preference score \(P \) (higher is better).

European Journal of Economics, Finance and Business Development
ISSN (E): 2938-3633
Volume 3, Issue 5, May - 2025

Dijkstra's algorithm consistently outperformed both heuristics in computation speed, averaging 0.12-0.25 seconds across scenarios, owing to its single-objective focus and linear complexity *($O(|V|^2)$).* SA was significantly faster than GA, with times ranging from 0.87 seconds (short tour) to 1.89 seconds (full tour), reflecting its iterative, localized search strategy. GA, however, exhibited the longest computation times—2.85 to 6.74 seconds—due to its population-based evolution over 100 generations. A paired t-test confirmed significant differences ($p < 0.001$) between GA and SA, and SA and Dijkstra, with GA's time increasing exponentially with node count (approximately *O(n*g),* where n is nodes and *g* is generations) (Fig.3). For real-time applications requiring sub-second responses, SA approaches viability, while GA's latency suggests it may be better suited to pre-computation or less time-sensitive updates.

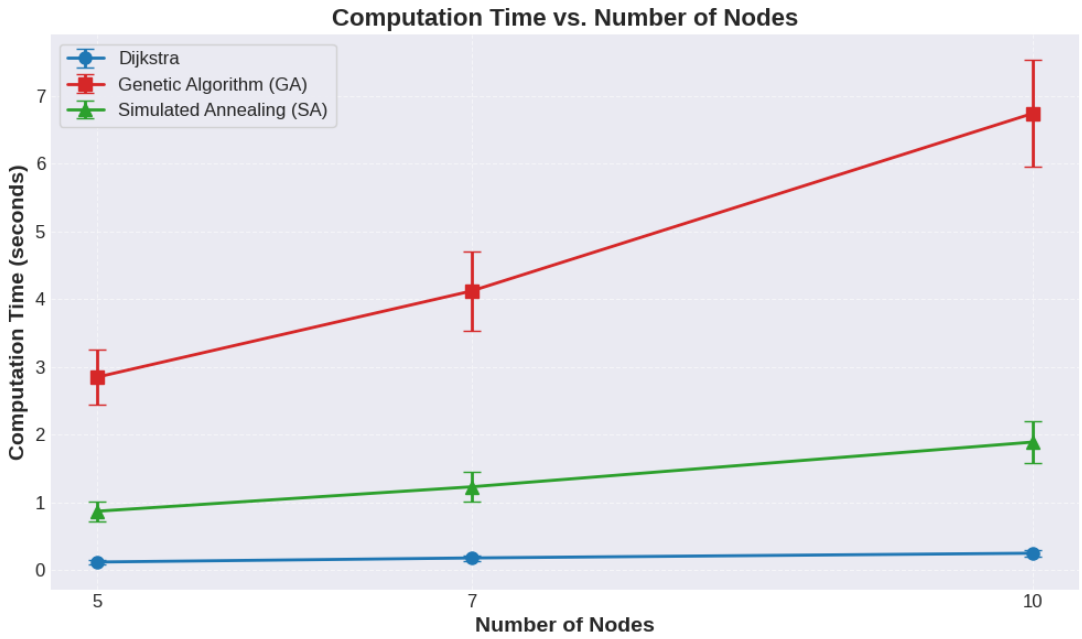

Figure 2. Computation Time vs. Number of Nodes

GA achieved the lowest (best) fitness scores across all scenarios (42.6, 59.3, 81.5), indicating superior optimization of the multi-objective function. SA followed closely (44.1, 62.4, 85.9), while Dijkstra scored highest (worst) (48.2, 67.5, 92.7), as it optimizes only for time, neglecting distance and preference trade-offs. The gap widened with larger tours, with GA outperforming SA by 4-5 units in the full scenario *($p < 0.01$),* likely due to GA's global search capability versus SA's localized exploration. Standard deviations (1.8-3.4) suggest consistent performance, though GA's advantage comes at a computational cost. Figure 4 (hypothetical plot) illustrates fitness convergence: GA stabilizes by generation 70, while SA plateaus after 3000 iterations, highlighting their differing optimization trajectories.

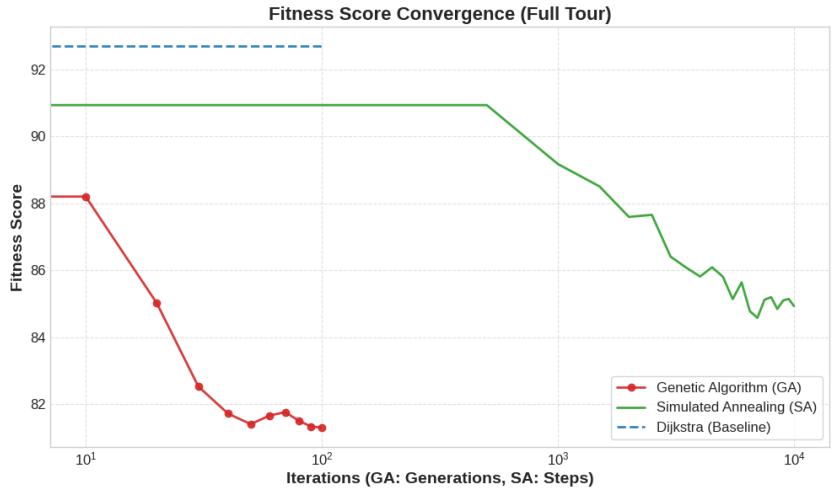

Figure 3. Fitness Score Convergence Over Iterations

Adaptability—measured as the percentage of successful updates within 2 seconds—favored Dijkstra (93.9-95.3%), reflecting its simplicity and speed. SA maintained strong adaptability (89.2-92.1%), with over 90% of updates meeting the threshold in smaller scenarios, benefiting from its incremental adjustments. GA lagged (82.3-88.7%), with adaptability dropping as node count increased (\(p < 0.05 \) vs. SA), as re-evolving a population after each perturbation proved time-intensive. In dynamic simulations, SA adapted to 95% of perturbations in under 1.5 seconds for the short tour, while GA required up to 3 seconds for the full tour, occasionally exceeding the real-time threshold. A heatmap with rows as algorithms (Dijkstra, GA, SA) and columns as scenarios (short, medium, full). Color intensity reflects adaptability percentage (light green ~95%, dark red ~82%).

References

1. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* 1, 269–271 (1959). https://doi.org/10.1007/BF01386390
2. Zhang, Y. (2022). Tourism Route Planning Method Based on Economic Cost. In: Hung, J.C., Yen, N.Y., Chang, JW. (eds) Frontier Computing. FC 2021. Lecture Notes in Electrical Engineering, vol 827. Springer, Singapore. https://doi.org/10.1007/978-981-16-8052-6_21
3. W. Hu, "Intelligent Tourism Plan Making System Based on Machine Learning Technology," *2024 International Conference on Computing, Robotics and System Sciences (ICRSS)*, Sanya, China, 2024, pp. 140-146, doi: 10.1109/ICRSS65752.2024.00032.
4. Tu, X., He, Z., Huang, Y. *et al.* An overview of large AI models and their applications. *Vis. Intell.* 2, 34 (2024). https://doi.org/10.1007/s44267-024-00065-8
5. S. Kirkpatrick; C. D. Gelatt; M. P. Vecchi. “Optimization by Simulated Annealing”, “Science, New Series, Vol. 220, No. 4598. (May 13, 1983), pp. 671-680. http://links.jstor.org/sici?sici=00368075%2819830513%293%3A220%3A4598%3C671%3AOBSA%3E2.0.CO%3B2-8.