Implementation and In-Depth Analysis of Flight Optimization Systems for Travel Agencies: A Multi-Criteria Approach

Yasin Yeşilyurt
TOBB ETÜ Artificial Intelligence Engineering
Söğütözü cad. TOBB ETÜ Konukevi
Yenimahalle/Ankara
yasinyesilyurt@hotmail.com

Abstract—**This paper presents a multi-criteria flight optimization system for travel agencies, designed to recommend optimal flight routes by balancing cost, duration, and customer satisfaction. The proposed framework employs A*, Dijkstra's, and Genetic Algorithms implemented in Python to evaluate flight data spanning 1993-2023 from U.S. domestic routes, sourced from a Kaggle dataset containing 245,955 entries with parameters such as fare, distance, carrier dominance, and airport coordinates. Key innovations include a hybrid approach combining heuristic pathfinding (using Haversine distance for A*) with multi-objective A* to address multi-objective optimization. Results demonstrate the effectiveness of A* and Dijkstra's algorithms in minimizing travel costs and time, while genetic algorithm gives sufficient results, it is much slower than mentioned algorithms without proper optimization techniques. The system currently provides personalized flight recommendations based on user preferences, with visualized outputs generated via the NetworkX library. This work contributes a scalable framework for enhancing decision-making in travel planning systems.**

I. INTRODUCTION

The growing complexity of air travel planning necessitates advanced systems to optimize flight routes while balancing competing factors such as cost, duration, and customer satisfaction. Traditional travel recommendation tools often prioritize single objectives, such as minimizing price or travel time, but fail to account for multi-dimensional preferences or dynamic market conditions. This limitation underscores the need for intelligent systems capable of synthesizing diverse parameters to deliver personalized and context-aware solutions.

Existing approaches to route optimization predominantly rely on classical graph algorithms like Dijkstra's or heuristic methods such as A*. However, these methods struggle to handle multi-criteria optimization, where trade-offs between conflicting objectives (e.g., cost vs. comfort) must be quantified. While recent studies have explored hybrid frameworks combining machine learning and optimization techniques, their reliance on simplified datasets or synthetic benchmarks limits real-world applicability.

This paper addresses these gaps by proposing a hybrid algorithmic framework that integrates A* (heuristic Search), Dijkstra's algorithm (shortest-path), and Genetic Algorithms (evolutionary optimization) to optimize flight routes across three criteria: fare, travel time, and carrier-specific customer satisfaction metrics. The system leverages a comprehensive dataset of 245,955 U.S. domestic flights (1993-2023), which includes real-world attributes such as historical fares, carrier dominance, and geographic coordinates. Key innovations include:

- A Haversine-based heuristic for A* to compute geodesic distances between airports, improving route accuracy.
- A genetic crossover mechanism to evolve flight paths while preserving high-quality route segments.
- A dynamic weighting system to prioritize user-defined preferences (e.g., cost-sensitive vs. time-sensitive travelers).
- A multi-objective A* algorithm that evaluates trade-offs between fare, travel time, and customer satisfaction, enabling personalized recommendations.
- Visualization of flight networks and optimized routes using the NetworkX library, enhancing decision-making.

Preliminary results demonstrate the viability of A* and Dijkstra's algorithms for single-objective optimization but the genetic algorithm exhibits slower performance. Among the mentioned algorithms A* performs best in terms of cost and time minimization thus it has been selected as multi-criteria optimizer. The remainder of this paper is organized as follows: Section II details the methodology and dataset, Section III discusses implementation challenges and results, and Section IV outlines future work to refine hybrid algorithm performance and integrate real-time customer feedback.

II. METHODOLOGY

The proposed flight optimization system follows a structured pipeline (Fig. 1) comprising data preprocessing, algorithmic implementation, and multi-criteria evaluation.

A. Data Collection and Preprocessing

A dataset of 305,189 U.S. domestic flights (1993–2023) [1] was sourced from Kaggle, containing attributes such as fares, carriers, passenger counts, airport coordinates, and quarterly trends. Key preprocessing steps include:

- 1. Data Cleaning: Removal of 105,189 incomplete entries, retaining 200,000 flights with non-null values.
- 2. Subset Selection: Partitioning by year and quarter to reduce computational load.
- 3. Graph Construction: Representing airports as nodes and flights as edges in a weighted graph using Python's 'NetworkX' library. Edge weights combine:

Cost	Normalized fare ($) scaled by carrier dominance (market share).
Time	Flight duration derived from Haversine distance (km) between coordinates.

Workflow diagram of the flight path finder pipeline is shown in Fig. 1.

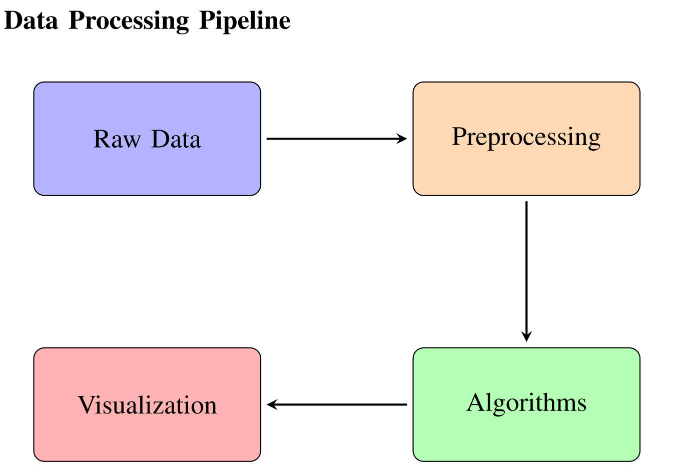


Fig. 1. Flight path finder pipeline workflow from data to visualization

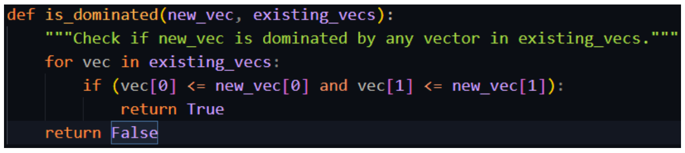
B. Algorithm Design

Three core algorithms with a multi-objective variant of these algorithms were implemented and compared:

1) A* Algorithm:
   - Heuristic Function: Geodesic distance between airports computed via the Haversine formula:

$$d = 2R \arcsin \sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_1 \cos\phi_2 \sin^2\left(\frac{\Delta\lambda}{2}\right)}$$

     where R is Earth's radius, and ϕ, λ denote latitude/longitude.
   - Cost Function: $C = \alpha \cdot \text{fare} + \beta \cdot \text{distance}$.
2) Dijkstra's Algorithm:
   - Computes globally optimal paths for a single criterion (e.g., minimal distance).
   - Extended to support dynamic reweighting of edges based on user preferences.
3) Genetic Algorithm (GA):
   - Population Initialization: 100 chromosomes generated via greedy random walks.
   - Crossover: Segments of parent paths sharing common nodes (e.g., P1: A→B→C→D and P2: A→X→C→Y yield children A→B→C→Y and A→X→C→D).
   - Mutation: Random node insertion/deletion (currently disabled due to instability in variable-length paths).
   - Fitness Function: Minimizes *cost* while penalizing excessive layovers.
4) Multi-Criteria A* Algorithm:
   - Heuristic Function: Haversine distance as in A*.
   - Cost Function: follows in Fig 2.
   - Cost Function: $C = \alpha \cdot \text{fare} + \beta \cdot \text{distance}$.



```
def is_dominated(new_vec, existing_vecs):
    """Check if new_vec is dominated by any vector in existing_vecs."""
    for vec in existing_vecs:
        if (vec[0] <= new_vec[0] and vec[1] <= new_vec[1]):
            return True
    return False
```

Fig. 2. Pareto Optimal checker for Multi-Criteria A* Algorithm

III. Implementation Details

- Environment: Python 3.9, Jupyter Notebook for interactive development.
- Libraries: 'NetworkX' for graph representation, 'pandas' for data manipulation, 'numpy' for numerical operations.
- Visualization: Matplotlib and NetworkX for geographic rendering of flight paths.

A. Challenges and Mitigations

1) Data Completeness and Geographic Accuracy:

- Challenge: Initial datasets lacked geographic coordinates for airport entries, compromising distance calculations.
- Mitigation: Found a different dataset that included most of its airport coordinates.

2) Computational Bottlenecks:

- Challenge: Rendering 200,000+ flights in `NetworkX` caused latency.
- Mitigation: Visualization was limited to 50–100 nodes when testing, complete graphs are rendered afterwards.

3) Multi-Parameter Optimization:

- Challenge: No standard function to unify cost, time, and comfort metrics.
- Mitigation: A user-configurable weighted sum (e.g., 20% cost, 80% distance) and multi-parameter A* was implemented.

4) Genetic Algorithm Implementation:

- Challenge 1: Python's list/tuple structures caused inefficient chromosome representation. That caused shape mismatch errors during crossover and mutation operations.
- Mitigation: Mutation instability was resolved by enforcing path continuity checks during chromosome editing. But a more planned chromosome representation is recommended for future work.
- Challenge 2: Random Heuristic Walk algorithm caused mutations to be mostly longer paths. This caused mutated chromosomes to be longer than the original ones.

- Mitigation: Implemented a natural selection (sorting the chromosomes by their length) is implemented.

References

[1] G. Eason et al., “On certain integrals,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551, 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism. Oxford: Clarendon, 1892.