# User Friendly Shopping List Optimization Agent

Özge Işıklar
*Artificial Intelligence Engineering*
*TOBB ETU*
Ankara, Turkey
ozgeisklar@gmail.com

***Abstract*—Efficient shopping in urban settings like Ankara requires balancing product costs and transportation expenses while ensuring all desired items are acquired from available stores. This study introduces a cost-optimization tool designed to create an optimal shopping plan for a given list of items, considering that not all products may be available at every store. Using a synthetic dataset representing Ankara's retail environment, the tool employs four optimization algorithms—Simulated Annealing (SA), Ant Colony Optimization (ACO), Genetic Algorithm (GA), and A*—to select the best stores and compute the least-cost route, factoring in product prices and travel costs. The system features a user-friendly interface that accepts dynamic shopping lists and starting coordinates, identifies unavailable items, and visualizes the optimal route via interactive graphs. Experimental results indicate that GA effectively minimize total costs by adapting to product availability constraints, while A* provides robust pathfinding for fixed starting points. This agent offers a practical solution for cost-conscious shopping, with potential extensions to broader logistics applications.**

**Keywords: Shopping optimization, Simulated Annealing, Ant Colony Optimization, Genetic Algorithm, A*, cost minimization, route planning.**

## I. Introduction

Shopping in a metropolitan area like Ankara can be complex due to varying product prices across different stores and the significant impact of transportation costs on overall spending. This project seeks to create a tool that minimizes the total cost for a given shopping list—taking into account both item prices and travel expenses—while also addressing the challenge that not all products are available at every store. Unlike conventional route optimization approaches that focus on time efficiency, this work emphasizes cost-effectiveness, offering users a budget-conscious and practical shopping plan.

The problem is framed as a constrained optimization challenge: given a shopping list, identify the subset of stores that collectively offer all available items at the lowest total cost, including travel between them. The tool leverages a synthetic dataset simulating Ankara's stores, with attributes such as coordinates, product offerings, product categories and prices. Four optimization algorithms—Simulated Annealing (SA), Ant Colony Optimization (ACO), Genetic Algorithm (GA), and A*—are implemented to solve this problem. Each algorithm models the store network as a weighted graph, where edges represent travel costs, and nodes are evaluated based on product availability and pricing.

The developed tool is designed with usability in mind, allowing users to input their shopping lists and starting locations interactively. It informs users of any unavailable items and generates an optimal route among the stores that provide the remaining products. Results are presented both textually and visually through interactive graphs. This paper evaluates the performance of the four algorithms and provides insights into their applicability for cost-driven shopping optimization.

## II. Literature Revİew

- Route optimization has been extensively explored in fields like operations research, logistics, and network systems, commonly modeled as variations of the Traveling Salesman Problem (TSP) or Vehicle Routing Problem (VRP). To tackle the complexity of these problems, researchers have successfully applied heuristic and metaheuristic algorithms, including Genetic Algorithms (GA), Simulated Annealing (SA), A*, and Ant Colony Optimization (ACO).
- Ito [1] demonstrates the use of a Genetic Algorithm for planning piping routes in industrial plant layouts. In this method, each gene represents a pipe path, and the algorithm evolves solutions using crossover and a fitness function based on spatial energy and path length. This interactive and adaptive approach shows how GAs can handle spatial constraints—an insight we adopt when optimizing the sequence of store visits in our cost-aware shopping routes.
- Grabusts et al. [2] apply Simulated Annealing to find the shortest route connecting dairy enterprises in Belarus, modeling it as a TSP. The algorithm effectively avoids local minima and determines an optimal 648 km path among eight locations using GPS data. This highlights SA's strength in spatial optimization, which we adapt to minimize total shopping costs (not just distance) in our project.
- Xia et al. [3] propose a hybrid algorithm combining Ant Colony Optimization with Bidirectional A* (BiA*-ACO) for taxi route planning using large-scale GPS trajectory data. This method significantly improves route efficiency—achieving 47
- The study "Multiple Ant-Colony Optimization for Network Routing" [4] introduces Multiple Ant-Colony Optimization (MACO), where several ant colonies function as agents that update routing tables in real-time to handle network congestion. While focused on telecommunica-

tions, this multi-agent approach provides valuable concepts for adapting ACO to dynamic shopping scenarios where availability and prices may change.
- Mukherjee and Acharyya [5] explore three ACO variations for optimizing throughput and reliability in network routing. By introducing a Tabu list, the algorithms avoid cycles and improve performance in large-scale networks. This idea can be translated to shopping route optimization, where many stores and constraints may exist.
- In summary, these studies show the flexibility and effectiveness of metaheuristic methods like GA, SA, A*, and ACO in solving complex optimization problems across diverse domains. Our project extends these techniques to a practical real-world problem—optimizing shopping routes in an urban environment by minimizing total cost, which includes both product prices and transportation expenses. We also emphasize user interaction and visualization to make the solution accessible and useful for everyday shoppers.

## III. Dataset Description

The synthetic dataset, tum_sentetik2.csv, simulates Ankara's shopping ecosystem, generated using Python with NumPy and Pandas. Below, detailed its attributes, structure, and creation process.

### A. Dataset Overview

- Purpose: Represents a diverse set of stores and products to test shopping route optimization algorithms.
- Size: Contains 8528 rows, each representing a store-product pair across 250 unique stores.
- Categories: Five distinct product categories:
  - Gıda (Food): 100 products (e.g., Süt, Peynir, Makarna).
  - Elektronik (Electronics): 86 products (e.g., Laptop, Telefon, SSD).
  - Giysi (Clothing): 93 products (e.g., Kot Pantolon, Tişört, Mont).
  - Kozmetik (Cosmetics): 91 products (e.g., Ruj, Şampuan, Nemlendirici).
  - Ev (Household): 82 products (e.g., Bulaşık Deterjanı, Matkap, Halı).
- Stores: 250 total, with 50 stores per category, named systematically (e.g., Gida_Magaza_0, Giysi_Magaza_1).

### B. Attributes

The dataset has five columns:

- Mağaza (Store): String identifier (e.g., "Gida_Magaza_42"), indicating the store and its category.
- X_Koordinat (X Coordinate): Float, representing the store's x-position in a 100x100 grid.
- Y_Koordinat (Y Coordinate): Float, representing the store's y-position in a 100x100 grid.
- Ürün (Product): String, the product offered by the store (e.g., "Laptop", "Süt").
- Fiyat (Price): Float, the price in Turkish Lira (TL), varying by category.
- Color : Represents the color of product based on its category (used in representation of different categories on graph)

### C. Data Generation Process

- Product Lists:
  - Five NumPy arrays were created for each category, with lengths: Gıda , Elektronik, Giysi , Kozmetik, Ev.
  - Products were manually defined to reflect realistic offerings (e.g., Elektronik includes "Laptop" and "SSD", Gıda includes "Süt" and "Makarna").
- Store Creation:
  - 50 stores per category were generated (total 250), with names prefixed by category (e.g., "Kozmetik_Magaza_0").
  - Coordinates (x,y)(x,y) were assigned randomly within a 100x100 grid.
- Product Assignment:
  - Each store was assigned 20–50 products from its category's array.
- Price Assignment:
  - Prices were randomly set per store-product pair in specific ranges:
    - Gıda: 10–200 TL.
    - Elektronik: 100–10,000 TL.
    - Giysi: 50–2,000 TL.
    - Kozmetik: 10–1,000 TL.
    - Ev: 100–1,000 TL.

### D. Characteristics

- Diversity: Covers everyday needs (food, clothing) and specialized items (electronics, cosmetics).
- Spatial Distribution: Randomized coordinates simulate Ankara's urban layout within a simplified grid.
- Price Variability: Reflects realistic economic ranges, enabling cost-based optimization testing.

This dataset provides a robust foundation for evaluating route optimization algorithms, though it lacks real-world dynamics like traffic or store hours.

## IV. Methodology

### A. Problem Definition

Given a shopping list $S = \{s1, s2, ..., sn\}$, a store set $M = \{m1, m2, ..., mk\}$ from tum_sentetik2.csv with coordinates $(xi, yi)$, and prices $P(mi, sj)$, find a route $R = [mr1, mr2, ..., mrl]$ that:

- Procures all items in $S$.
- Minimizes total cost

$$C = Cp + Ct$$

where $Cp = \sum P(mi, sj)$ and $Ct = \sum tij \times 100$ (travel time in hours × 100 TL/hour). Travel time $tij$ is Euclidean distance divided by 60 units/hour.

*B. Algorithm Implementations*

In this study, four algorithms implemented for route optimization—one heuristic (A*) and three metaheuristic methods (Genetic Algorithm, Simulated Annealing, and Ant Colony Optimization)

*1) **A* Algorithm**:* The A* algorithm is a heuristic-based search method that finds the least-cost path from a start node to a goal, adapted here to optimize a shopping route across stores in tum_sentetik2.csv. It balances exploration and exploitation using three key components: the cost-so-far $(g)$, the heuristic estimate $(h)$, and the total estimated cost $(f = g + h)$. In this implementation, the goal is not a fixed end node but a state where all shopping list items are procured at minimal total cost (product prices plus travel expenses).

- Graph Representation:
  - A graph $Gastar = (V, E)$ is constructed using NetworkX, where $V$ is the set of stores offering at least one item from the shopping list $S$, filtered from the dataset. Each node has attributes: position $(xi, yi)$ and category (e.g., Gıda, Elektronik).
  - Edges $E$ connect all pairs of stores, weighted by travel time $tij = \frac{euclidean((xi,yi),(xj,yj))}{60}$ (hours, assuming a speed of 60 units/hour). Travel cost is $tij \times 100$ TL.
- Inputs:
  - Start node: Randomly selected from $V$
  - Shopping list $S$: User-defined subset of available products.
- Cost Components:
  - $g(n)$ (Cost-so-far): Cumulative cost from the start node to the current node $n$, comprising:
    - Travel cost: Sum of $tij \times 100$for each edge traversed.
    - Product cost: Sum of prices $P(mi, sj)$ for items assigned to stores along the path.
  - $h(n)$ (Heuristic): Estimated cost from $n$ to a "goal state" (all items collected). Here, it's defined as the Euclidean distance from $n$ to the start node divided by 60 (hours), multiplied by 100 TL/hour:

$$h(n) = \frac{euclidean(pos[n], pos[start])}{60} \times 100$$

    - This heuristic assumes a round-trip scenario, underestimating the remaining cost to close the route, ensuring admissibility (never overestimating true cost).
  - $f(n) = g(n) + h(n)$: Total estimated cost, guiding the search toward the optimal path.

*2) **Genetic Algorithm**:* The Genetic Algorithm (GA) is an evolutionary optimization method inspired by natural selection, adapted here to solve the shopping route problem using tum_sentetik2.csv. It evolves a population of candidate solutions over generations, optimizing for minimal total cost (product prices plus travel expenses) by iteratively applying selection, crossover, and mutation operations. Each individual encodes both an item-to-store mapping and a store visitation order, drawing from Ito's [1] approach to route representation.

- Graph Representation:
  - A graph $Gga = (V, E)$ is constructed using NetworkX, where $V$ is the set of stores offering at least one item from the shopping list $S$, filtered from the dataset. Each node has attributes: position $(xi, yi)$ and category (e.g., Gıda, Elektronik).
  - Edges $E$ connect all pairs of stores, weighted by travel time $tij = \frac{euclidean((xi,yi),(xj,yj))}{60}$ (hours, assuming a speed of 60 units/hour). Travel cost is $tij \times 100$ TL.
- Individual Representation: An individual is a tuple (store_of_item,store_order) where:
  - store_of_item : decides which item should be obtained from which store
  - store_order : decides the selected stores should be visited in which order
- Population Initialization: Greedily select stores to cover all items in $S$. Iteratively pick the store $mi$ with the most uncovered items. For store order randomly shuffles the selected stores.
- Fitness Function: New generations are obtained by fitness function which calculates cost:

$$C = Cp + Ct$$

  where $Cp$: sum of product prices for each pair, $Ct$: sum of $tij \times 100$ between consecutive stores in ordered stores.
- Selection: The selection mechanism is relies on elitist selection. This mechanism selects individuals based on its feasibility and its quality. It sorts by $C$ value and selects the top $n/2$ of population
- Crossover: For item's store selection uniform crossover is used. For store order determination two point crossover is used.
- Mutation: For item's store selection with probability of 0.1, it reassigns the item to a random store. For store order determination with with probability of 0.1, it swaps two store's positions.

*3) **Simulated Annealing**:* Simulated Annealing (SA) is a stochastic optimization technique inspired by the annealing process in metallurgy, adapted here to minimize the total cost (product prices plus travel expenses) for shopping route planning using tum_sentetik2.csv. It iteratively refines a single solution by exploring neighboring states, accepting worse solutions probabilistically to escape local minima, as demonstrated in Grabusts et al. [2] for TSP optimization.

- Graph Representation:
  - A graph $Gsa = (V, E)$ is constructed using NetworkX, where $V$ is the set of stores offering at least one item from the shopping list $S$, filtered from the dataset. Each node has attributes: position $(xi, yi)$ and category (e.g., Gıda, Elektronik).

  - Edges $E$ connect all pairs of stores, weighted by travel time $tij = \frac{euclidean((xi,yi),(xj,yj))}{60}$ (hours, assuming a speed of 60 units/hour). Travel cost is $tij \times 100$ TL.
- Solution Representation: A solution is a tuple (store_of_item,store_order) where:
  - store_of_item : decides which item should be obtained from which store
  - store_order : decides the selected stores should be visited in which order
- Initial Solution:
  - Store of items initalization: For each item $sj \in S$, selects the store $mi$ with the lowest price $P(mi, sj)$
  - Route Initialization: Randomly shuffles the selected stores.
- Cost Function: New neighbors are evaluated by cost function which calculates cost:

$$C = Cp + Ct$$

  where $Cp$: sum of product prices for each pair, $Ct$: sum of $tij \times 100$ between consecutive stores in ordered stores.
- Neighbor Definition: Neighbors are defined by 4 randomly selected moves:
  - Swap Route: Swap position of 2 stores
  - Replace Route: Remove a random store and replace it with a valid random store.
  - Change Store: For a random item, reassign to a different store
  - Merge Category: For a random category select the store minimizing the total cost of all items in that category.
- Neighbor Acceptance Rule: accept neighbor if:
  - neighbor_cost¡current_cost (improvement), or
  - Probabilistic acceptance: a random number $\leq e^{(neighbor\_cost - current\_cost)/T}$

*4) **Ant Colony Optimization**:* Ant Colony Optimization (ACO) is a metaheuristic inspired by the foraging behavior of ants, leveraging pheromone trails to solve combinatorial optimization problems. Here, it is adapted to optimize shopping routes in tum_sentetik2.csv, minimizing total cost (product prices plus travel expenses). Drawing from Xia et al. [3], "Multiple Ant-Colony Optimization" [4], and Mukherjee and Acharyya [6], this implementation introduces a "Home" (Ev) node to transform the open-path shopping problem into a closed-loop TSP-like structure, enhancing ACO's applicability.

- Graph Representation:
  - A graph $Gaco = (V, E)$ is constructed using NetworkX, where $V$ is the set of stores offering at least one item from the shopping list $S$, filtered from the dataset. Each node has attributes: position $(xi, yi)$ and category (e.g., Gıda, Elektronik).
  - Edges $E$ connect all pairs of stores, weighted by travel time $tij = \frac{euclidean((xi,yi),(xj,yj))}{60}$ (hours, assuming a speed of 60 units/hour). Travel cost is $tij \times 100$ TL.
  - Purpose of "Ev": Converts an open-path problem (start anywhere, collect items) into a TSP-style closed loop (start at "Ev", visit stores, return to "Ev"), simplifying ACO's path-building logic, as closed loops align with ACO's strengths in TSP optimization [5].
- Solution Representation:
  - A solution is a route: a list of nodes starting and ending at "Ev".
  - Items are implicitly assigned to the first store in the route offering them, determined during cost calculation.
- Cost Function: New solutions are evaluated by cost function which calculates cost:

$$C = Cp + Ct$$

  where $Cp$: sum of product prices for each pair, $Ct$: sum of $tij \times 100$ between consecutive stores in ordered stores.
- Algorithm Workflow:
  - Initialization:Pheromone levels are set to 1.0 for all edges in $Gaco$
  - Iterations:
    - In each iteration, 10 ants independently construct their routes. Every ant begins its journey at "Ev" (home). While there are still items left to collect, the ant considers unvisited stores (excluding "Ev") that offer at least one of the remaining items. For each candidate store $mj$ the probability of selection is calculated based on the pheromone level and a heuristic value.
    - Moving from the current store $mi$ to a candidate store $mj$ has a probability. This probability is proportional to pheromone intensity and heuristic value. $h(mi, mj) = 1/(tij + avg\_price/100)$: Heuristic, where $tij$ is travel time and avg_price is the average price of missing items at $mj$.
    - The next store is selected based on these probabilities using a weighted random choice. If the total probability across all candidates is zero, a store is chosen randomly from the available options. Once a store is selected, it is added to the route, and the ant updates the list of items it has collected. The route construction continues until all items are collected, at which point the ant returns to "Ev". After completing the route, the algorithm computes the total cost for that ant's path, including both travel and item costs, as well as the specific store each item was purchased from.

## V. Test Results

This section evaluates the performance of four optimization algorithms—A*, Genetic Algorithm (GA), Simulated Anneal-

ing (SA), and Ant Colony Optimization (ACO)—for solving the shopping route problem using the tum_sentetik2.csv dataset. The tests assess each algorithm's ability to minimize total cost ($C = Cp + Ct$), where $Cp$ is the product cost and $Ct$ is the travel cost (100 TL per hour). Visualizations and detailed analyses provide insights into route formation and item sourcing strategies.

### A. Test Setup

The shopping list $S$ was consistently defined across all algorithms to ensure a fair comparison:

- $$S = \left\{ \begin{array}{l} \text{Avakado, Labne, Patates, Sucuk, Mont,} \\ \text{Tişört, Ruj, Laptop, Makas, Bulaşık Deterjanı} \end{array} \right\}$$
- Categories (dynamically derived from the dataset):
  - Gıda: Avakado, Labne, Patates, Sucuk
  - Giysi: Mont, Tişört
  - Kozmetik: Ruj
  - Elektronik: Laptop
  - Ev: Makas, Bulaşık Deterjanı
- Parameters :
  - A*: Heuristic based on minimum spanning tree (MST) for remaining stores.
  - GA : Population size = 50, generations = 100, mutation rate = 0.1.
  - SA: Initial temperature = 1000, cooling rate = 0.995, iterations = 1000.
  - ACO: 10 ants, 100 iterations, $\alpha$ =1.0, $\beta$ =2.0, evaporation rate = 0.5, Q=100.
  - Special Note for ACO: A synthetic "Ev" (Home) node was added at coordinates (0.0, 0.0) to transform the problem into a TSP-like closed loop, starting and ending at "Ev".

### B. Algorithm Performance

***A* Algorithm:***

**Total Cost**

**Total Cost:** 2809.45 TL
**Product Cost (Cp):** 1643.38 TL
**Travel Cost (Ct):** 1166.07 TL

| Step | Store |
|---|---|
| Start | Elektronik_Magaza_20 |
| 1 | Giysi_Magaza_33 |
| 2 | Ev_Magaza_3 |
| 3 | Gida_Magaza_0 |
| 4 | Kozmetik_Magaza_1 |
| 5 | Ev_Magaza_39 |
| 6 | Gida_Magaza_42 |
| 7 | Giysi_Magaza_44 |
| End | Elektronik_Magaza_4 |

TABLE I
A* ALGORITHM ROUTE SEQUENCE

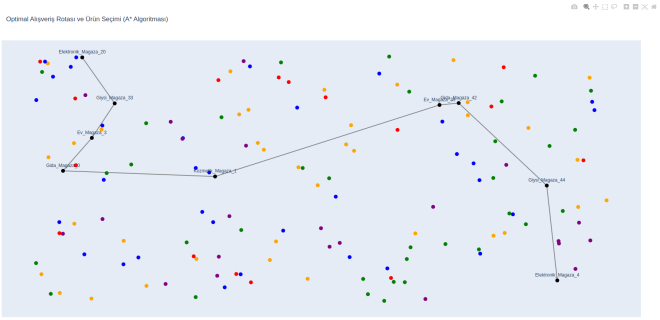

Fig. 1. Optimal Shopping Route and Item Selection (A* Algorithm)

***Genetic Algorithm:***

**Total Cost**

**Total Cost:** 1978.26 TL
**Product Cost (Cp):** 1583.98 TL
**Travel Cost (Ct):** 394.28 TL

| Step | Store |
|---|---|
| 1 | Gida_Magaza_0 |
| 2 | Kozmetik_Magaza_1 |
| 3 | Gida_Magaza_31 |
| 4 | Giysi_Magaza_24 |
| 5 | Ev_Magaza_42 |
| 6 | Elektronik_Magaza_4 |
| 7 | Giysi_Magaza_44 |
| 8 | Gida_Magaza_18 |
| 9 | Gida_Magaza_2 |
| 10 | Ev_Magaza_39 |

TABLE II
GENETIC ALGORITHM ROUTE SEQUENCE

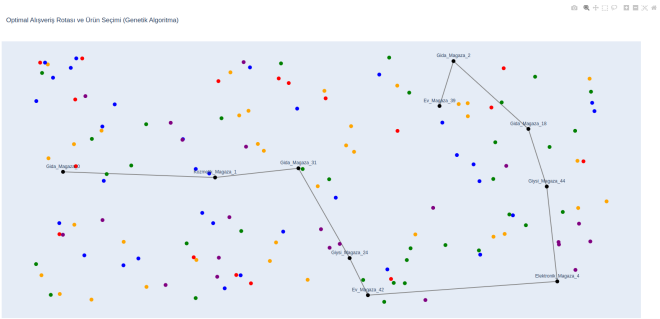

Fig. 2. Optimal Shopping Route and Item Selection (Genetic Algorithm)

***Simulated Annealing (SA):***

Total Cost

**Total Cost:** 2291.45 TL
**Product Cost (Cp):** 1814.39 TL
**Travel Cost (Ct):** 477.06 TL

| Step | Store |
|---|---|
| 1 | Ev_Magaza_29 |
| 2 | Elektronik_Magaza_4 |
| 3 | Giysi_Magaza_44 |
| 4 | Elektronik_Magaza_4 |
| 5 | Gida_Magaza_42 |
| 6 | Ev_Magaza_39 |
| 7 | Kozmetik_Magaza_1 |
| 8 | Giysi_Magaza_33 |
| 9 | Gida_Magaza_27 |

TABLE III
SIMULATED ANNEALING ROUTE SEQUENCE

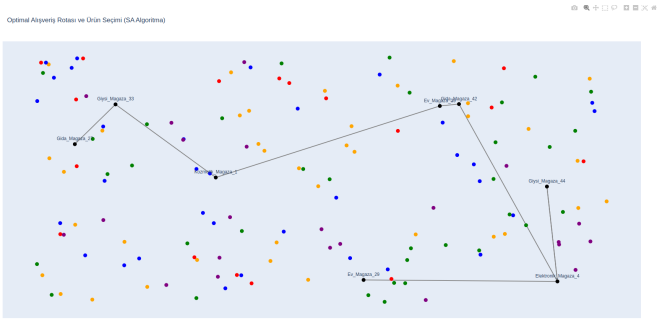


Fig. 3. Optimal Shopping Route and Item Selection (Simulated Annealing)

***Ant Colony Optimization (ACO):***

Total Cost

**Total Cost:** 2343.19 TL
**Product Cost (Cp):** 1699.00 TL
**Travel Cost (Ct):** 644.19 TL

| Step | Store |
|---|---|
| Start | Ev |
| 1 | Gida_Magaza_8 |
| 2 | Kozmetik_Magaza_38 |
| 3 | Gida_Magaza_39 |
| 4 | Gida_Magaza_45 |
| 5 | Ev_Magaza_28 |
| 6 | Gida_Magaza_18 |
| 7 | Ev_Magaza_39 |
| 8 | Elektronik_Magaza_4 |
| 9 | Giysi_Magaza_44 |
| 10 | Giysi_Magaza_24 |
| End | Ev |

TABLE IV
ANT COLONY OPTIMIZATION ROUTE SEQUENCE

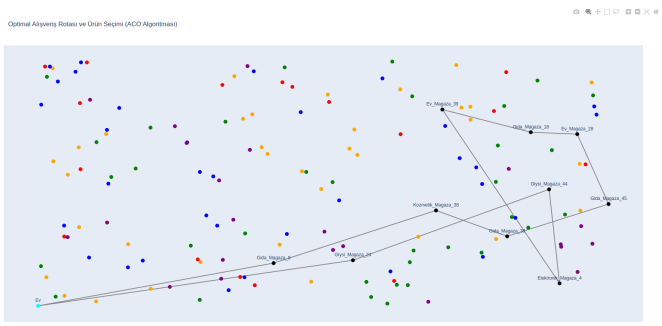


Fig. 4. Optimal Shopping Route and Item Selection (Ant Colony Optimization)

ACO's use of the "Ev" node introduces distinct constraints compared to A*, GA, and SA:

- Closed Loop Requirement:ACO transforms the problem into a TSP-like structure by starting and ending at "Ev". This ensures a complete tour but forces the route to return to the starting point, potentially increasing $Ct$.
- Impact on Route Formation:The fixed start/end at "Ev" (coordinates (0, 0)) constrains the route's flexibility. A*, GA, and SA can end anywhere, optimizing for the shortest open path. ACO's heuristic h(n) and pheromone updates prioritize paths that collect items efficiently, but the return to "Ev" can lead to detours, as seen in the visualization (Fig. 4).
- Evaluation Difference:ACO's performance should be evaluated with its constraint in mind. While it ensures a return to the starting point (realistic for some scenarios), it may not always yield the shortest path in terms of $Ct$.

graphicx [table,xcdraw]xcolor

### C. Comparative Insights

The algorithms' performance highlights their distinct strategies and trade-offs:

*1) Cost Efficiency::* GA's total cost (1978.26 TL) is 13.7% lower than SA's (2291.45 TL), 15.6% lower than ACO's (2343.19 TL), and 29.6% lower than A*'s (2809.45 TL). GA's

| Item | A* Source (Price, TL) | GA Source (Price, TL) | SA Source (Price, TL) | ACO Source (Price, TL) |
|---|---|---|---|---|
| Avakado | Gida_Magaza_0 (16.01) | Gida_Magaza_0 (16.01) | Gida_Magaza_27 (51.99) | Gida_Magaza_39 (92.35) |
| Labne | Gida_Magaza_42 (90.20) | Gida_Magaza_18 (52.29) | Gida_Magaza_42 (90.20) | Gida_Magaza_18 (52.29) |
| Patates | Gida_Magaza_42 (37.71) | Gida_Magaza_31 (56.86) | Gida_Magaza_42 (37.71) | Gida_Magaza_45 (10.57) |
| Sucuk | Gida_Magaza_42 (70.62) | Gida_Magaza_2 (15.55) | Gida_Magaza_42 (70.62) | Gida_Magaza_8 (25.64) |
| Mont | Giysi_Magaza_33 (225.64) | Giysi_Magaza_24 (225.64) | Giysi_Magaza_33 (225.64) | Giysi_Magaza_24 (225.64) |
| Tişört | Giysi_Magaza_44 (254.18) | Giysi_Magaza_44 (254.18) | Giysi_Magaza_44 (254.18) | Giysi_Magaza_44 (254.18) |
| Ruj | Kozmetik_Magaza_1 (71.81) | Kozmetik_Magaza_1 (71.81) | Kozmetik_Magaza_1 (71.81) | Kozmetik_Magaza_38 (71.81) |
| Laptop | Elektronik_Magaza_4 (520.15) | Elektronik_Magaza_4 (520.15) | Elektronik_Magaza_4 (520.15) | Elektronik_Magaza_4 (520.15) |
| Makas | Ev_Magaza_3 (112.88) | Ev_Magaza_42 (171.99) | Ev_Magaza_29 (247.91) | Ev_Magaza_28 (171.99) |
| Bulaşık Deterjanı | Ev_Magaza_39 (244.18) | Ev_Magaza_39 (244.18) | Ev_Magaza_39 (244.18) | Ev_Magaza_39 (244.18) |

TABLE V
ITEM SOURCING BY ALGORITHM AND PRICE

| Algorithm | Route | Total Cost (TL) | Product Cost (Cp) | Travel Cost (Ct) |
|---|---|---|---|---|
| A* | Elektronik_Magaza_20 → Giysi_Magaza_33 → Ev_Magaza_3 → Gida_Magaza_0 → Kozmetik_Magaza_1 → Ev_Magaza_39 → Gida_Magaza_42 → Giysi_Magaza_44 → Elektronik_Magaza_4 | 2809.45 | 1643.38 | 1166.07 |
| GA | Gida_Magaza_0 → Kozmetik_Magaza_1 → Gida_Magaza_31 → Giysi_Magaza_24 → Ev_Magaza_42 → Elektronik_Magaza_4 → Giysi_Magaza_44 → Gida_Magaza_18 → Gida_Magaza_2 → Ev_Magaza_39 | 1978.26 | 1583.98 | 394.28 |
| SA | Ev_Magaza_29 → Elektronik_Magaza_4 → Giysi_Magaza_44 → Elektronik_Magaza_4 → Gida_Magaza_42 → Ev_Magaza_39 → Kozmetik_Magaza_1 → Giysi_Magaza_33 → Gida_Magaza_27 | 2291.45 | 1814.39 | 477.06 |
| ACO | Ev → Gida_Magaza_8 → Kozmetik_Magaza_38 → Gida_Magaza_39 → Gida_Magaza_45 → Ev_Magaza_28 → Gida_Magaza_18 → Ev_Magaza_39 → Elektronik_Magaza_4 → Giysi_Magaza_44 → Giysi_Magaza_24 → Ev | 2343.19 | 1699.00 | 644.19 |

TABLE VI
SUMMARY OF ROUTES AND COSTS BY ALGORITHM

low $Ct$ (394.28 TL) reflects its effective route clustering, while A*'s high $Ct$ (1166.07 TL) underscores its greedy limitations. SA and ACO fall in between, with ACO's higher $Ct$ (644.19 TL) due to the closed-loop constraint.

*2) Item Sourcing:* GA and ACO generally selected cheaper sources (e.g., "Sucuk" at 15.55 TL and 25.64 TL, respectively), while A* and SA opted for more expensive options (70.62 TL). ACO's first-encountered assignment can lead to higher $Cp$ if cheaper stores are visited later, whereas GA and SA explicitly optimize item-to-store mappings. A*'s greedy sourcing balances $Cp$ but doesn't globally optimize.

## VI. CONCLUSION

This study implemented and compared four optimization algorithms—A* (A*), Genetic Algorithm (GA), Simulated Annealing (SA), and Ant Colony Optimization (ACO)—to solve the shopping route optimization problem using the a synthetic dataset. The goal was to minimize the total cost, balancing product prices and travel expenses, for a shopping list comprising diverse items across multiple categories. Through rigorous testing, detailed metrics, and visual analyses, the evaluation revealed distinct performance profiles for each algorithm. Notably, the Genetic Algorithm (GA) emerged as the most effective, achieving the lowest total cost of 1978.26 TL by efficiently optimizing both item sourcing and route planning. This conclusion summarizes the key outcomes, assesses the algorithms' strengths, and offers directions for future enhancements to improve their applicability in real-world shopping scenarios.

## REFERENCES


[1] T. Ito, "A Genetic Algorithm Approach to Piping Route Path Planning," *Journal of Intelligent Manufacturing*, vol. 10, no. 2, pp. 103–114, 1999.

[2] P. Grabusts, J. Musatovs, and V. Golenkov, "The Application of Simulated Annealing Method for Optimal Route Detection Between Objects," *Procedia Computer Science*, vol. 149, pp. 95–101, 2019.

[3] D. Xia, B. Shen, Y. Zheng, W. Zhang, D. Bai, Y. Hu, and H. Li, "A Bidirectional-A-Star-Based Ant Colony Optimization Algorithm for Big-Data-Driven Taxi Route Recommendation," *Multimedia Tools and Applications*, vol. 83, pp. 16313–16335, 2024. doi:10.1007/s11042-023-15498-4

[4] K. M. Sim and W. H. Sun, "Multiple Ant-Colony Optimization for Network Routing," in *Proc. of the First International Symposium on Cyber Worlds (CW'02)*, pp. 1–8, 2002.

[5] D. Mukherjee and S. Acharyya, "Ant Colony Optimization Technique Applied in Network Routing Problem," *International Journal of Computer Applications*, vol. 1, no. 15, pp. 74–80, 2010.


- Github Link:https://github.com/ozgeisklar/yap441_proje
- Youtube Link: https://www.youtube.com/watch?v=WLLtAEICA1c