TSP and VRP Problems with ACO and PSO

IE49.A Collective Computation
Instructor: Ebubekir Koç

Emirhan Şimşek
2021402189

github.com/Emirhan777/TSP_VRP_Using_ACO_PSO

January 16, 2026

Contents

Abstract **2**

1 Introduction **2**

2 Methods **3**
2.1 Traveling Salesman Problem (TSP) 4
2.1.1 TSP with ACO 4
2.1.2 TSP with PSO 10
2.1.3 TSP with Common Method 16
2.2 Vehicle Routing Problem (VRP) 17
2.2.1 VRP with ACO 17
2.2.2 VRP with PSO 23
2.2.3 VRP with Common Method 29

3 Comparison of Methods **30**

4 Discussion **31**

5 Conclusions **33**

6 References **34**

Abstract

I am interested in logistics problems in industrial engineering, especially classic routing problems such as the Traveling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP). In this project, I study a TSP-like formulation (TSE) with time-interval style constraints and solve it using Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and an exact dynamic programming method. I compare their performance and show how the metaheuristics converge to the same solution on small graphs while sometimes struggling to avoid suboptimal tours as the graph size and combinations grow.

1 Introduction

Logistics and routing problems sit at the core of industrial engineering because they directly affect cost, service quality, and operational feasibility. Classic formulations like the Traveling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP) capture essential constraints of distribution, pickup and delivery, and scheduling. Even small changes in graph size or constraints can significantly increase the search space, making exact optimization challenging as problem instances grow.

This report studies a TSP-like formulation (TSE) with a fixed start node and weighted directed edges that represent traversal costs and time-interval considerations. The same instance is solved with Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and an exact Held-Karp dynamic programming method. The goal is to compare solution quality, runtime behavior, and convergence patterns, and to highlight how metaheuristics can provide practical solutions when exact methods become computationally expensive.

2 Methods

All graphs use directed edges with weights w_{uv} that represent traversal cost or distance between nodes u and v. In the figures, the thin edges show the underlying graph connectivity; in ACO they are colored by pheromone intensity (plasma colormap, dark for low pheromone and yellow for high pheromone), while in PSO and exact methods they are drawn in gray. The thick semi-transparent edges indicate the best tour or the best routes found so far; for VRP these thick edges are colored per vehicle (red and green).
The problem instance uses 10 nodes (A–J) and 23 base connections. Each base edge is added in both directions, so the directed graph has 46 edges in total. Edge weights represent distances, and because every connection is two-directional, travel is allowed in both directions along each pair of nodes. The initial graph with all nodes, directed edges, and distances is shown below.

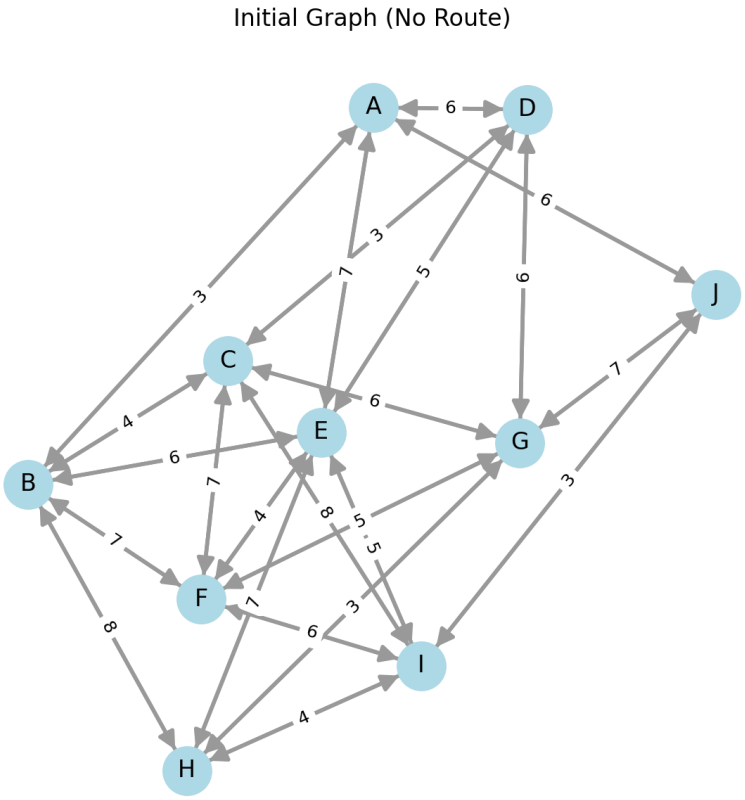

Figure 1: Initial problem graph with edge weights (distances).

2.1 Traveling Salesman Problem (TSP)

The TSP is defined on a weighted directed graph with a designated start node (depot), taken to be node A. The objective is to find a minimum-cost Hamiltonian cycle that starts at A, visits every other node exactly once, and returns to A. The constraints are: each node is visited exactly once, the tour is a single cycle, and the total cost is the sum of the selected directed edges.

2.1.1 TSP with ACO

Each ant builds a tour that starts at the depot, visits each node once, and returns to the start. The selection probability from node u to a candidate neighbor v is

$$p_{uv} = \frac{\tau_{uv}^{\alpha}\,\eta_{uv}^{\beta}}{\sum_{x \in N(u)} \tau_{ux}^{\alpha}\,\eta_{ux}^{\beta}}, \qquad \eta_{uv} = \frac{1}{w_{uv}},$$

where τ_{uv} is pheromone on edge (u, v) and w_{uv} is the edge weight. A tour cost is computed as $L = \sum_{(u,v)\in \text{tour}} w_{uv}$, pheromones evaporate with rate ρ, and reinforcement is added via

$$\tau_{uv} \leftarrow (1-\rho)\tau_{uv} + \sum_{k} \frac{Q}{L_k}$$

for tours k that used edge (u, v), with optional random exploration. Multiple restarts are run with different seeds and the best tour is retained.

Results: The ACO runs produced feasible tours that start at A, visit all nodes exactly once, and return to A. In this study, 5 restarts with 20 iterations each were run; the best tour and its cost are reported in the figures and used for comparison with the exact solution. Example iterations from each restart are shown below.

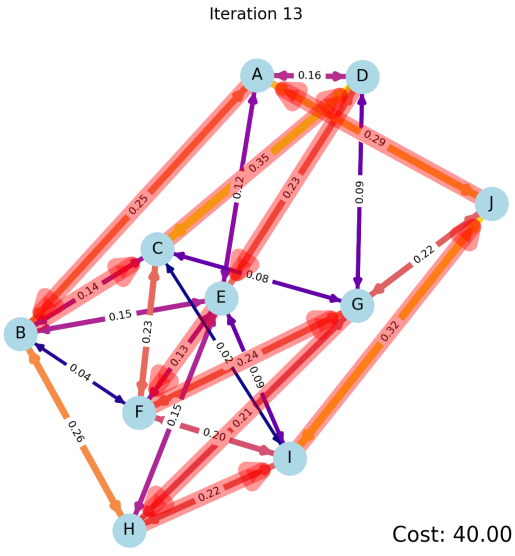

Figure 2: ACO Restart 1: iteration 13

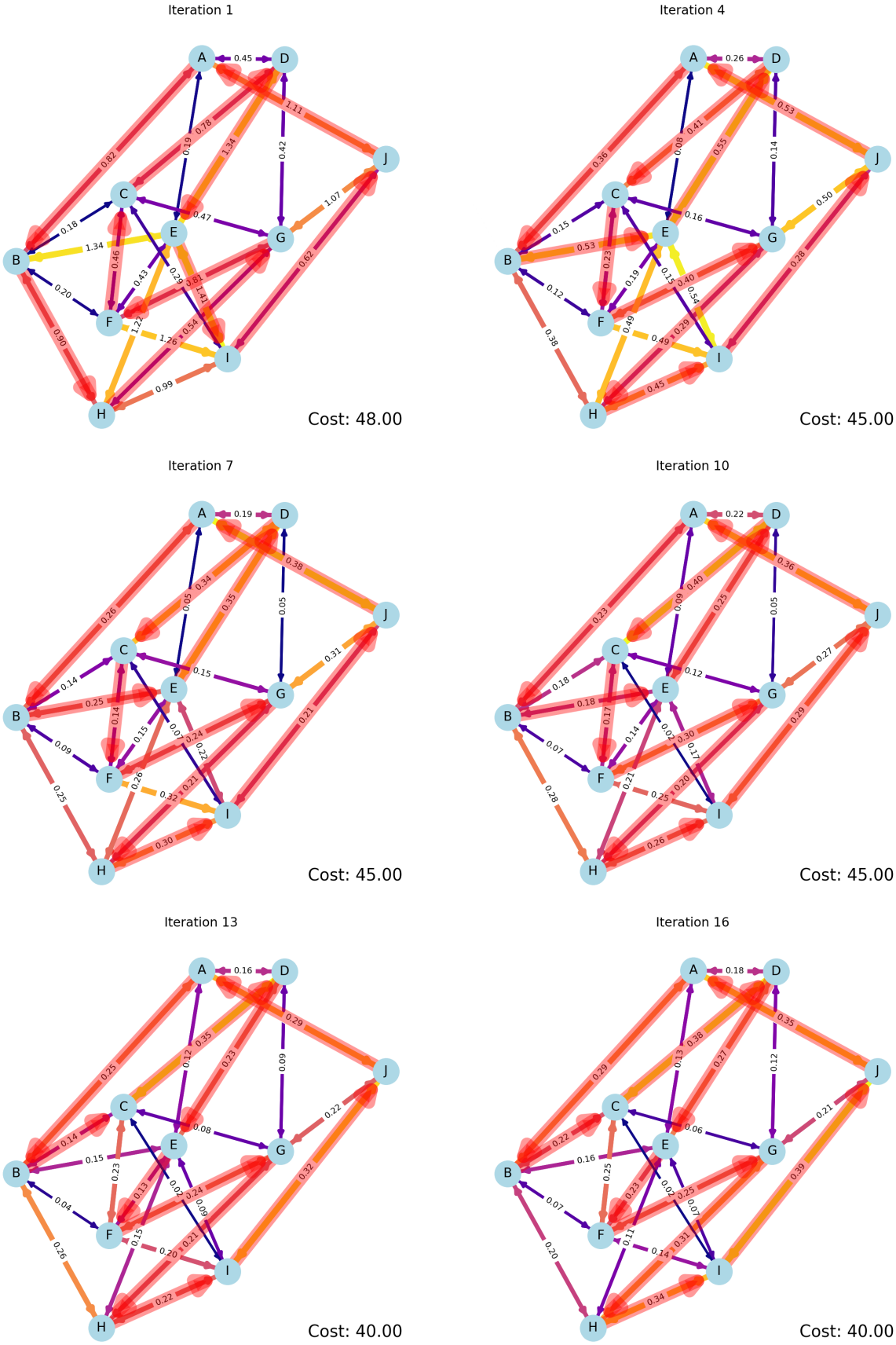

Figure 3: ACO Restart 1: iterations 1, 4, 7, 10, 13, 16

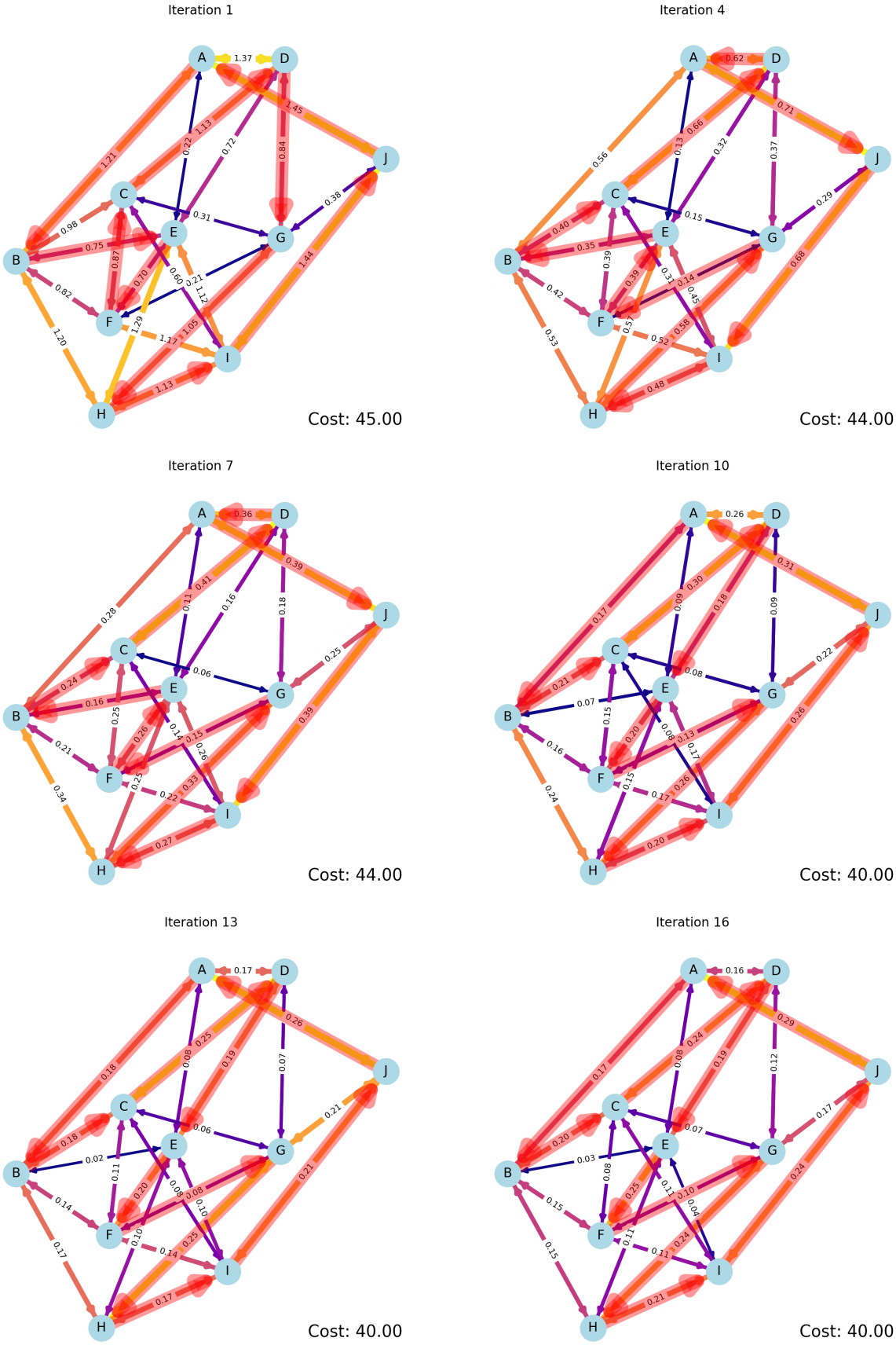

Figure 4: ACO Restart 2: iterations 1, 4, 7, 10, 13, 16

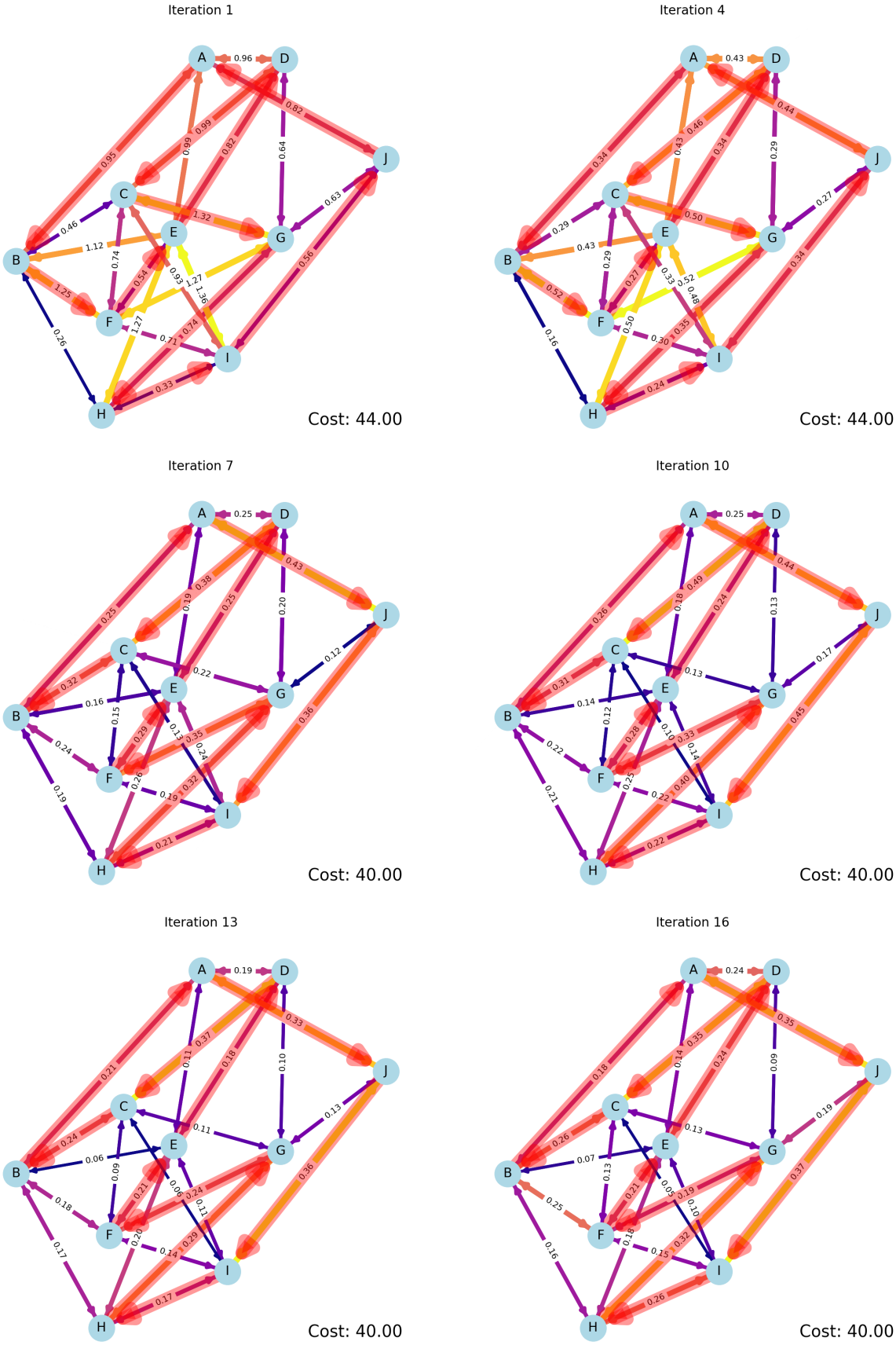

Figure 5: ACO Restart 3: iterations 1, 4, 7, 10, 13, 16

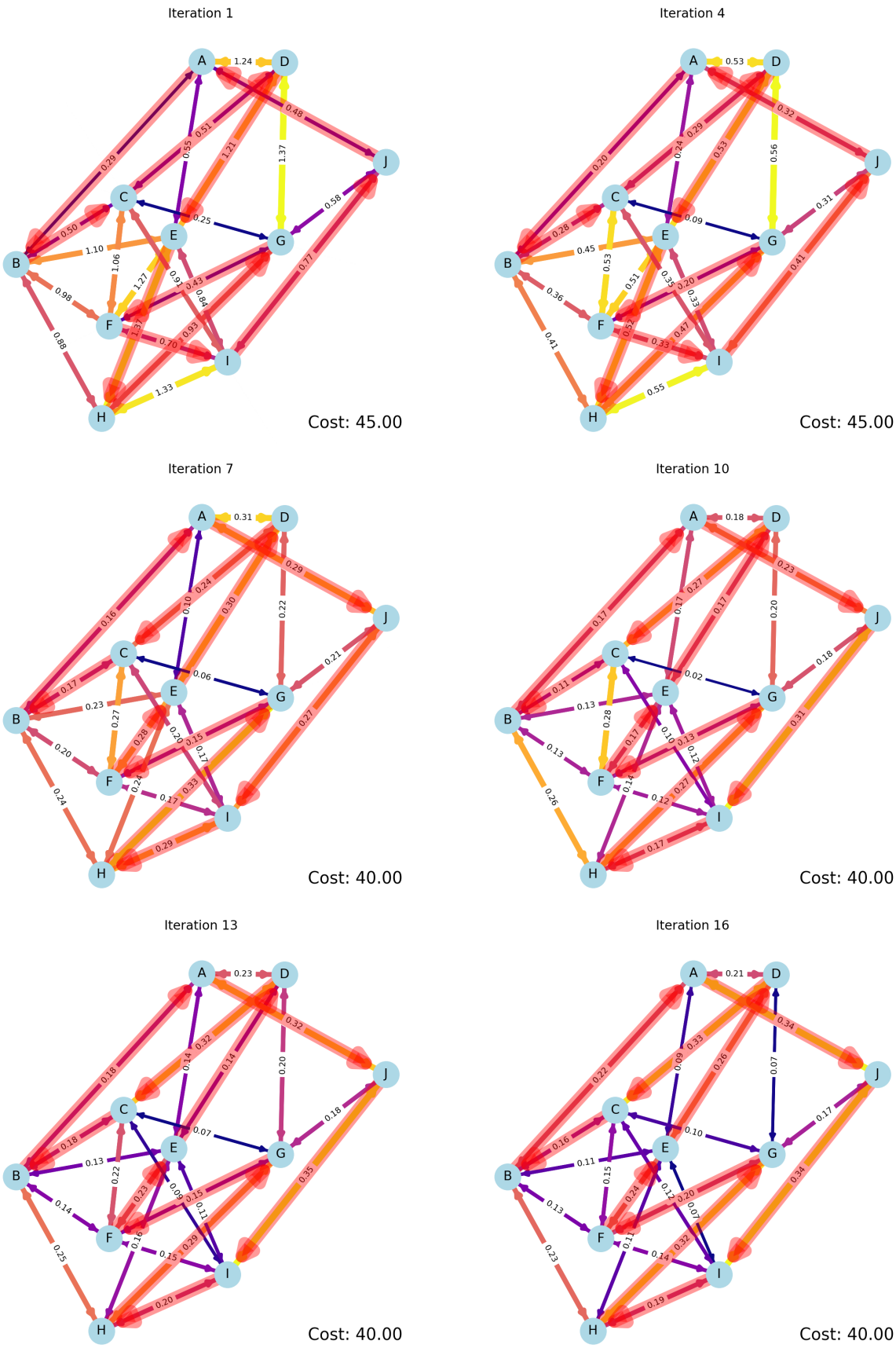

Figure 6: ACO Restart 4: iterations 1, 4, 7, 10, 13, 16

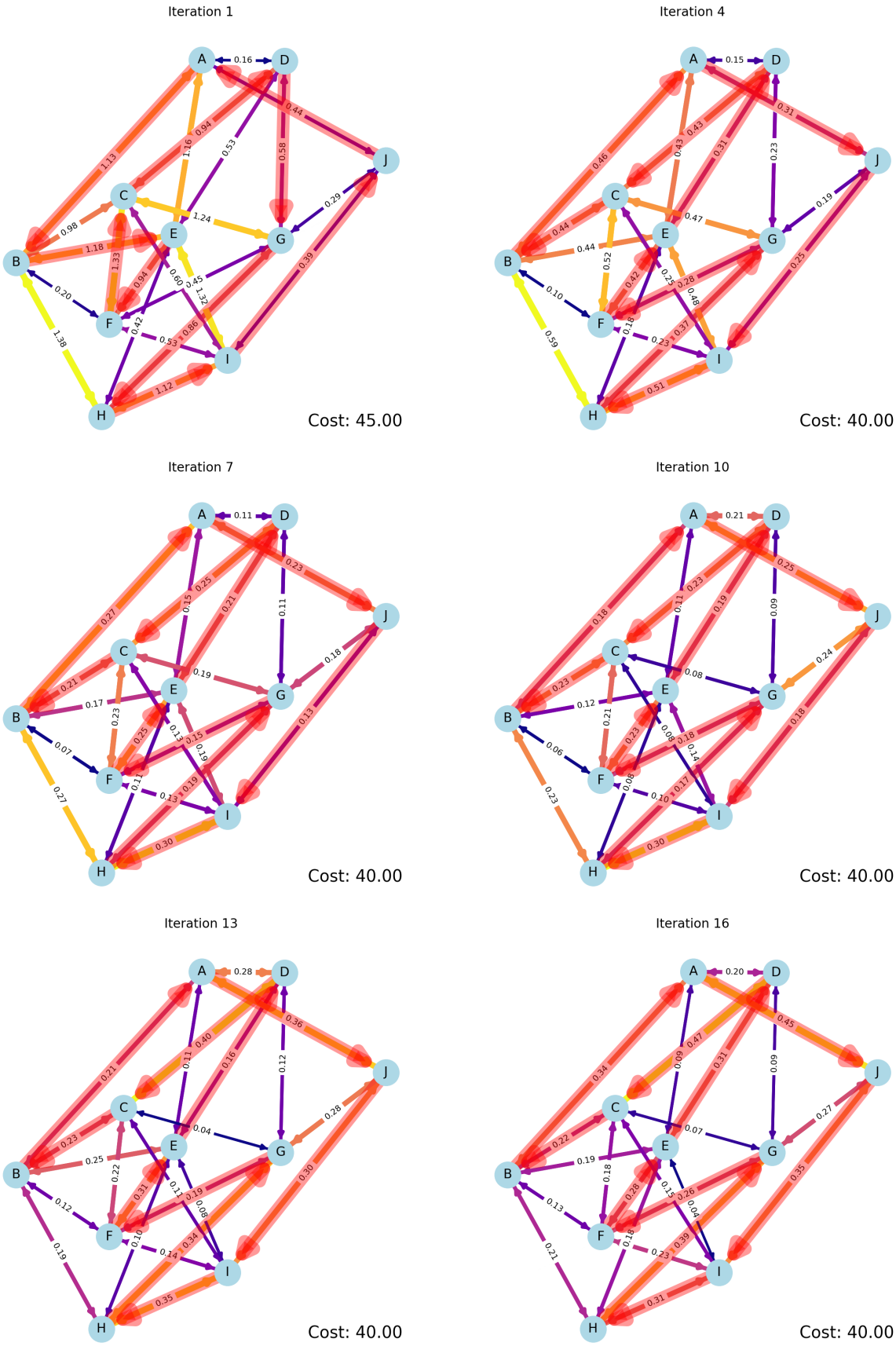

Figure 7: ACO Restart 5: iterations 1, 4, 7, 10, 13, 16

2.1.2 TSP with PSO

In PSO, each particle encodes a full permutation of nodes, which defines a tour cost $L = \sum_{(u,v)\in\text{tour}} w_{uv}$. Velocity is modeled as a sequence of swaps that transforms the current permutation toward the particle's best (pbest) and the global best (gbest); the updated tour is obtained by applying selected swaps, with occasional mutations and a local 2-opt improvement. The algorithm iterates until convergence, and multiple restarts are used to reduce the risk of local minima.

Results: The PSO runs produced feasible tours; the best tour found by the swarm is shown in the figures and used for comparison with the exact solution. Five restarts with 20 iterations each were run, and the best cost stabilized by the end of the iteration window in each restart. Some early costs were reported as ∞ because a particle's permutation implied a missing edge in this sparse directed graph, so the tour was infeasible. As swap updates, mutation, and the 2-opt repair step progressed, infeasible tours were replaced by feasible ones and the global best cost became finite. Example iterations from each restart are shown below.

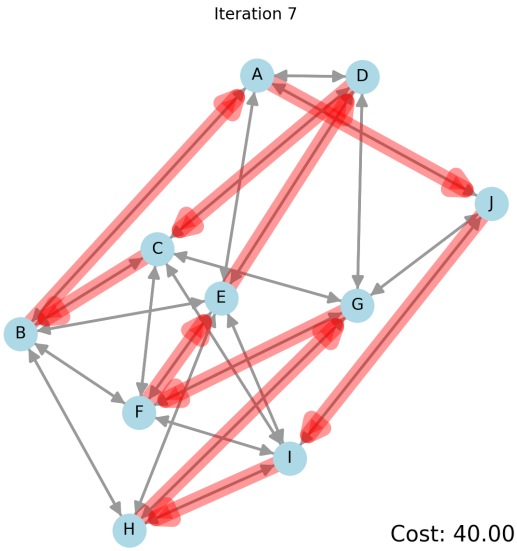

Figure 8: PSO Restart 1: iteration 7

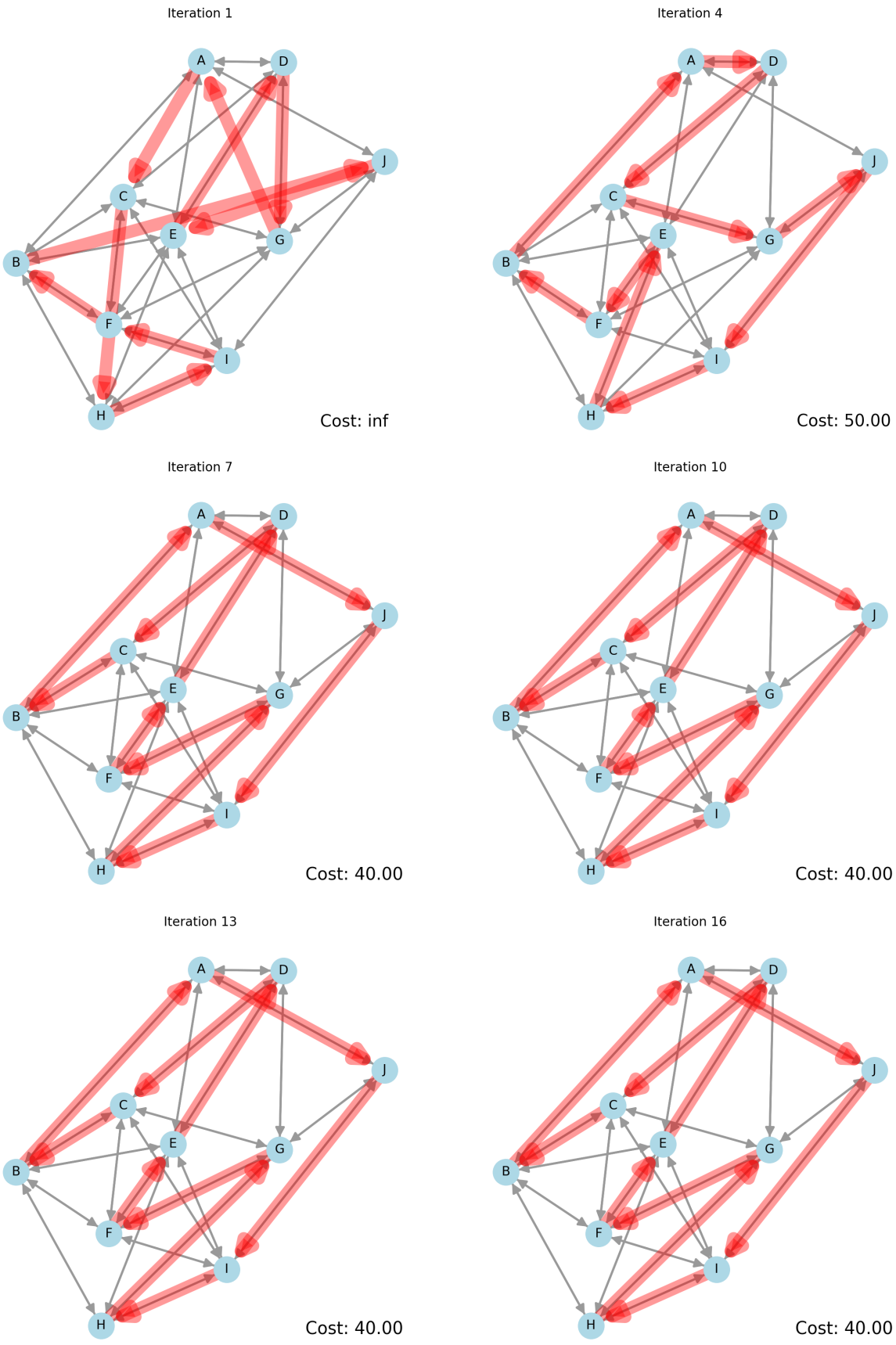

Figure 9: PSO Restart 1: iterations 1, 4, 7, 10, 13, 16

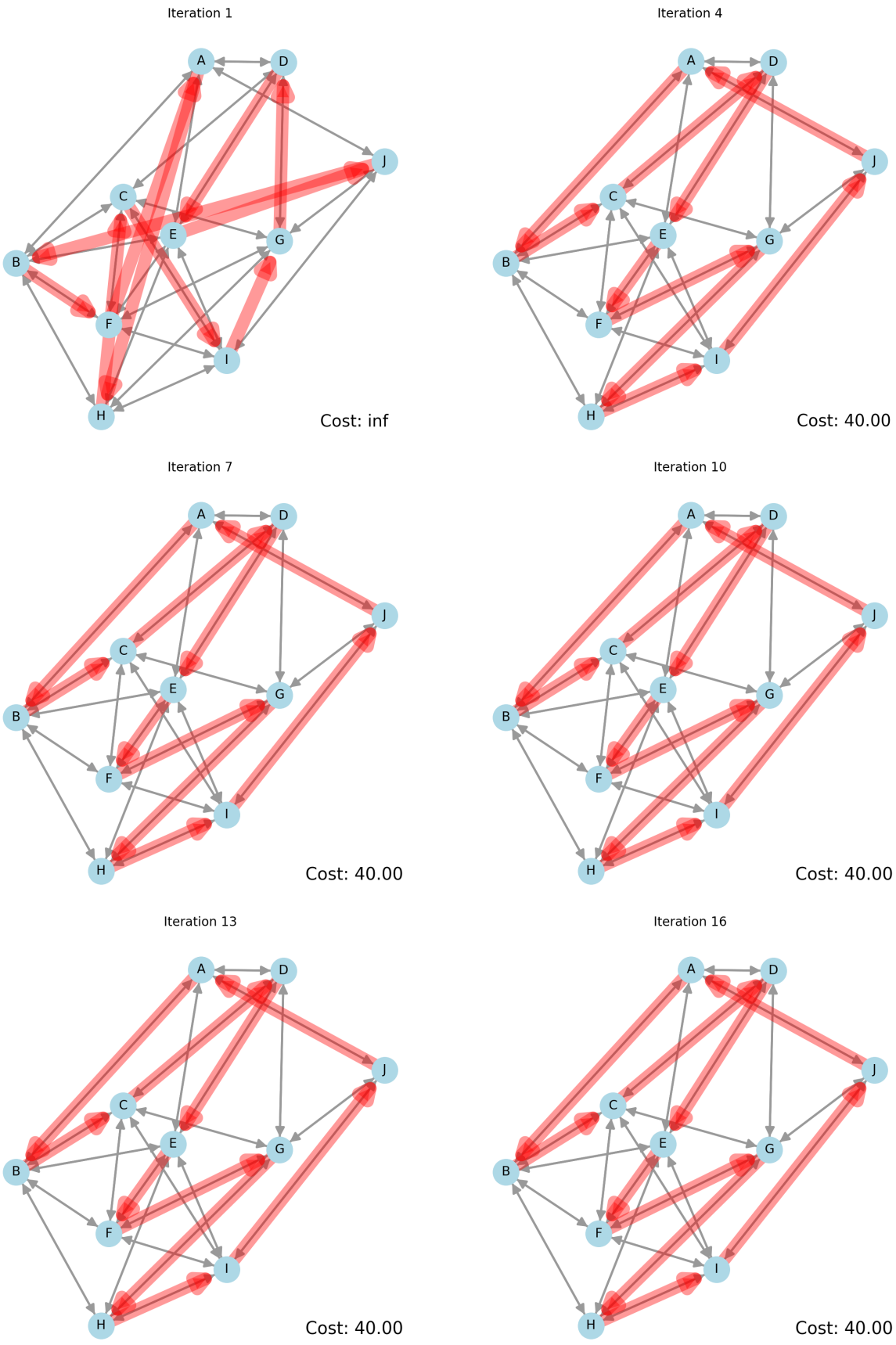

Figure 10: PSO Restart 2: iterations 1, 4, 7, 10, 13, 16

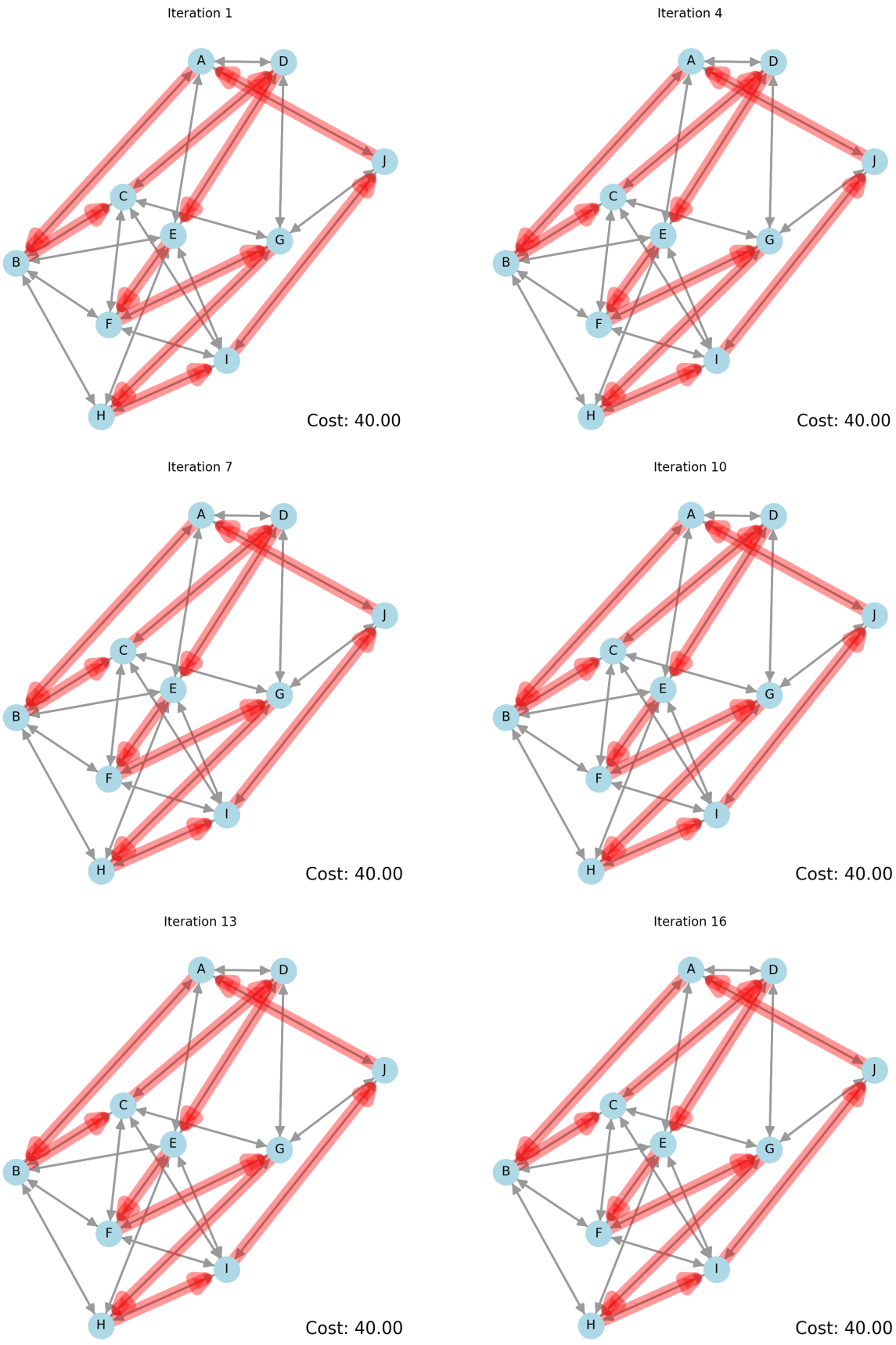

Figure 11: PSO Restart 3: iterations 1, 4, 7, 10, 13, 16

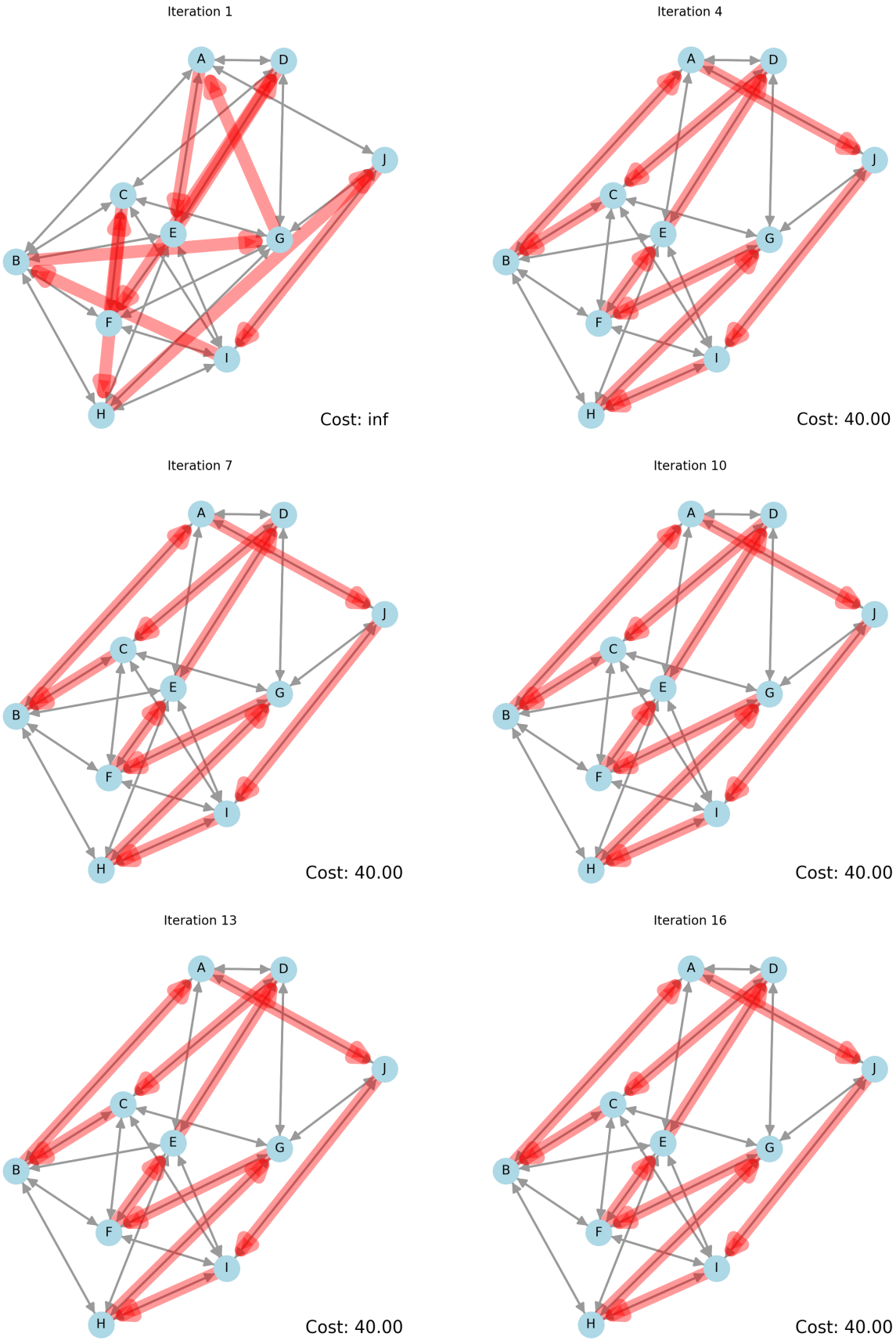

Figure 12: PSO Restart 4: iterations 1, 4, 7, 10, 13, 16

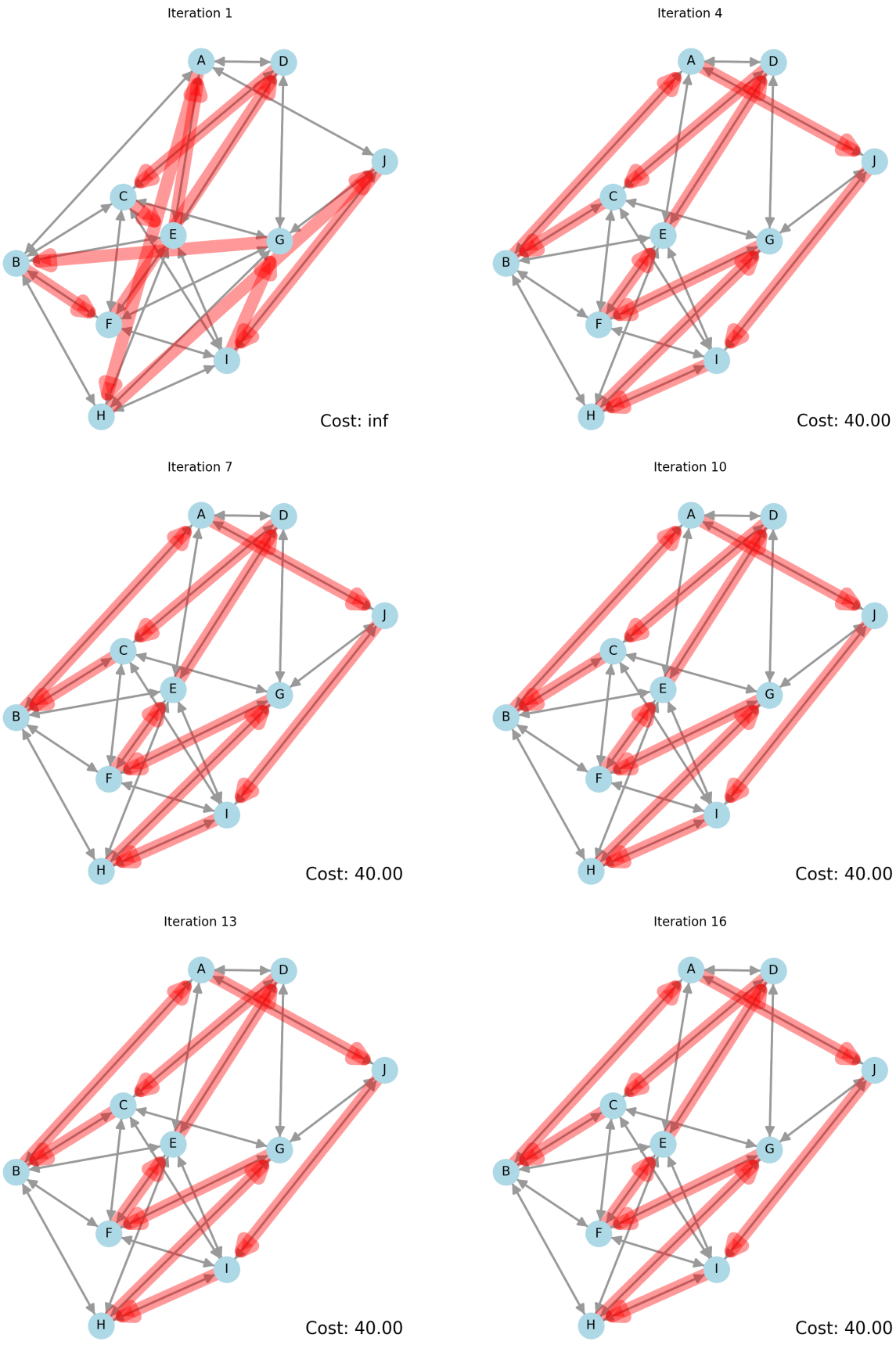

Figure 13: PSO Restart 5: iterations 1, 4, 7, 10, 13, 16

2.1.3 TSP with Common Method

The exact solution is computed with Held-Karp dynamic programming. For a subset S of nodes and endpoint $j \in S$, the recurrence is

$$DP[S, j] = \min_{k \in S \setminus \{j\}} \left(DP[S \setminus \{j\}, k] + w_{kj} \right),$$

and the optimal tour cost is $\min_j DP[V \setminus \{s\}, j] + w_{js}$, where s is the start node. This guarantees the optimal tour for the given graph.

Results: The exact method finds a tour with total cost 40. Because Held-Karp is an exact algorithm, this cost is guaranteed to be optimal for the given graph. The exact optimal tour is visualized below.

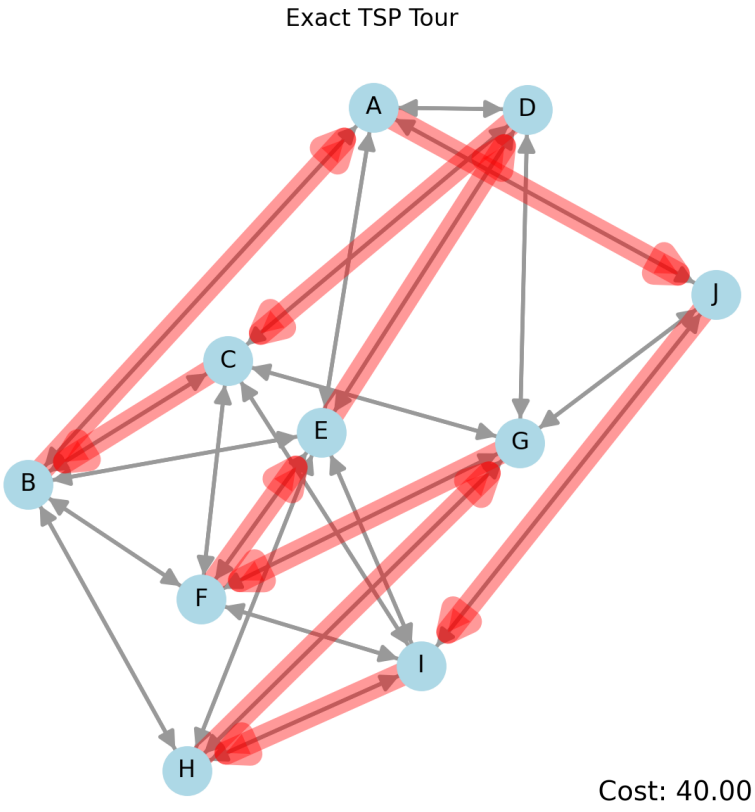

Figure 14: Exact solution tour (Held-Karp)

2.2 Vehicle Routing Problem (VRP)

The VRP extends TSP to multiple routes. In this project, there are two vehicles with fixed depots, A and H. The objective is to minimize the combined traversal cost of both routes. The constraints are: each customer node is visited exactly once by exactly one vehicle, each route starts at its own depot (A or H) and returns to that same depot, and the set of customer nodes is partitioned across the two routes without overlap.

2.2.1 VRP with ACO

The VRP variant uses two depots (A and H). Each ant constructs two routes that together cover all nodes, and the total cost is

$$L = \sum_{r \in \{1,2\}} \sum_{(u,v) \in r} w_{uv}.$$

Route construction follows the same probabilistic rule as in ACO, with pheromone updates based on the combined cost across both vehicles.

Results: The best result achieved in 200 iterations was a total cost of 46. It was not possible to reach the optimal solution here because the optimal routes include a segment going from H to G and back (as shown in the exact solution), but the ant construction rule always moves to an unvisited node to cover all nodes, so that back-and-forth segment is never selected. This can be resolved with a different construction approach than the one used in this project. Example iterations from each restart are shown below.

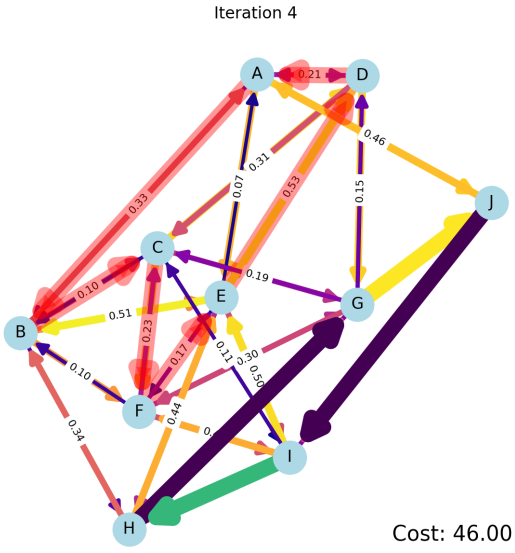

Figure 15: VRP ACO Restart 1: iteration 4

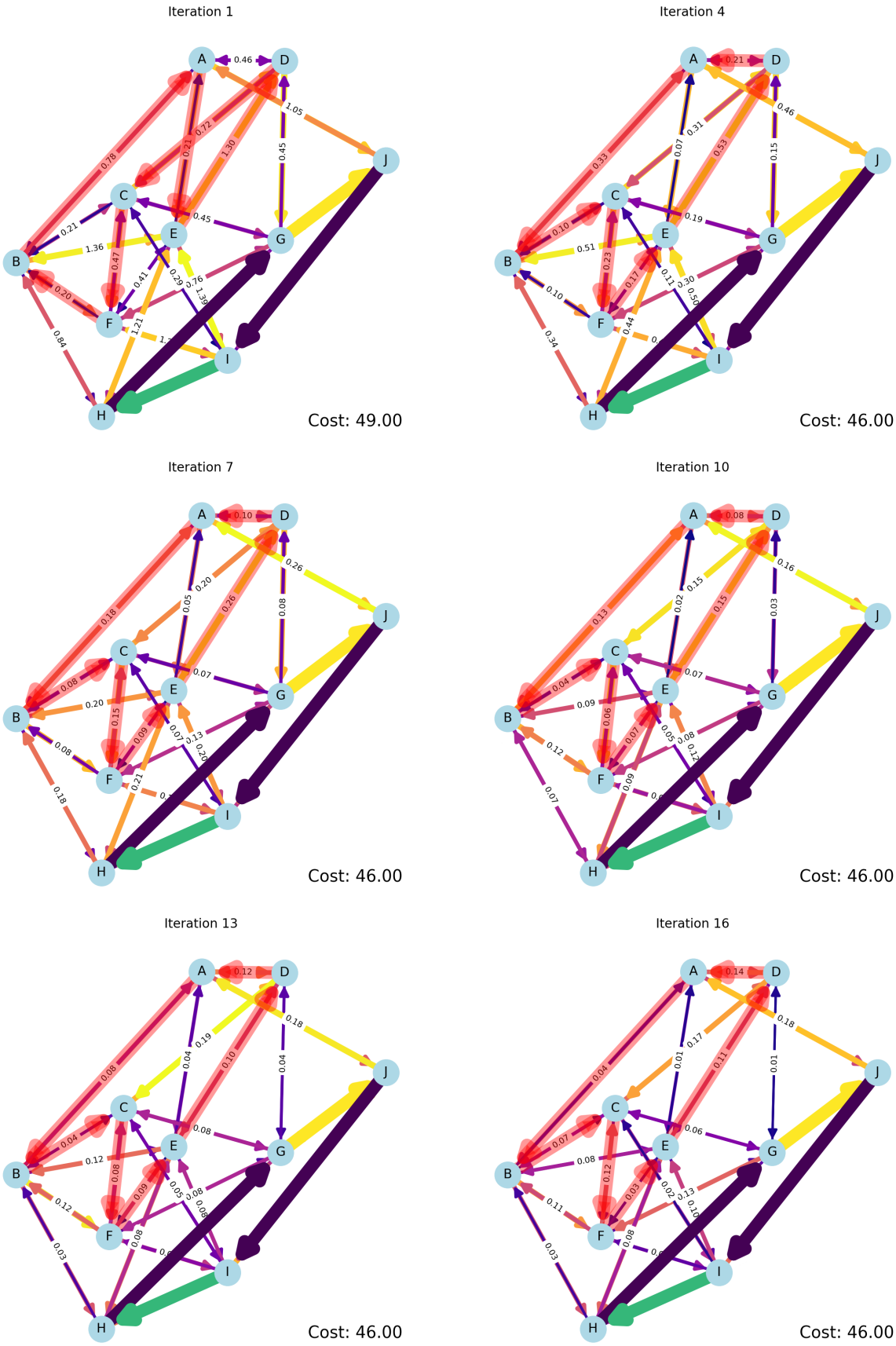

Figure 16: VRP ACO Restart 1: iterations 1, 4, 7, 10, 13, 16

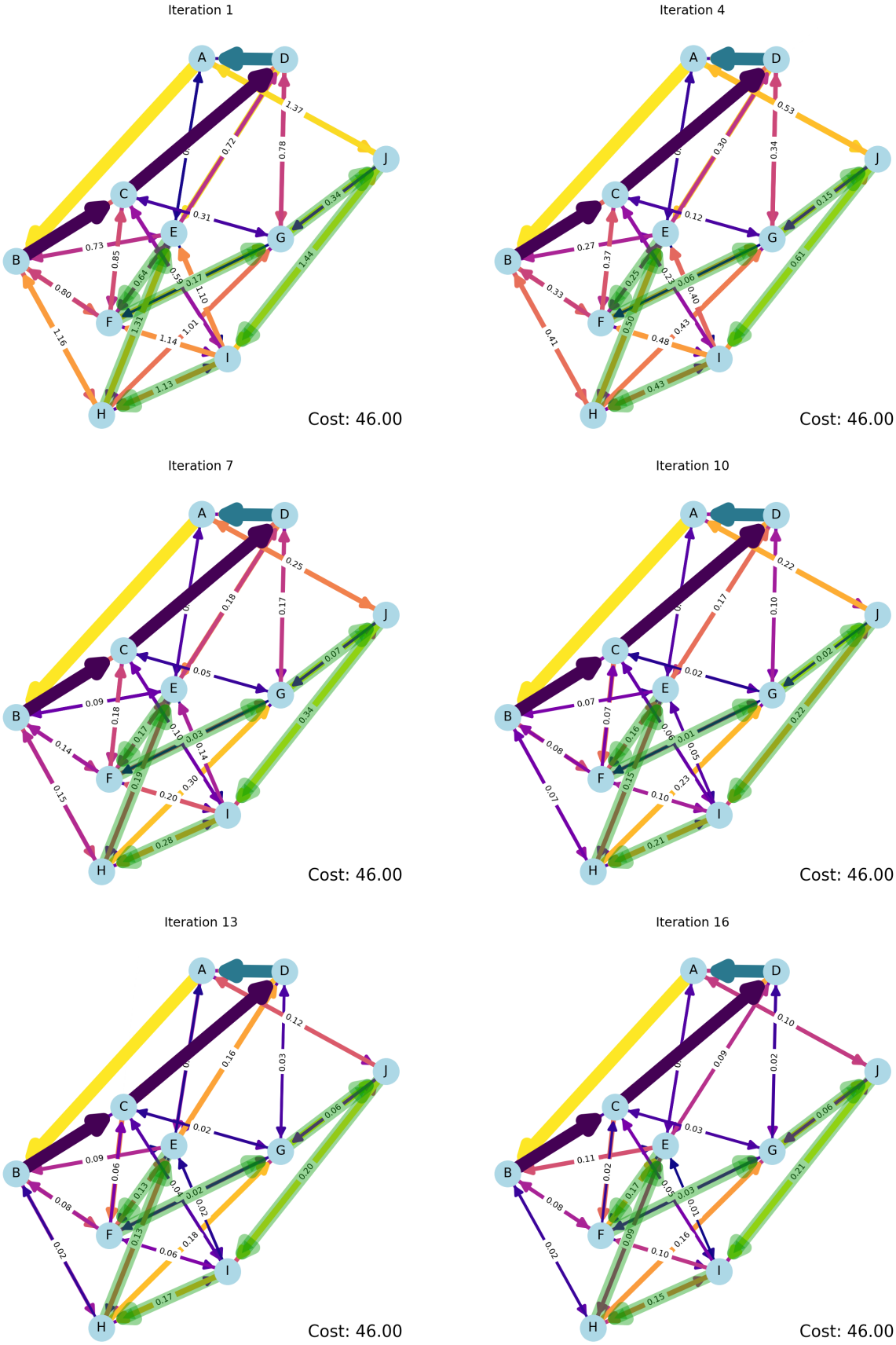

Figure 17: VRP ACO Restart 2: iterations 1, 4, 7, 10, 13, 16

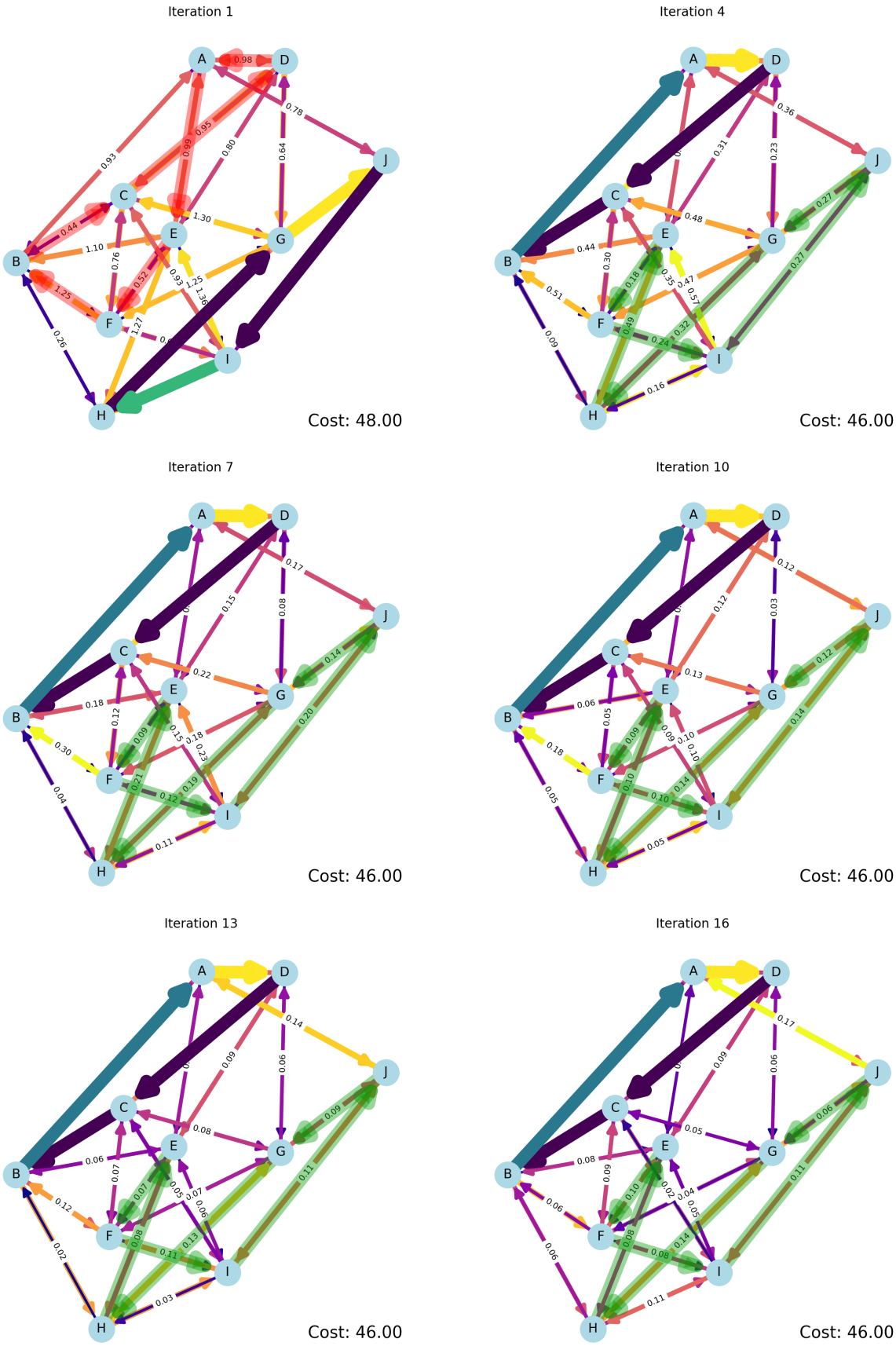

Figure 18: VRP ACO Restart 3: iterations 1, 4, 7, 10, 13, 16

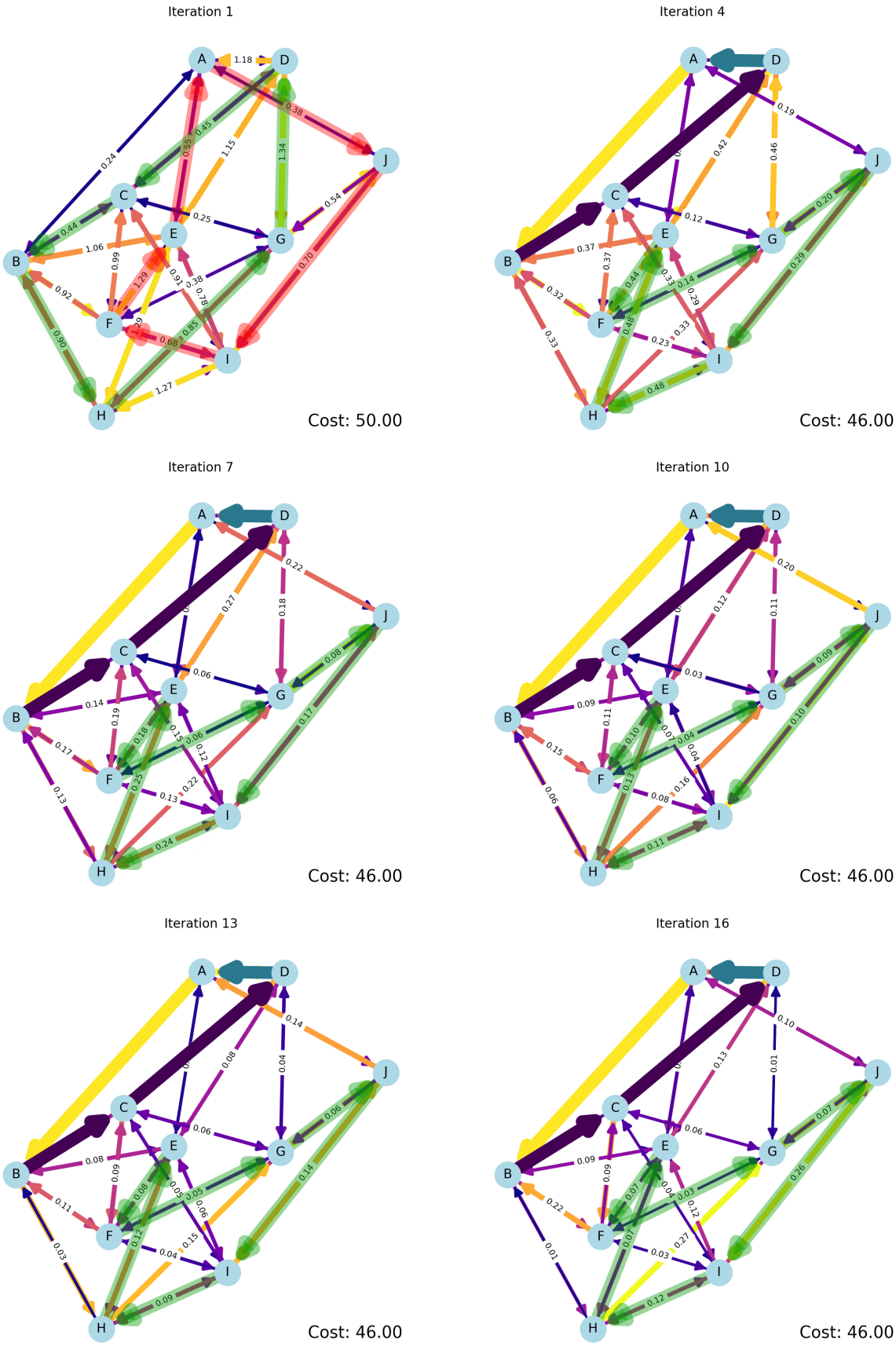

Figure 19: VRP ACO Restart 4: iterations 1, 4, 7, 10, 13, 16

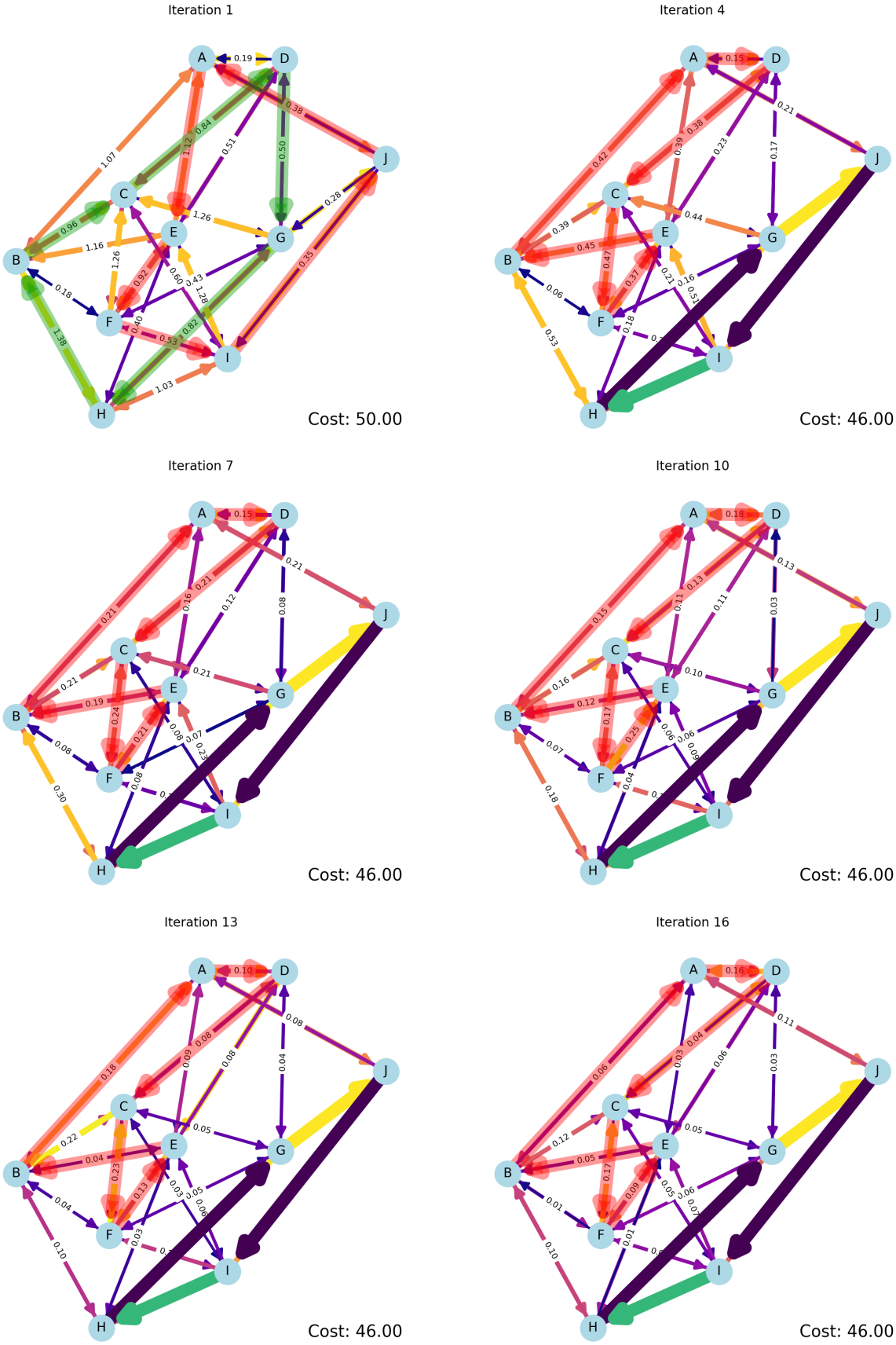

Figure 20: VRP ACO Restart 5: iterations 1, 4, 7, 10, 13, 16

2.2.2 VRP with PSO

For VRP with PSO, a particle is represented by a permutation of nodes and a split index that divides the permutation into two routes starting at A and H. The total cost is the sum of both route costs, and swap-based updates move the permutation toward pbest and gbest while the split index is mutated to explore different allocations. Restarts are used to improve robustness.

Results: The best solution found was cost 40, reached within 200 iterations. It appeared at iteration 121 in restart 1 and iteration 188 in restart 2, which is noticeably slower convergence than the TSP case. In some restarts (e.g., restart 1 and restart 3), the swarm initially could not find a feasible solution, so the cost was reported as ∞ until a valid split and routes were discovered. Example iterations from each restart are shown below.

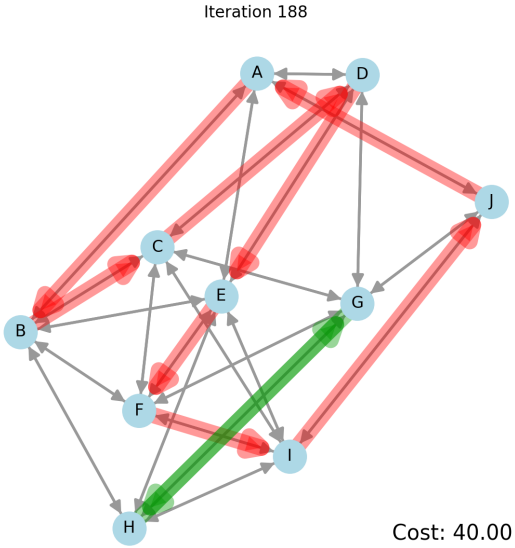

Figure 21: VRP PSO Restart 2: iteration 188

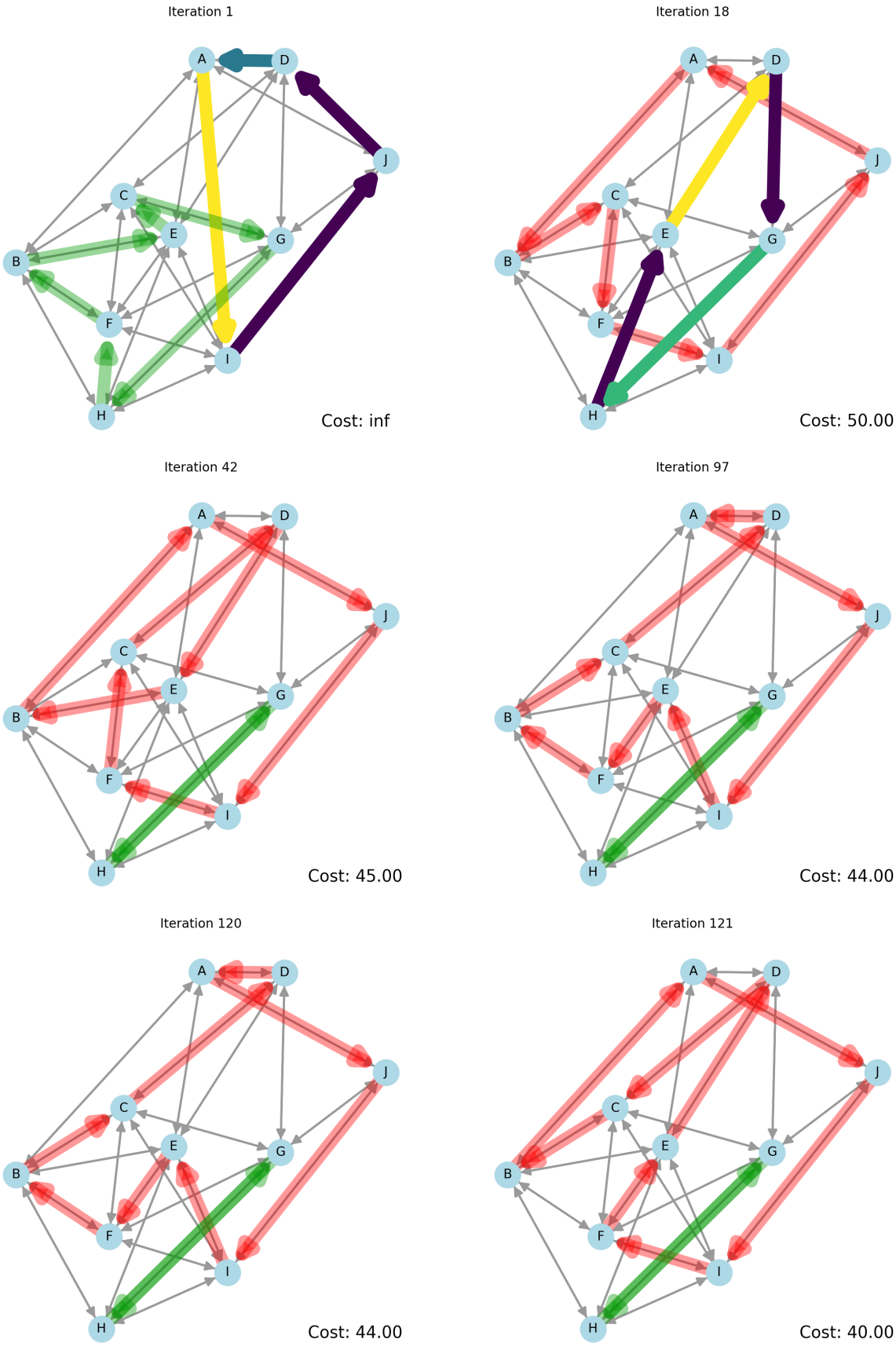

Figure 22: VRP PSO Restart 1: iterations where the cost changed

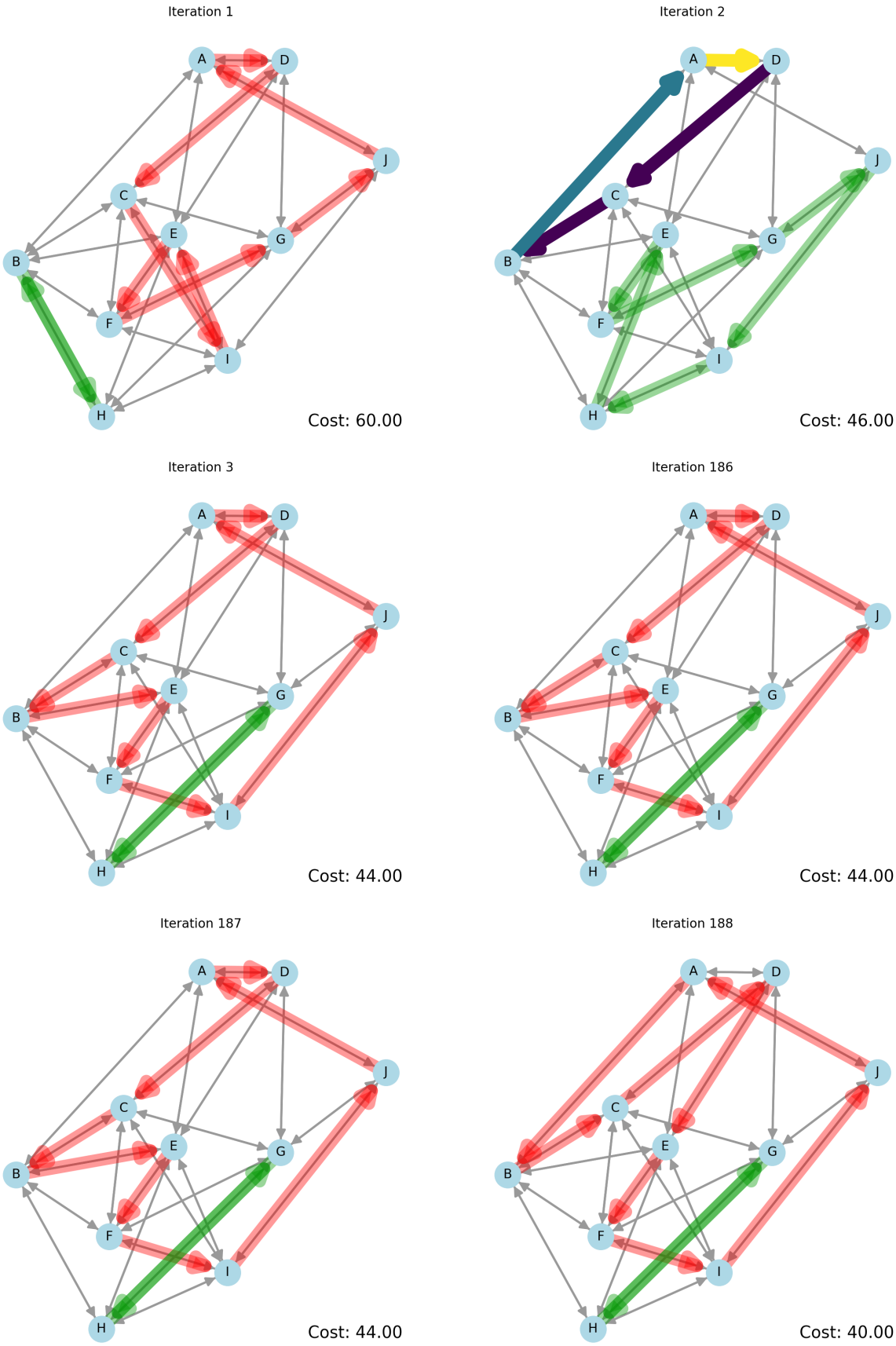

Figure 23: VRP PSO Restart 2: iterations where the cost changed

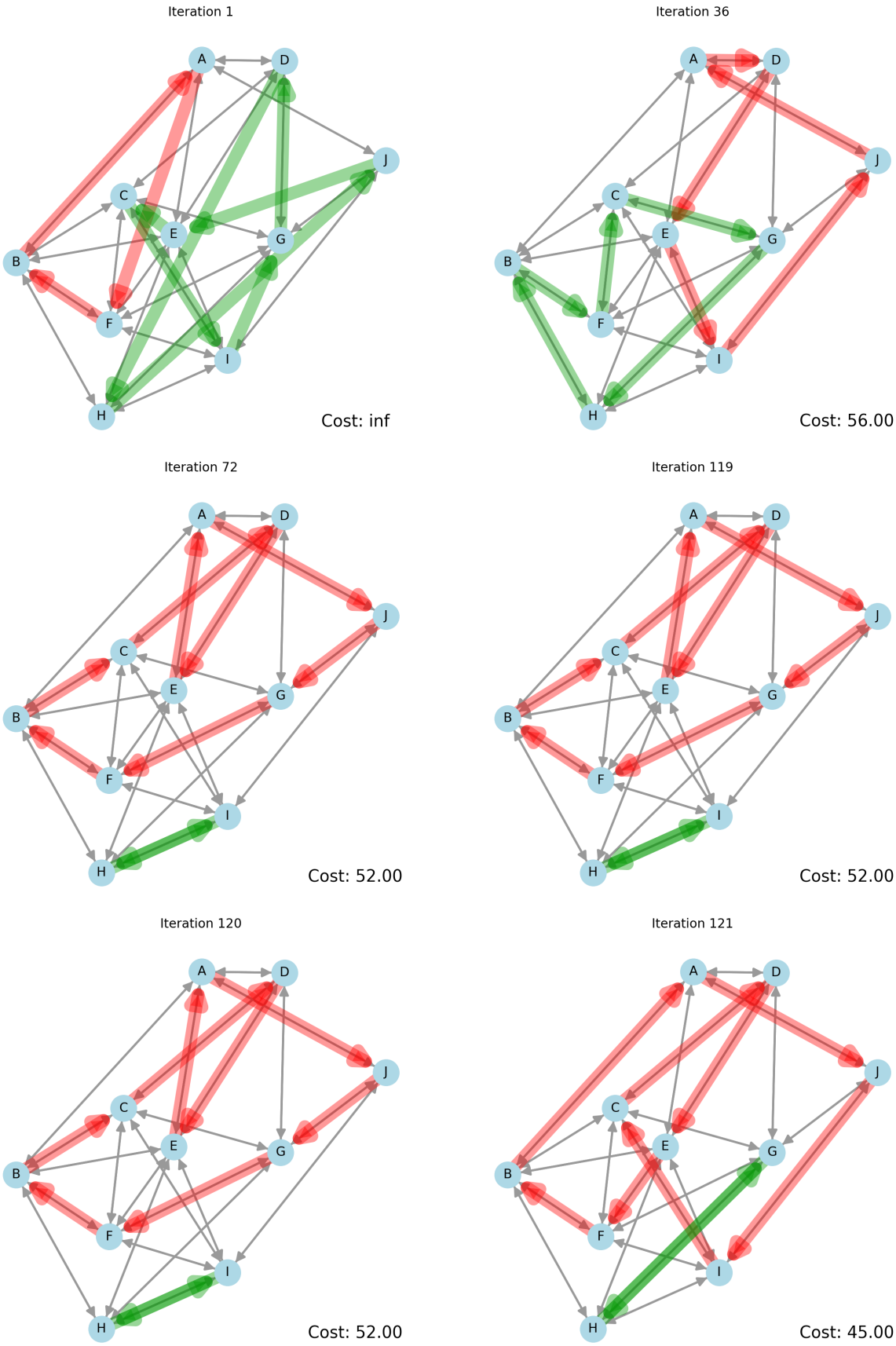

Figure 24: VRP PSO Restart 3: iterations where the cost changed

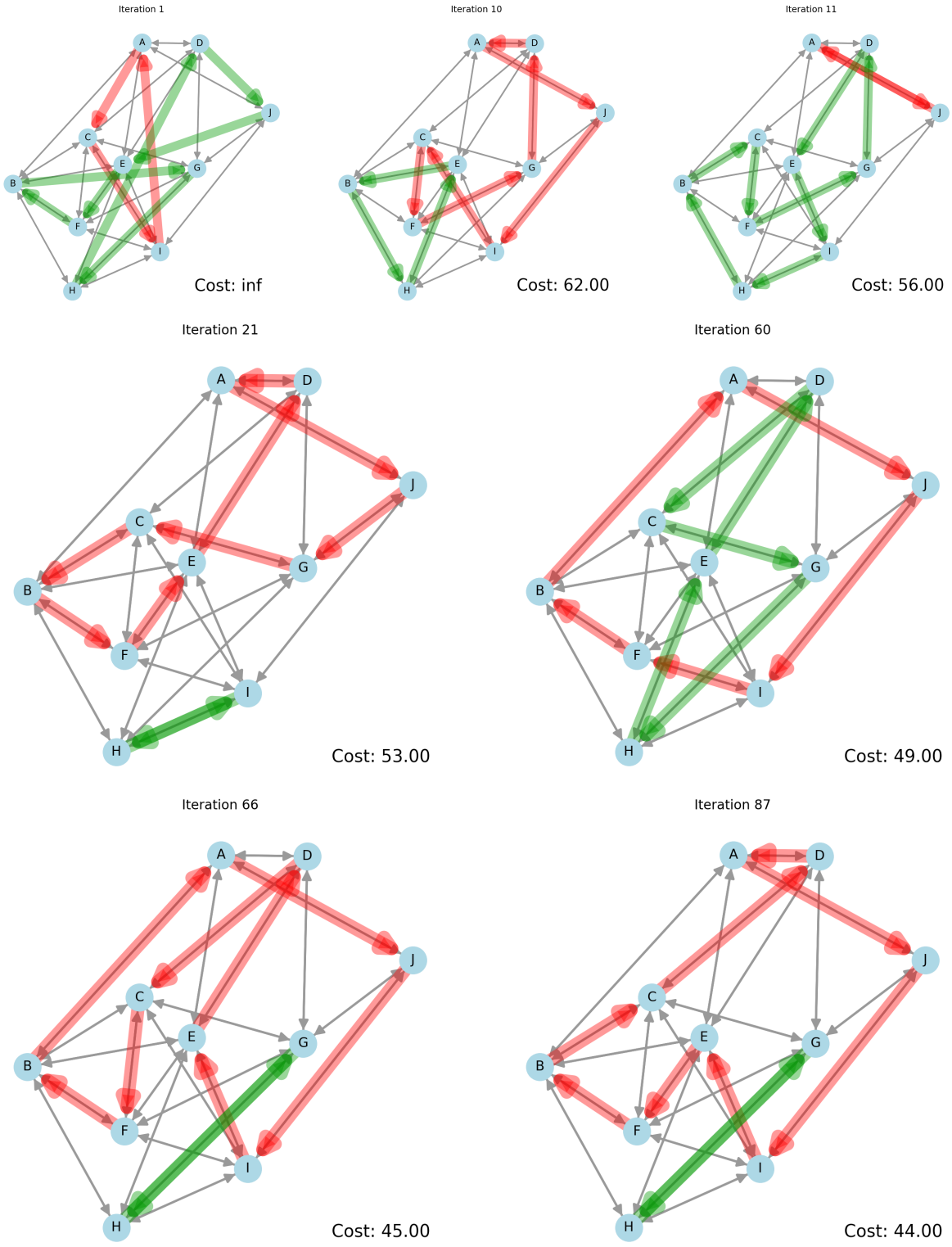

Figure 25: VRP PSO Restart 4: iterations where the cost changed

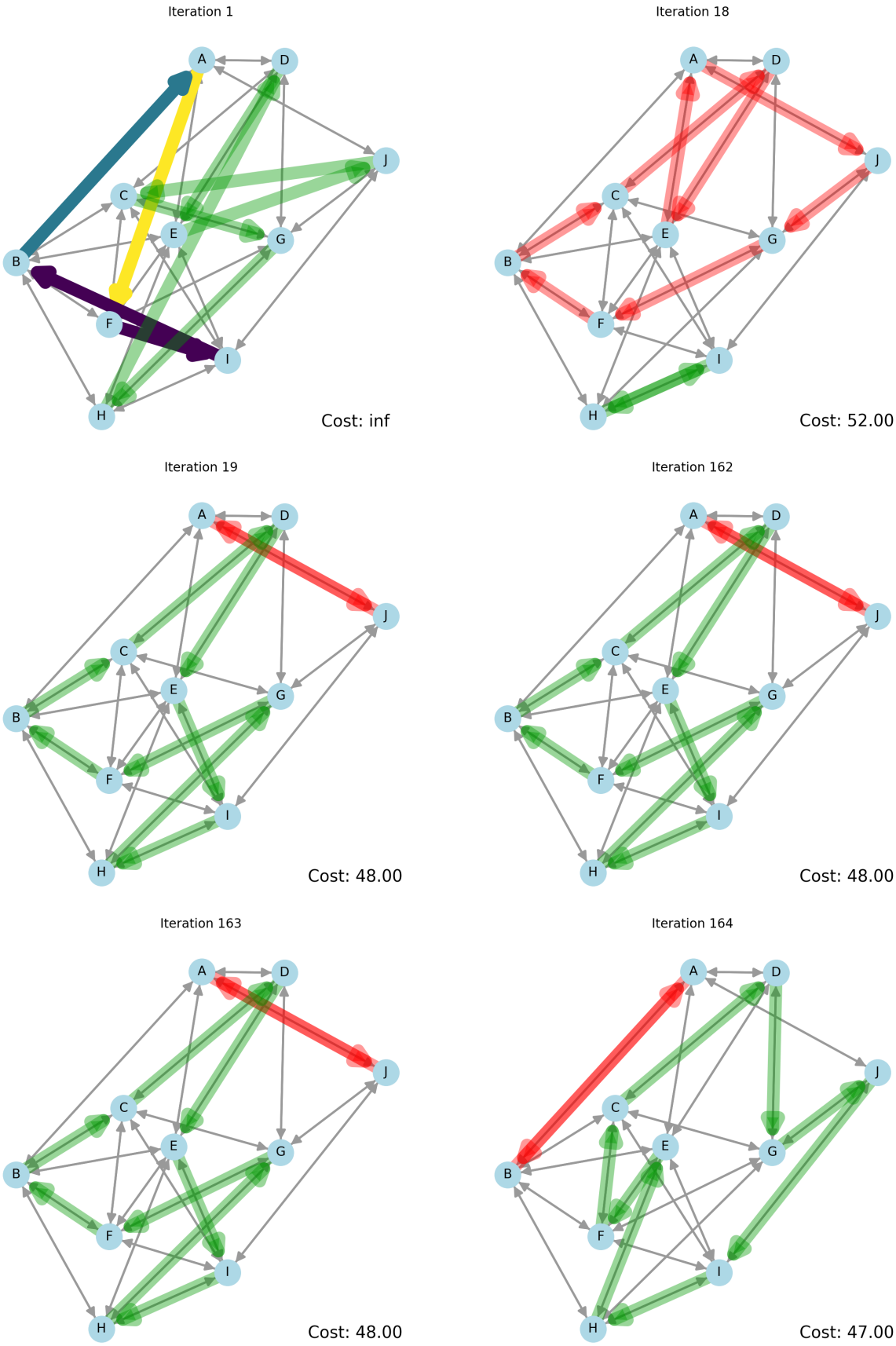

Figure 26: VRP PSO Restart 5: iterations where the cost changed

2.2.3 VRP with Common Method

The exact VRP solver enumerates all partitions of nodes between the two depots and solves each subset with Held-Karp, then selects the partition with the smallest combined cost. This yields the optimal two-route solution for the given graph at the expense of exponential time.

Results: The exact method finds a solution with total cost 40. Because this is an exact solver, this cost is guaranteed to be optimal (or one of the optimal solutions if there are ties). The exact optimal routes are visualized below.

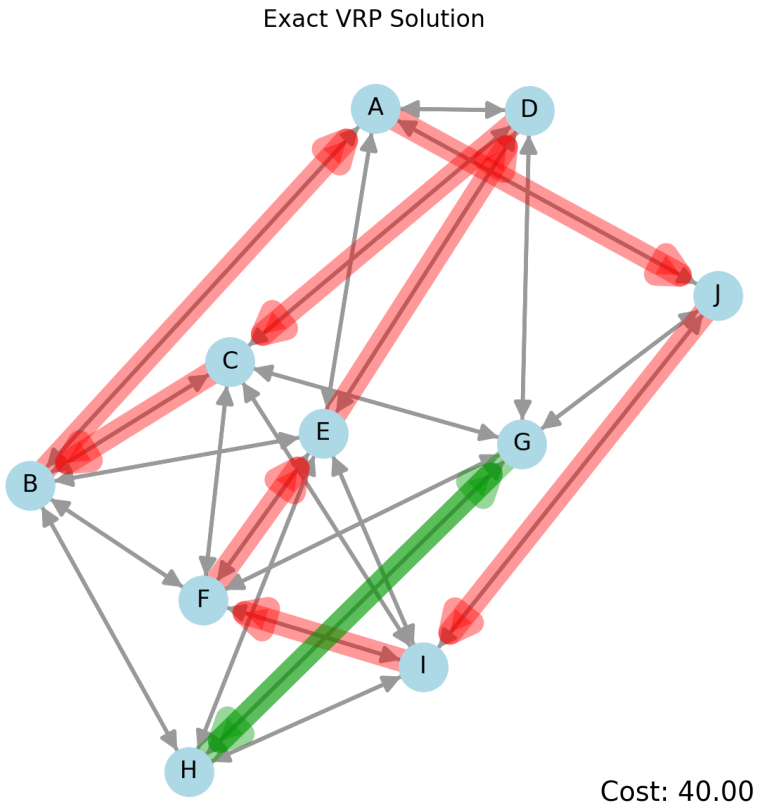

Figure 27: Exact VRP solution (two depots)

3 Comparison of Methods

ACO: Flexible and easy to parallelize, often converges quickly to good solutions on larger graphs. In this project, the ACO implementation was not sufficient to reach the optimal VRP solution because the construction rule could not represent the required back-and-forth segment.

PSO: Effective on permutation spaces with swap-based updates and local search; can be competitive in solution quality, but may require many iterations to converge. Here, PSO needed substantially more iterations to reach the optimal VRP solution, and some runs did not reach it within the iteration limit.

Exact method: Guarantees optimality and serves as a benchmark on small instances, but runtime grows exponentially, making it impractical as node counts increase.

ACO constructs routes step by step using probabilistic decisions guided by pheromone trails and heuristic edge desirability, reinforcing edges that appear in good tours. PSO instead maintains a swarm of full candidate solutions (permutations) and updates them via swap-based velocities toward each particle's best and the global best, moving through solution space rather than building routes incrementally.

4 Discussion

These algorithms are useful in routing and scheduling settings where decisions must be made quickly or repeated many times, such as delivery planning, warehouse picking, ride-sharing dispatch, technician routing, and production logistics. ACO and PSO can provide high-quality tours or multi-route plans without exhaustively searching the solution space, which makes them practical for larger graphs or time-sensitive applications.

Compared to exact methods, metaheuristics trade guaranteed optimality for speed and scalability. Their main advantages are lower computational cost, faster convergence to good solutions, and the ability to run efficiently in parallel across multiple particles or ants and multiple restarts. This makes them well-suited for real-world instances where exact solvers become infeasible as the number of nodes grows. Exact methods remain valuable as benchmarks on small instances or when optimality is required and runtime is acceptable.

Larger graph (ACO): For a larger 18-node complete graph, 5 restarts with 200 iterations were run. The best tour found was at restart 3, iteration 108, shown below. The algorithm did not reach the optimal solution within 200 iterations, so this iteration budget was not enough for the larger instance.

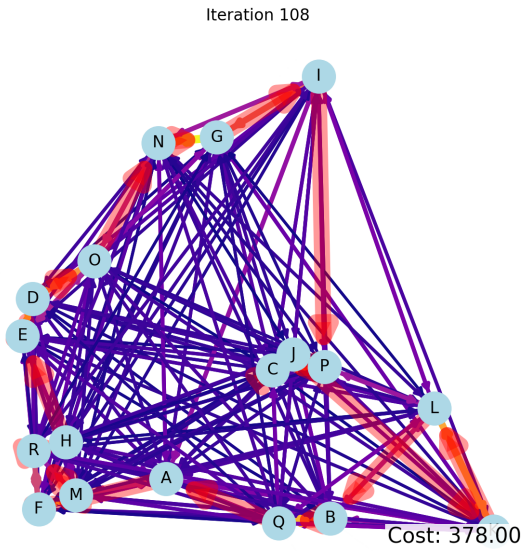

Figure 28: ACO (larger graph): Restart 3, iteration 108

Figure 29: ACO simulation (TSP). View: Google Drive link

Figure 30: PSO simulation (TSP). View: Google Drive link

Figure 31: ACO simulation (VRP). View: Google Drive link

Figure 32: PSO simulation (VRP). View: Google Drive link

The ACO simulation TSP (Figure 28) has these iterations for instance. After iteration 7, cost is reduced to 44 from 45 and the path has changed.

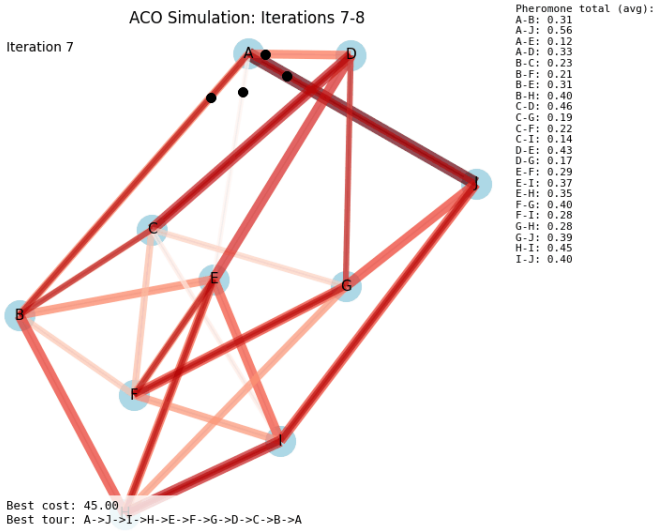

Figure 33: ACO TSP simulation: iterations 7 and 8

After making the simulations and seeing that the ants moving can be represented visually in an animation, I came up with an idea of creating a mobile game. The ants find the shortest path from the origin to the destination point that the user chooses. I am interested in mobile coding, so this idea appealed to me. You can find the example code I did for this game in `code10.py` in the GitHub repository where there are all the codes of the outputs of all problems and method solutions: https://github.com/Emirhan777/TSP_VRP_Using_ACO_PSO.

5 Conclusions

ACO and PSO are practical for TSE-style logistics problems because they can produce good tours quickly, while the exact method guarantees optimality for small instances. In the experiments, ACO and PSO matched the exact solution on small graphs and converged reliably, but for some larger graph combinations they did not always avoid non-optimal tours. This highlights the trade-off between exactness and scalability in logistics optimization.

These methods and algorithms have their own advantages and disadvantages. As the network grows, the advantage of swarm algorithms grows as well. They are computationally less demanding and can manage larger problems with shorter time and fewer computational operations compared to exact methods. Based on the experiments, I would use ACO for the TSE and PSO for the VRP.

In conclusion, swarm methods provide a practical balance between solution quality and runtime for realistic logistics sizes, while exact methods remain valuable for small instances and validation. This study shows the advantages and disadvantages of each approach on the studied problems.

6 References

References

[1] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, *The Traveling Salesman Problem*, Wiley, 1985.

[2] P. Toth and D. Vigo (eds.), *The Vehicle Routing Problem*, SIAM, 2002.

[3] M. Dorigo and L. M. Gambardella, "Ant colonies for the traveling salesman problem," *BioSystems*, vol. 43, no. 2, pp. 73–81, 1997.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[5] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 10, no. 1, pp. 196–210, 1962.