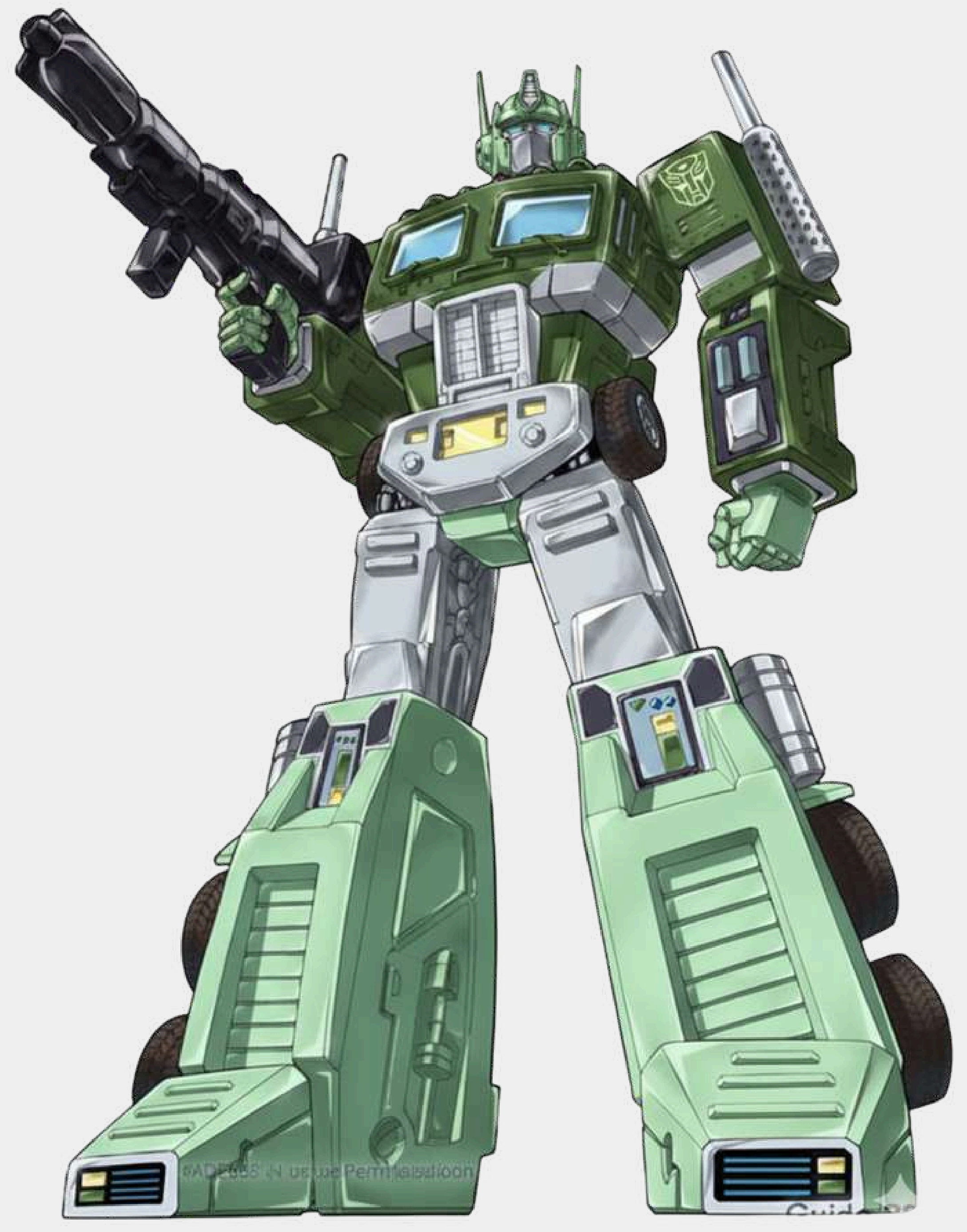
Team Ne3Na3

N3N3 Optimus Price

Ahmed Samir
Mohammed Hany
Mohammad Emad
Loay Medhat

Optimization with AI



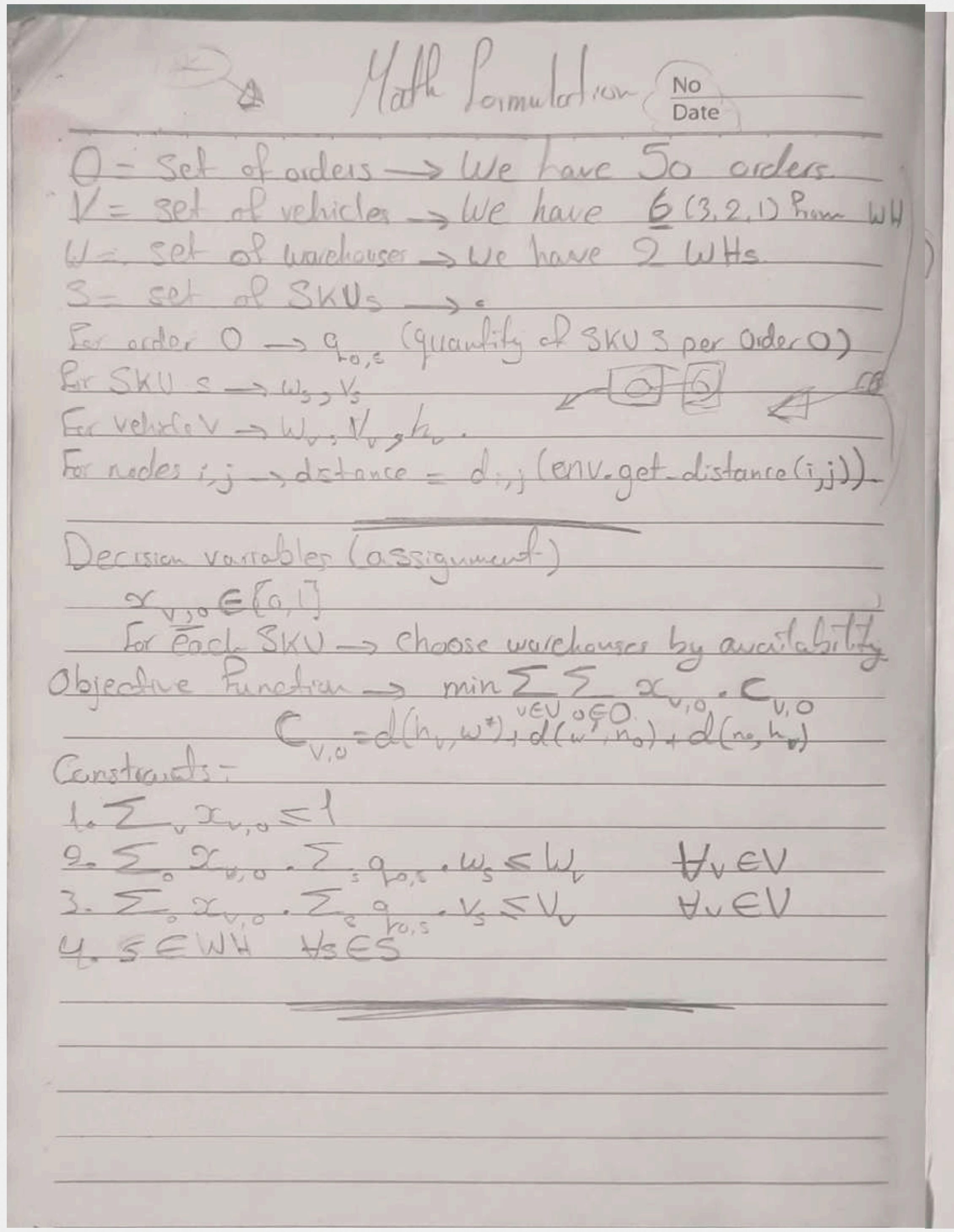
Problem Analysis

Hackathon evaluation function:

Scenario Score=Your Scenario Cost + Benchmark Solver Cost × (100 – Your Fulfillment %)

Then, it came down to us. It was all about priorities. Fulfillment isn't just a constraint, but a business standard we must uphold. Thus, we had our objective: fulfillment-first delivery (100% orders if possible). After that, minimize cost = fixed vehicle costs (dominant) + distance-driven operating costs. We thought things could be solved with the Hungarian method at first, making use of what we learned in college at least.

Samples of our thought processes



Math Formulation

No

Date

O = set of orders → We have 50 orders

V = set of vehicles → We have 6 (3, 2, 1) from WH

W = set of warehouses → We have 2 WHs.

S = set of SKUs → .

For order O → $q_{o,s}$ (quantity of SKU s per order O)

For SKU s → w_s, v_s

For vehicle v → W_v, V_v, h_v.

For nodes i,j → distance = $d_{i,j}$ (env.get_distance(i,j)).

Decision variables (assignment)

$x_{v,o} \in \{0,1\}$

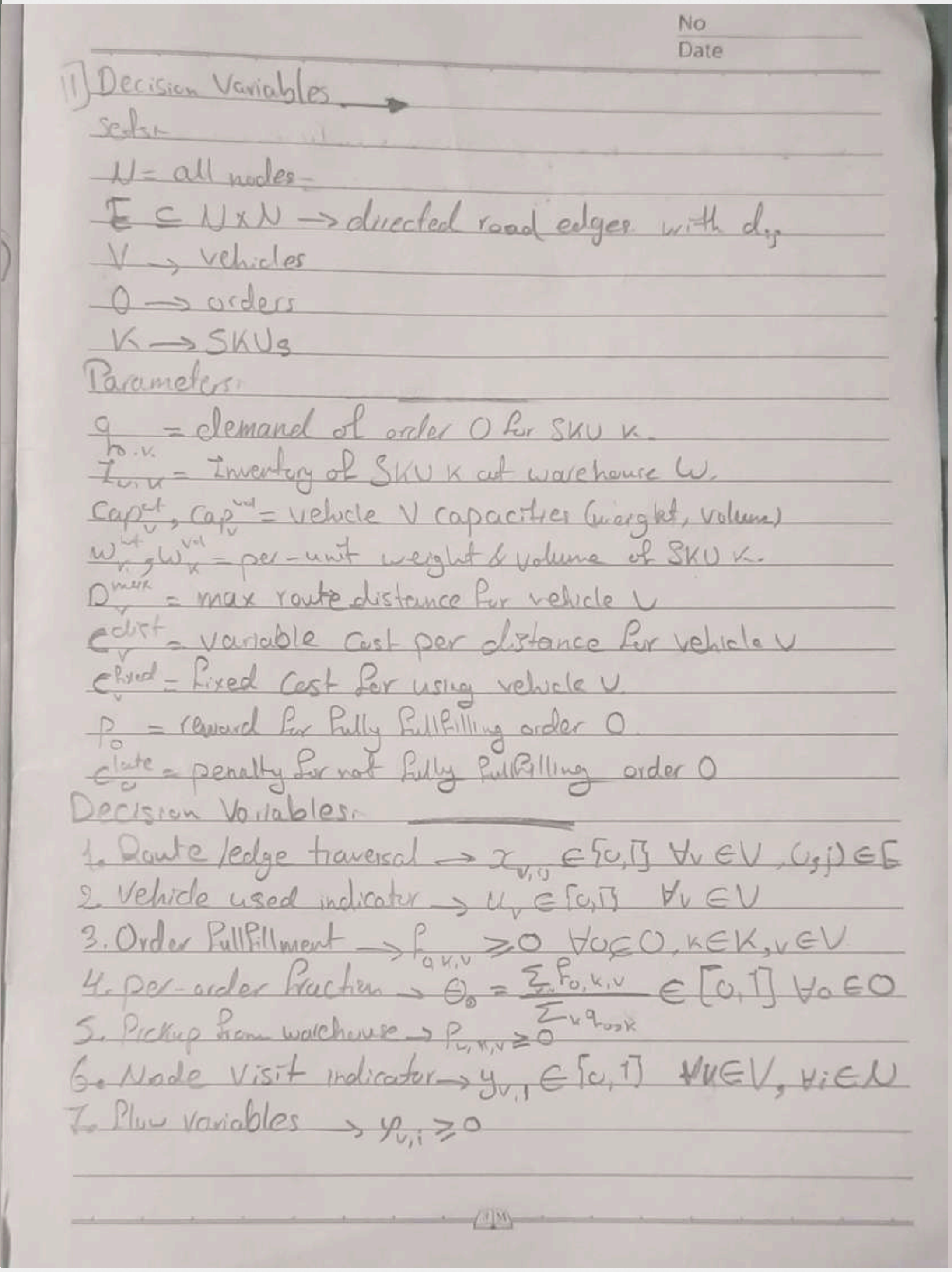
For each SKU → choose warehouses by availability

Objective Function → $\min \sum_{v \in V} \sum_{o \in O} x_{v,o} \cdot C_{v,o}$

$C_{v,o} = d(h_v, w^*) + d(w^*, n_o) + d(n_o, h_v)$

Constraints:

1. $\sum_v x_{v,o} \le 1$
2. $\sum_o x_{v,o} \cdot \sum_s q_{o,s} \cdot w_s \le W_v \quad \forall v \in V$
3. $\sum_o x_{v,o} \cdot \sum_s q_{o,s} \cdot v_s \le V_v \quad \forall v \in V$
4. $s \in WH \quad \forall s \in S$



No

Date

1) Decision Variables →

Sets:

N = all nodes.

$E \subseteq N \times N$ → directed road edges with d_{ij}

V → vehicles

O → orders

K → SKUs

Parameters:

$q_{o,k}$ = demand of order O for SKU k.

$I_{w,k}$ = Inventory of SKU k at warehouse w.

cap_v^{wt}, cap_v^{vol} = vehicle v capacities (weight, volume)

w_k^{wt}, w_k^{vol} = per-unit weight & volume of SKU k.

D_v^{max} = max route distance for vehicle v

c_v^{dist} = variable cost per distance for vehicle v

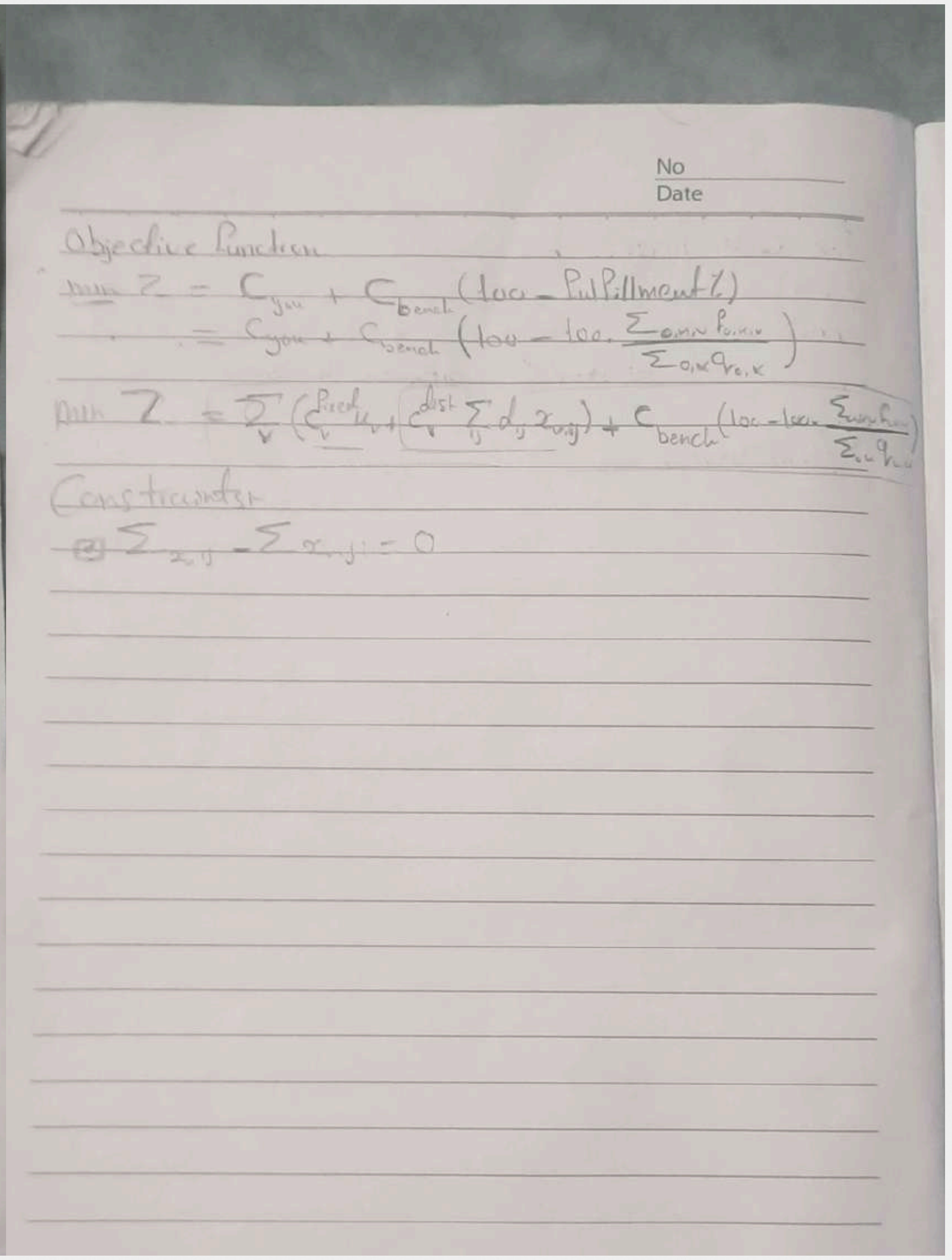
c_v^{fixed} = fixed cost for using vehicle v.

p_o = reward for fully fullfilling order O.

c_o^{late} = penalty for not fully fullfilling order O

Decision Variables:

1. Route/edge traversal → $x_{v,ij} \in \{0,1\} \ \forall v \in V, (i,j) \in E$
2. Vehicle used indicator → $u_v \in \{0,1\} \ \forall v \in V$
3. Order fullfillment → $f_{o,k,v} \ge 0 \ \forall o \in O, k \in K, v \in V$
4. per-order fraction → $\theta_o = \frac{\sum f_{o,k,v}}{\sum_k q_{o,k}} \in [0,1] \ \forall o \in O$
5. Pickup from warehouse → $p_{w,k,v} \ge 0$
6. Node visit indicator → $y_{v,i} \in \{0,1\} \ \forall v \in V, \forall i \in N$
7. Flow variables → $\varphi_{v,i} \ge 0$



No

Date

Objective Function

$\min Z = C_{you} + C_{bench}(100 - \text{Fulfillment}\%)$

$= C_{you} + C_{bench}\left(100 - 100 \cdot \frac{\sum_{o,k,v} f_{o,k,v}}{\sum_{o,k} q_{o,k}}\right)$

$\min Z = \sum_v \left(c_v^{fixed} u_v + c_v^{dist} \sum_{ij} d_{ij} x_{v,ij}\right) + c_{bench}\left(100 - 100 \cdot \frac{\sum_{o,k,v} f_{o,k,v}}{\sum_{o,k} q_{o,k}}\right)$

Constraints:

① $\sum x_{v,ij} - \sum x_{v,ji} = 0$

Market Segmentation

Beltone new work as a rating agency faces fixed costs (platforms, apps, staff, integrations) and variable costs (travel, per-visit pay, surge inspections). Physical attendance is needed for new-issuer checks, collateral/asset sighting, major discrepancies/fraud, and some ESG/regulatory verifications; routine surveillance can be remote. Adopt remote-first, risk-triggered on-site visits to cut variable costs while keeping rating integrity.

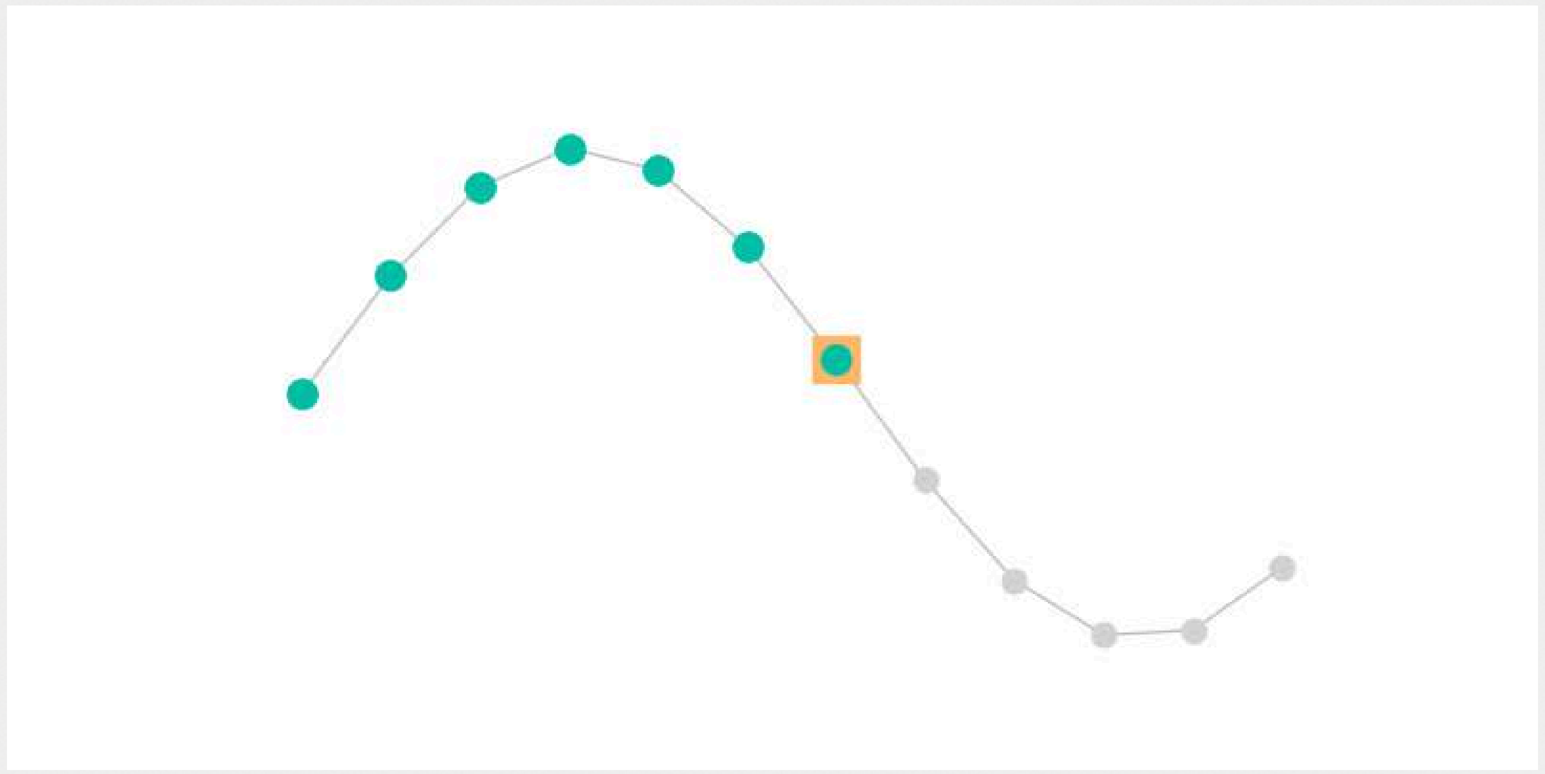
2030 VISION OF EGYPT

But, this is an AI hackathon, right?

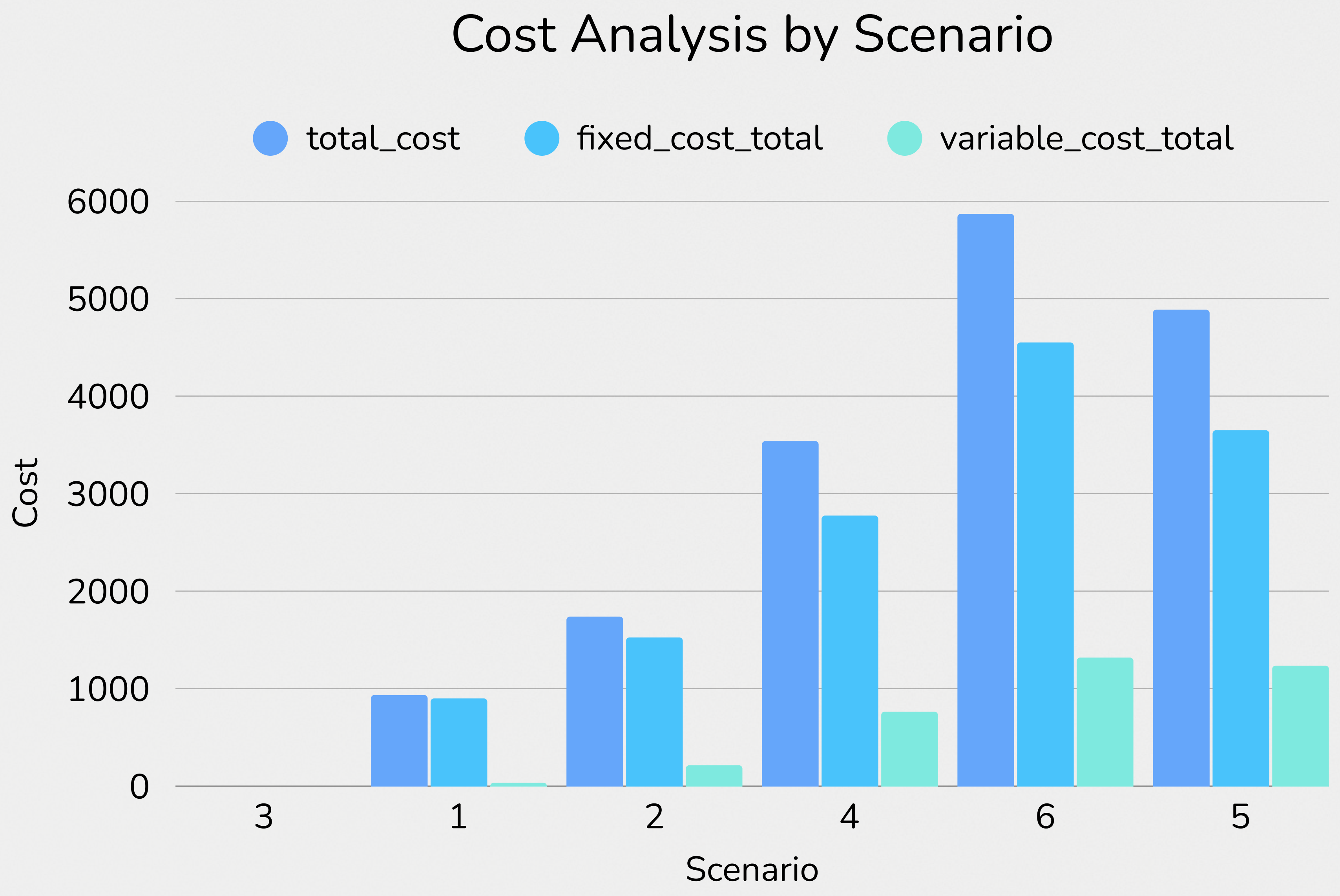
Gotta do some AI

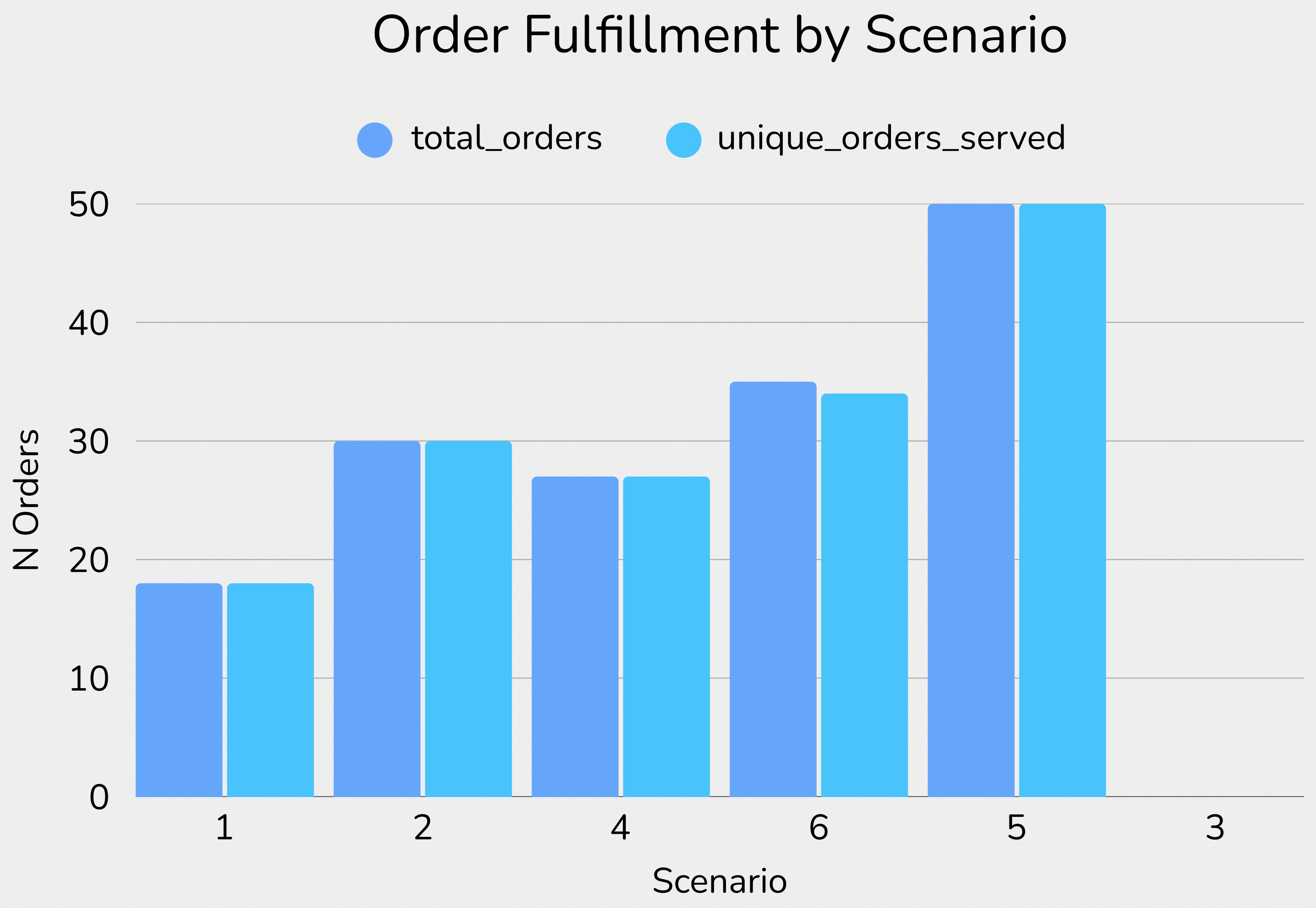
Reinforcement Learning

It is all about linear algebra, matrices, and states, so that means we can implement it from scratch with numpy. The idea is that we create a simulation, an environment, where an agent learns how to allocate the vehicles to the right warehouses to fulfill the orders. The agent keeps improving through episodes and iterations by receiving penalties and rewards based on the objective function, which is the percentage of orders fulfilled using a certain number of vehicles.



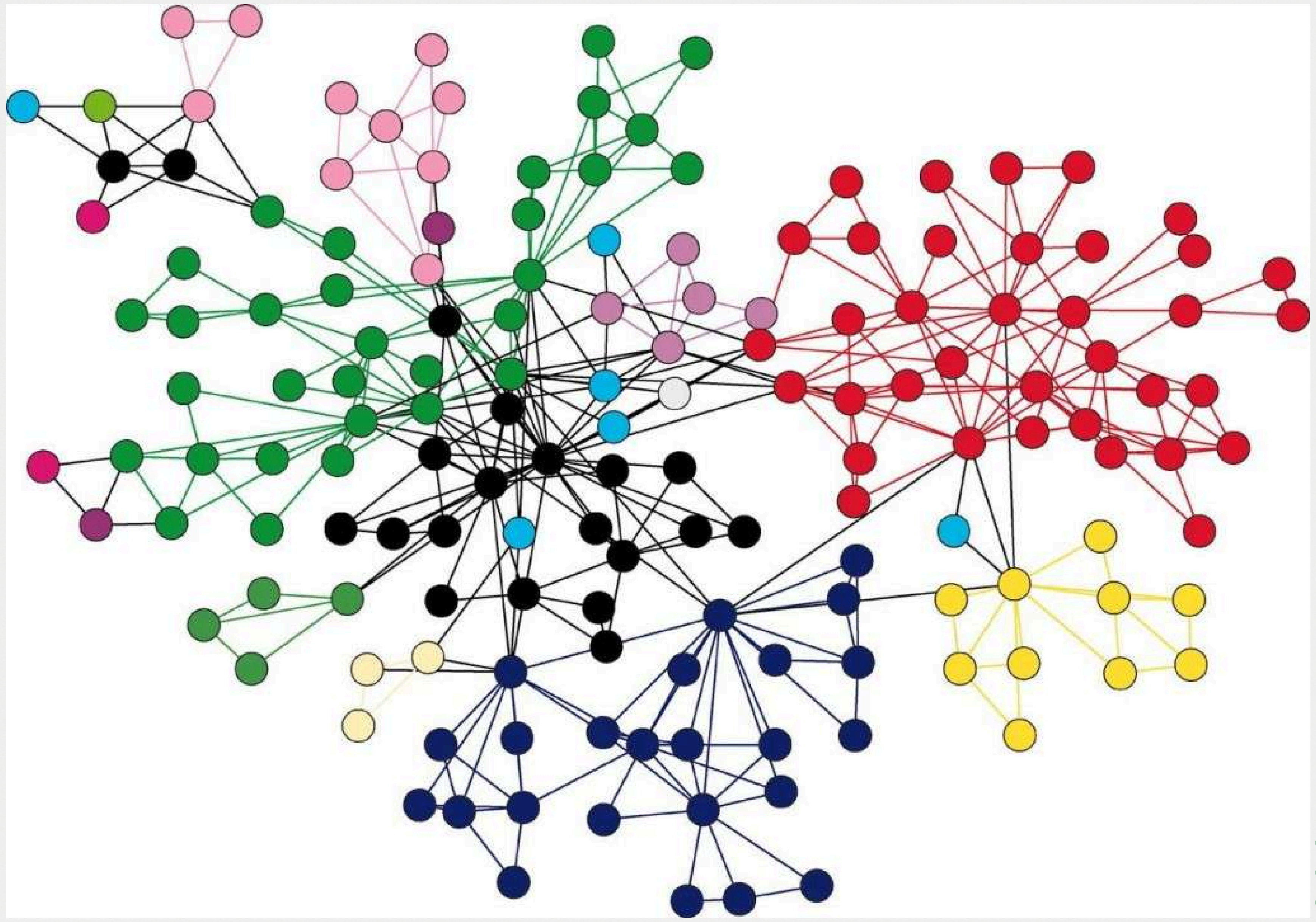
Reinforcement Learning Results



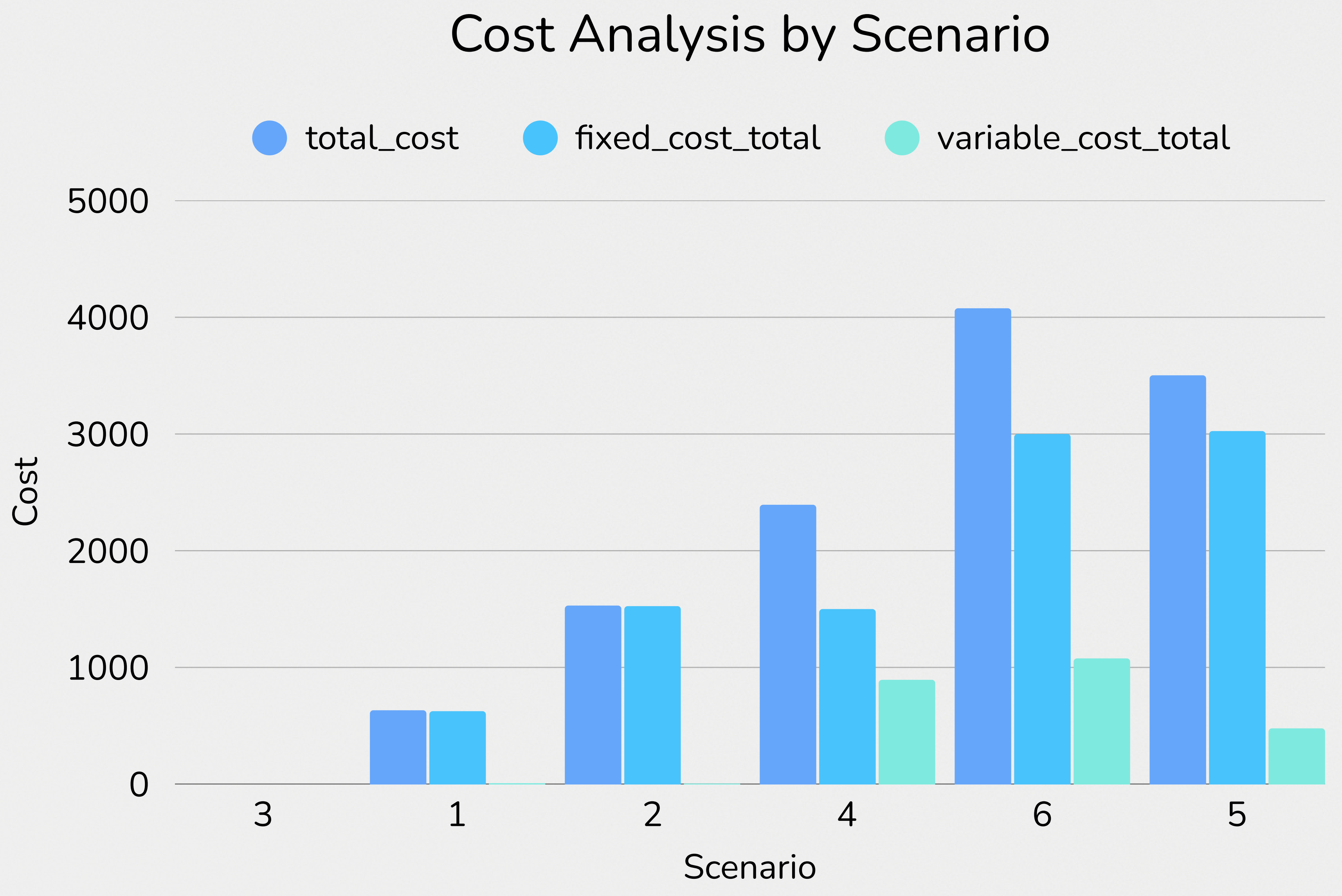


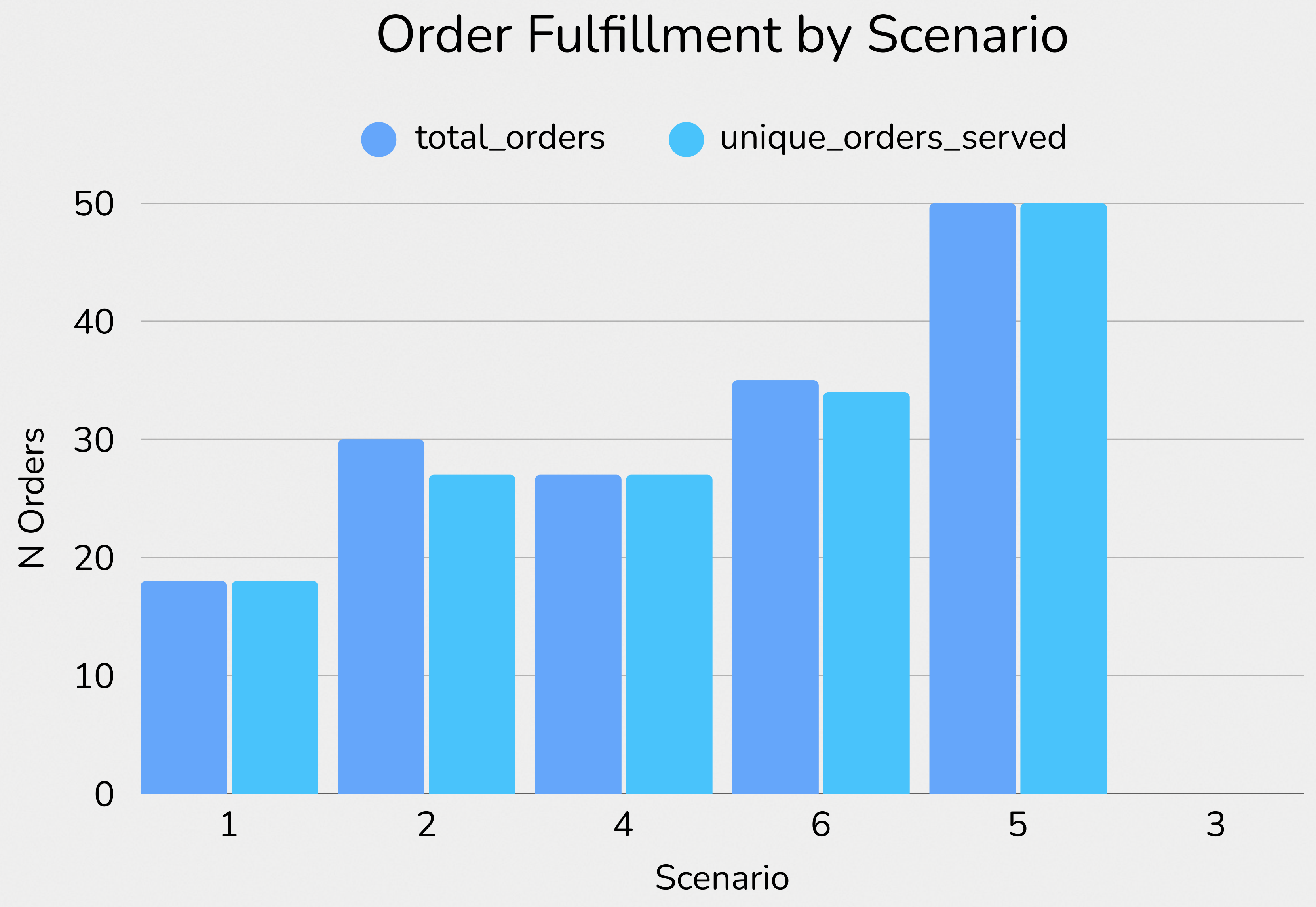
Hybrid Algorithms

This approach focuses on combining graph search, greedy allocation, and evolutionary optimization to solve a logistics problem. It builds a simulation environment where routes between warehouses and customers are computed using Dijkstra's algorithm, inventory is distributed through a greedy heuristic, and the best delivery plans are refined using a hybrid genetic algorithm that evolves over iterations. Each solution is evaluated based on cost, distance, and fulfillment rate, receiving penalties for inefficiency and unfulfilled orders, until an optimized routing and allocation plan emerges.



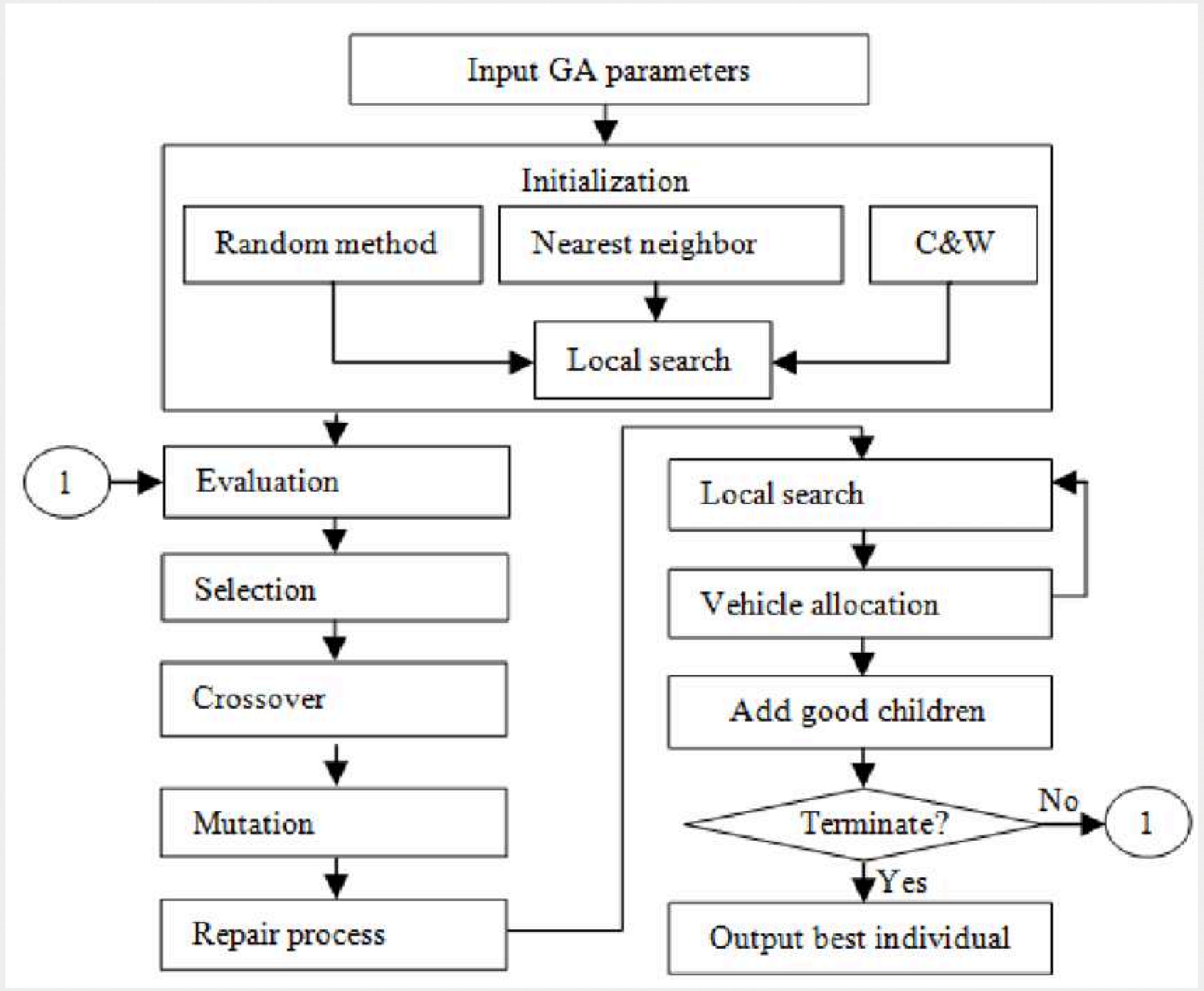
Hybrid Results



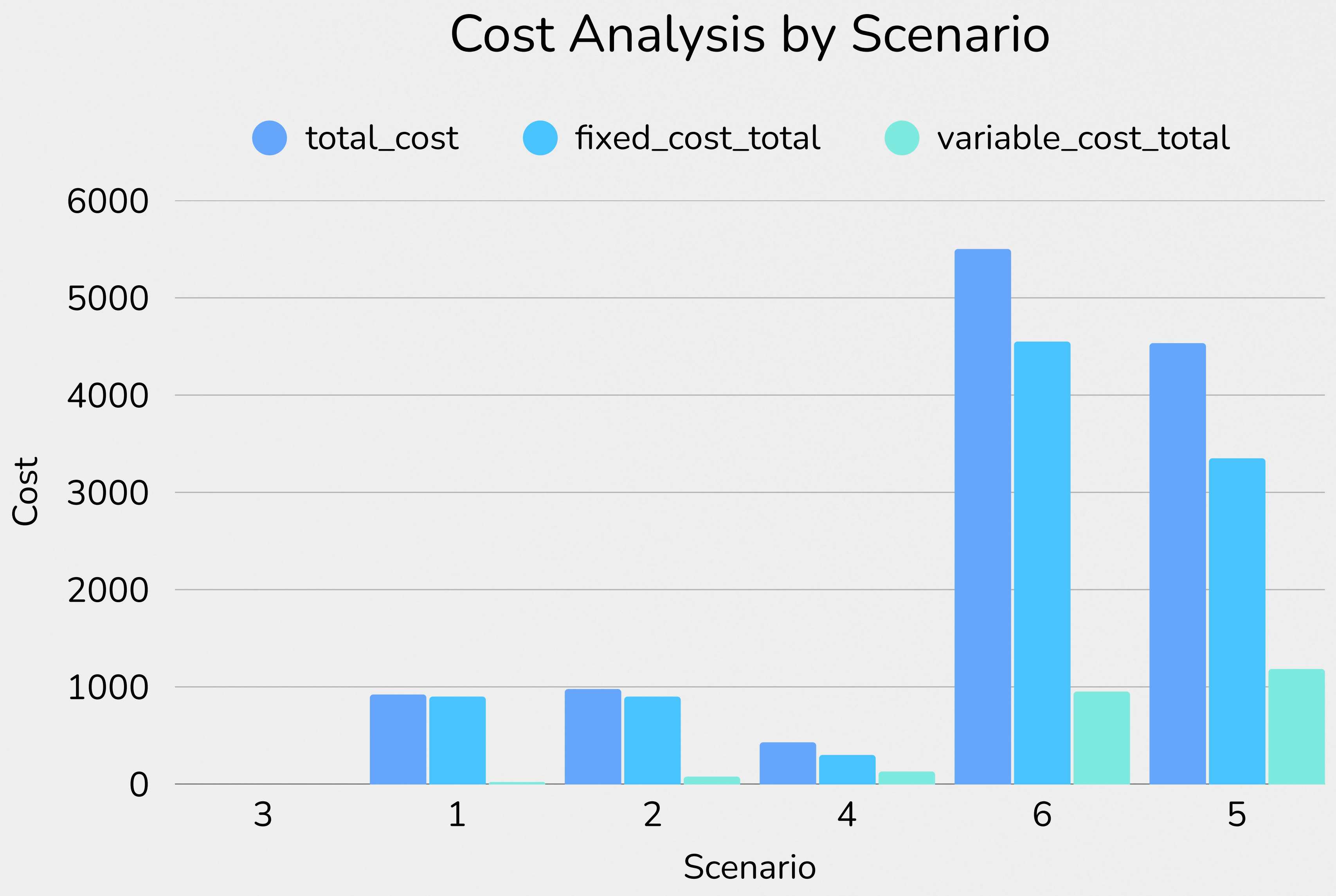


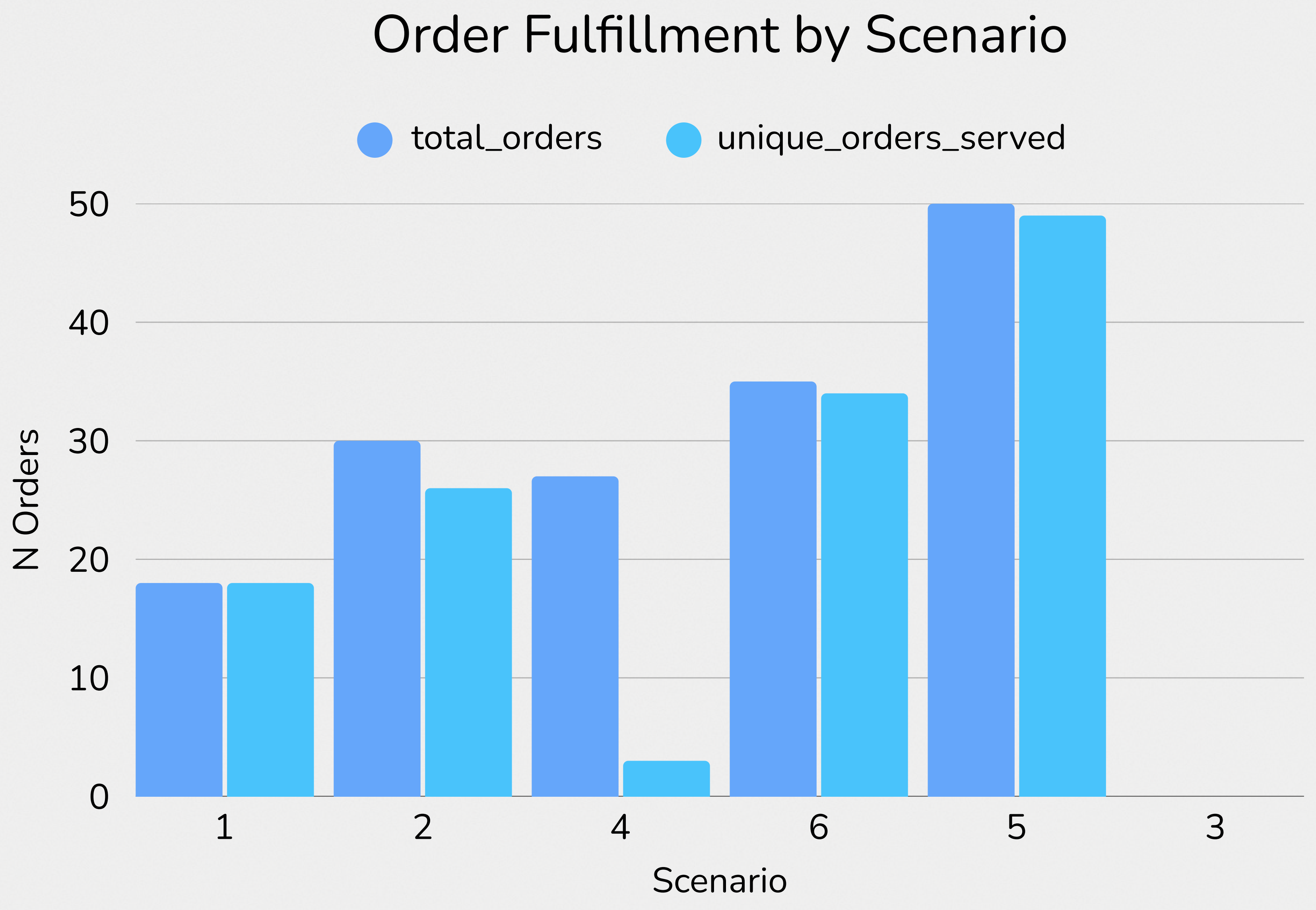
Memetic Algorithm

A Memetic Algorithm (MA) is an advanced optimization method that combines the global search ability of evolutionary algorithms (like genetic algorithms) with the local refinement of heuristic or gradient-based optimization.
It evolves a population of candidate solutions through selection, crossover, and mutation, but also applies local search to improve individual solutions before the next generation. This hybrid approach balances exploration and exploitation, allowing faster and more accurate convergence to near-optimal solutions.



Memetic Results





Strengths vs Weaknesses

Memetic	Hybrid	RL
Effective for combinatorial problems	High fulfillment consistency	Penalty&reward help prioritize a variable over others
High solution quality	Scalable optimization	Connects relationships using GNN
Robustness	Adaptive route construction	Agent learns a reliable strategy

Memetic	Hybrid	RL
Suboptimal local search	Suboptimal local search	Costly computations
High computational cost	High computational cost	Agent May Be "Short-Sighted"
Complex parameter tuning	High dependency on initial allocation	Requires significant time to tune

Summary

Metric	Memetic	Hybrid	RL
Avg Fulfillment	65.49%	**96.92%**	82.86%
Median Cost	**949.6**	2393.63	2638.85
Median Distance	**51.88**	166.32	221.15
Median Time	233.86	43.66	**19.41**
Best Cost per Order	37.6	**35.15**	51.93

Quantum is the future? No, Quantum is Now!

As the hackathon was having its opening ceremony, Google was holding an event to disclose that they have achieved a verifiable quantum advantage. Now, we can’t hear of this news and not try to implement Quantum Optimization, especially when quantum excels in finding feasible solutions when classical solvers cannot.

QUBO vs RL on local scenarios

Sadly, there was not enough time to test this on IBM brisbane (real quantum hardware), but the simulated solution on our local computers developed within few hours shows potential. It needs further work and test on the quantum hardware to validate it, but we believe there is a firm possibility it outperforms all classical solutions for this problem.

Metric	Prototype	Baseline
Success Rate (%)	75	100
Avg. Runtime (s)	51.5	0.53
Avg. Cost (on success)	1,005.22	1,660.37
Avg. Fulfillment (%)	66.67	98.33

Team Ne3Na3

N3N3 Optimus Price

Thank you!