Inventory Optimization Problem with Two Conflicting Objectives

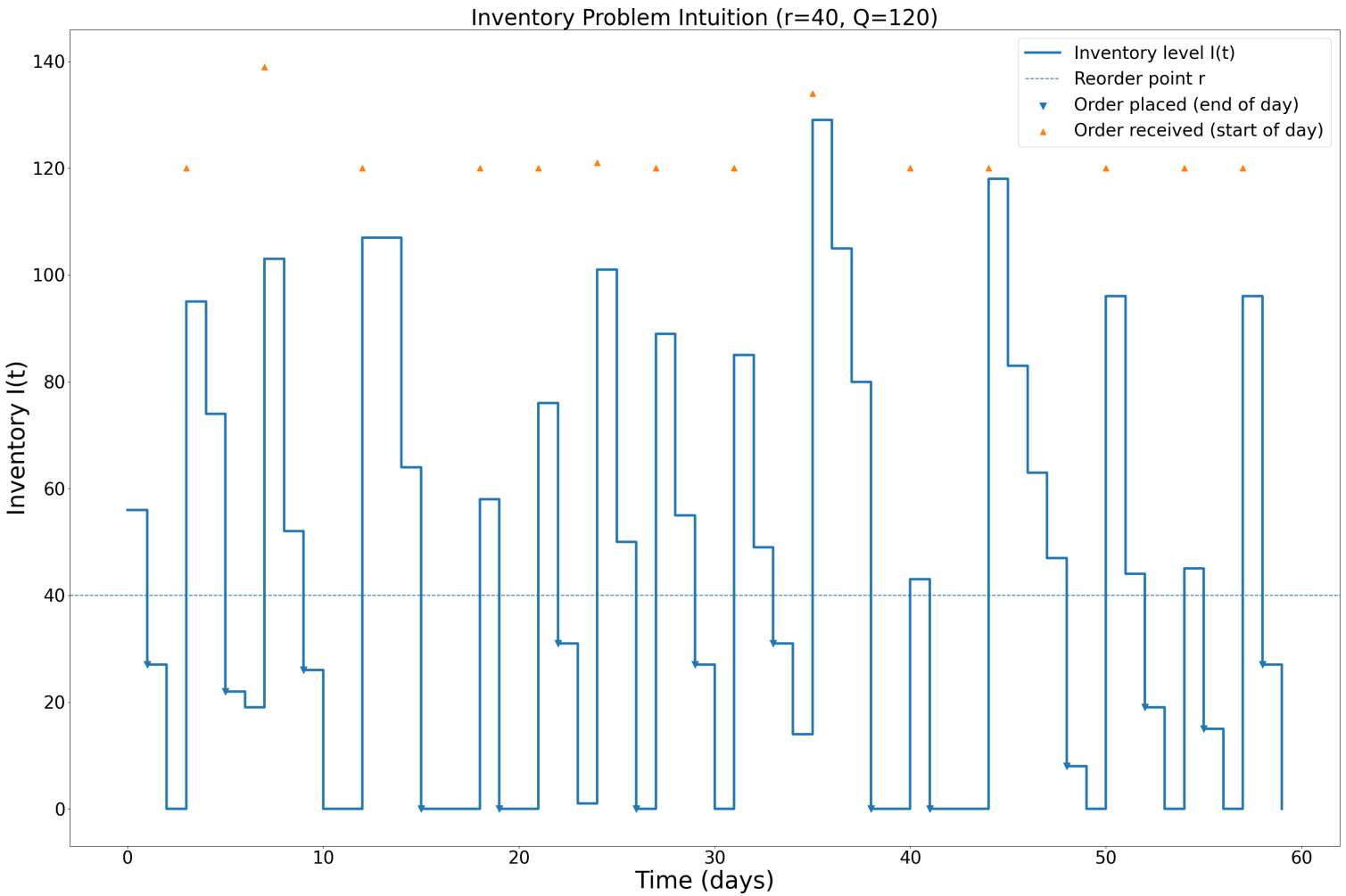
Supervisor: Professor Montaz Ali

Boikanyo Molele, Ibrahim Dan-Dije, Pia Rielly, Vukani Mhlongo, Ayanda Ndimande and Mazwi Masango

January 10, 2026

Outline

Stochastic Inventory Problem



Problem Outline

- Minimise inventory cost function $f = f(x, \xi)$
- Maximise Service level function $g = g(x, \xi)$
- Decision variable $x^T = (r, Q)$ and
  - r = re-order threshold
  - Q = quantity by which to reorder
- Random Vector Variable $\xi^T \in (\xi_1, \xi_2, \xi_3)$.
  - $\xi_1 \sim$ Poisson (Customers)
  - $\xi_2 \sim$ Weibull (Customer Demand)
  - $\xi_3 \sim$ Uniform/ Specified probability distribution (Order Delay Time)

Objectives

- Minimise **Cost** = cost per unit $\times$ Inventory Quantity
- Maximise **Service level** = $\frac{\text{customers served}}{\text{total customers}}$
- Identify **Pareto set** from all possible combinations of r and Q
- Address the **Non-dominance Problem** arising from conflicting objectives using Multi-objective Particle Swarm Optimisation (**MOPSO**) variants.

Simulation Procedure for all (r, Q) Policy

Holding cost: h = cost per unit per period

1: Initialize inventory I
2: serviced $\leftarrow 0$, requiring $\leftarrow 0$
3: **for** $t = 1, \ldots, T$ **do**
4: **if** Receive order **then**
5: $I \leftarrow I + Q$
6: **end if**
7: Serve stochastic customers' demand
8: **if** demand is fully served **then**
9: serviced $\leftarrow$ serviced $+1$
10: **end if**
11: requiring $\leftarrow$ requiring $+1$
12: **if** $I < r$ **and** no order outstanding **then**
13: Place order of size Q with lead time L
14: **end if**
15: Record inventory level I
16: **end for**
17: Average Cost $\leftarrow h \times \text{mean}(I)$
18: Service Level $\leftarrow \dfrac{\text{serviced}}{\text{requiring}}$

Simulation and Optimization Parameters

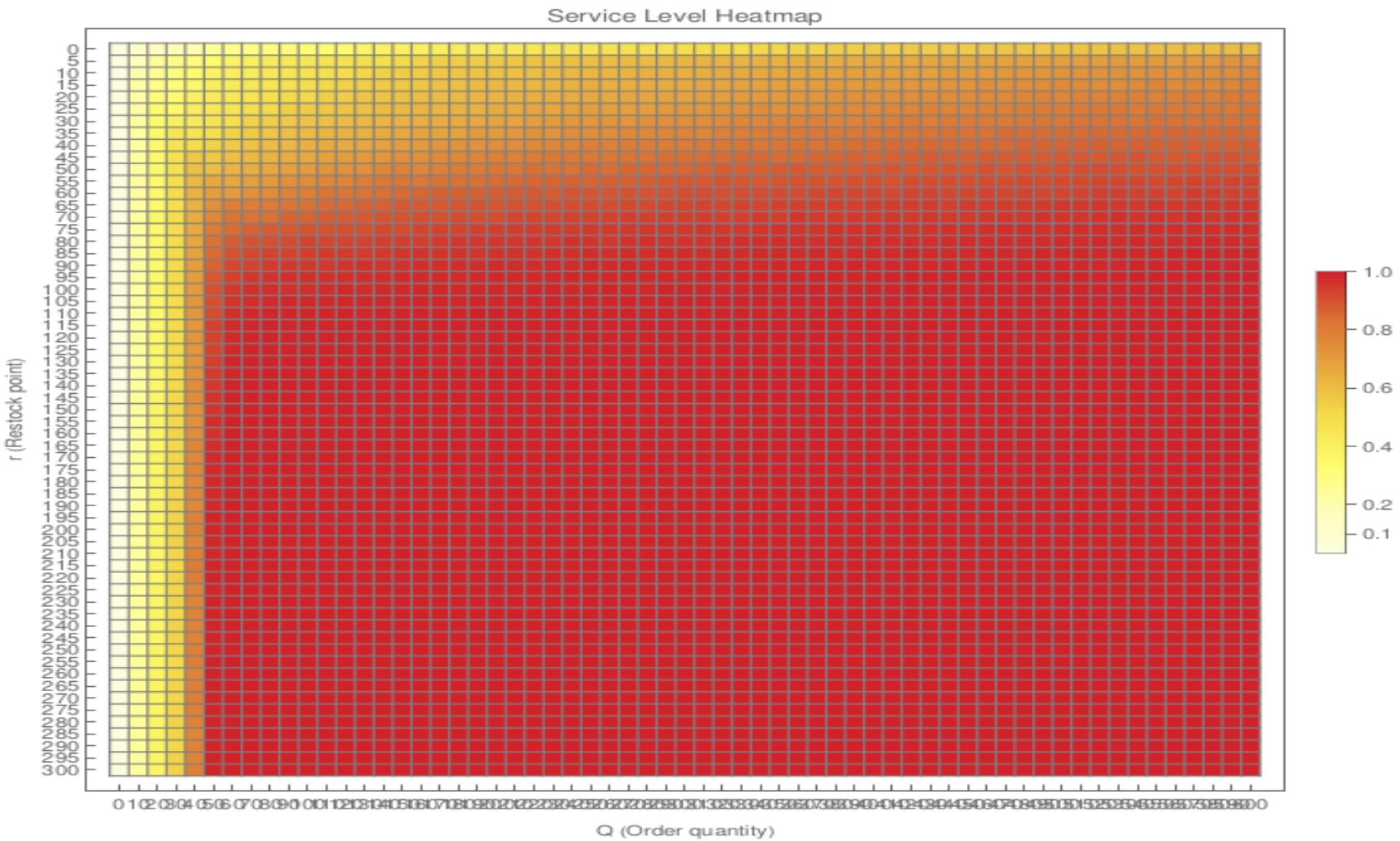
Simulation Parameters

- Planning horizon: $T = 180$ days
- Customer arrivals: Poisson rate $\lambda = 3$ per day
- Demand size: Weibull$(\lambda_w = 15,\ k = 2)$
- Lead time: $P(L = \ell) = \{0.25, 0.50, 0.25\},\ \ell = 1, 2, 3$
- Holding cost: $h = 1$ per unit per day
- Replications per policy: $n_{\text{rep}} = 10$

Optimization Parameters

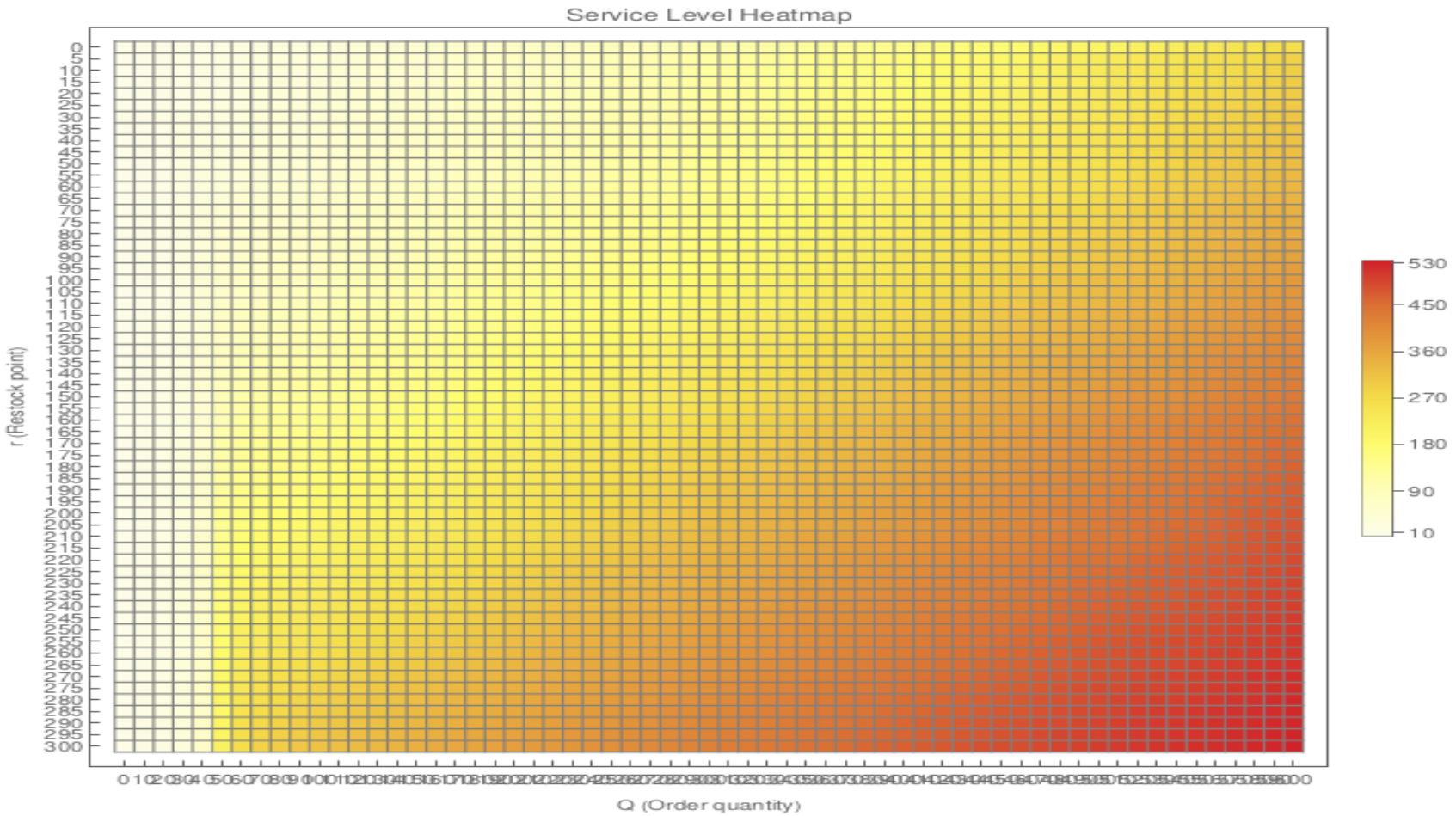
- Number of particles: $N = 100$
- Number of iterations: $I = 500$
- Inertia weight: $w = 0.6$
- Cognitive coefficient: $c_1 = 2$
- Social coefficient: $c_2 = 2$
- Decision variables:
  - Reorder point: $r \in [0, 100]$
  - Order quantity: $Q \in [0, 100]$

Some Simulation Results – Service Level



Service level performance of the inventory system, $Q \in [0 : 600]$ and $r \in [0 : 300]$.

Some Simulation Results - Cost



Cost level performance $Q \in [0 : 600]$ and $r \in [0 : 300]$.

Pareto Set

- Pareto set: set of all non-dominated solutions in the decision space, where no solution is strictly better than another in all objectives.

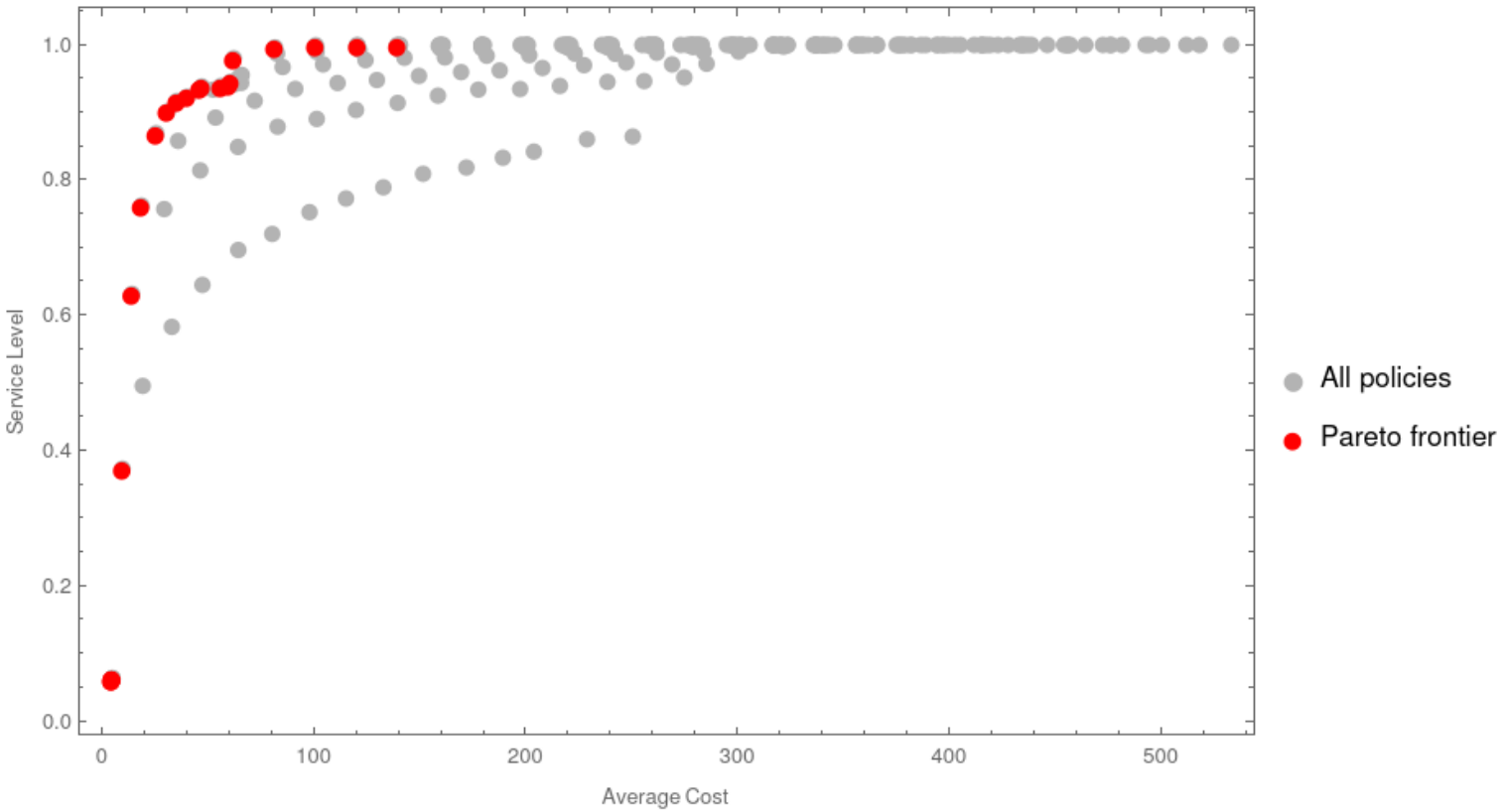
- Solution A dominates B if:

$$f(A) < f(B) \text{ and } g(A) > g(B)$$

where f is Cost and g is Service level.

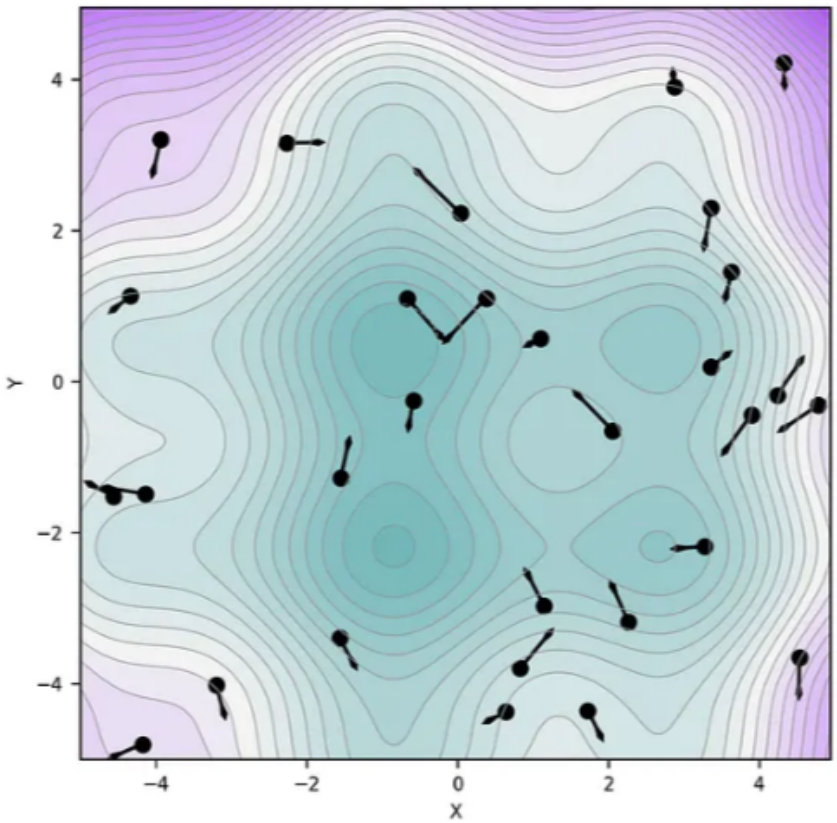
Pareto Set Results

{{0, 40}, {20, 0}, {20, 40}, {40, 0}, {40, 40}, {60, 0}, {60, 40}, {60, 80}, {80, 40}, {80, 80}, {100, 40}, {100, 80},
{120, 0}, {120, 40}, {120, 80}, {140, 40}, {140, 80}, {160, 80}, {180, 40}, {200, 40}, {220, 0}, {240, 40}}



Note the top list is the corresponding (r, Q) solution values.

Particle Swarm Optimisation (PSO) Illustrative Image



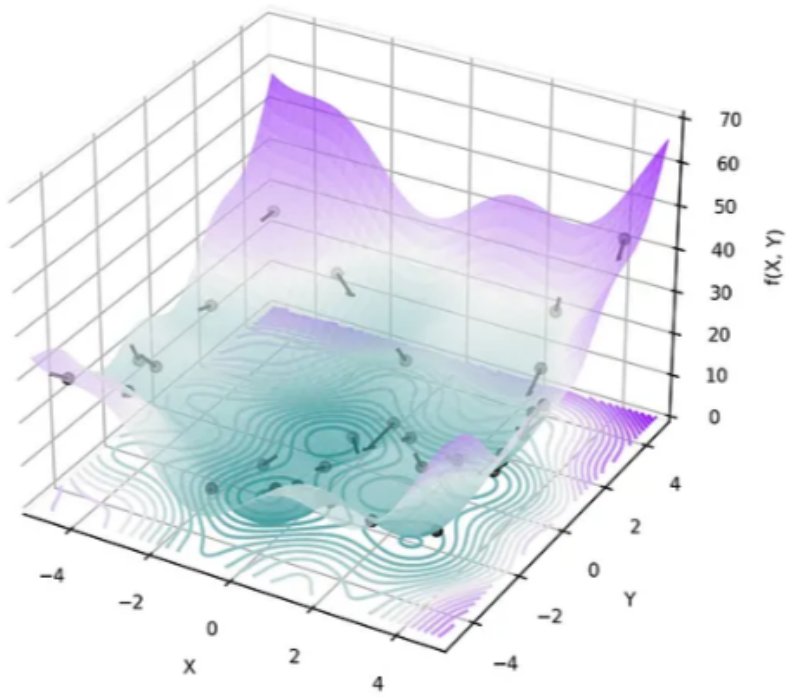


Image credit: reference [1]

PSO equations

- Velocity update: $v_i^{t+1} = w\, v_i^t + c_1\, r_1(p_i^{\text{best}} - x_i^t) + c_2\, r_2(g^{\text{best}} - x_i^t)$
  - w = inertia weight, $c1$, $c2$ = weightings, $r1, r2$ = random numbers
  - values for w, c1 and c2 taken from the literature.
- Position update: $x_i^{t+1} = x_i^t + v_i^{t+1}$
- Evaluate fitness
- Update personal and global bests
- Multiobjective Challenge
  - Nondominance conflict

Adapting PSO to Multi-Objective Problems

1. Dominance-filter method
2. Iterative pareto-curves
3. Ideal-point method

MOPSO: Dominance-filter method

- Each particle is compared pairwise with all others.
- A particle wins if it is no worse in all objectives and better in at least one (pareto dominance principle).
- Each win increments the particle's dominance score.
- Particles with higher dominance score are considered better.
- Apply the normal PSO velocity update equation, with p_i^{best} and g^{best} determined as above.

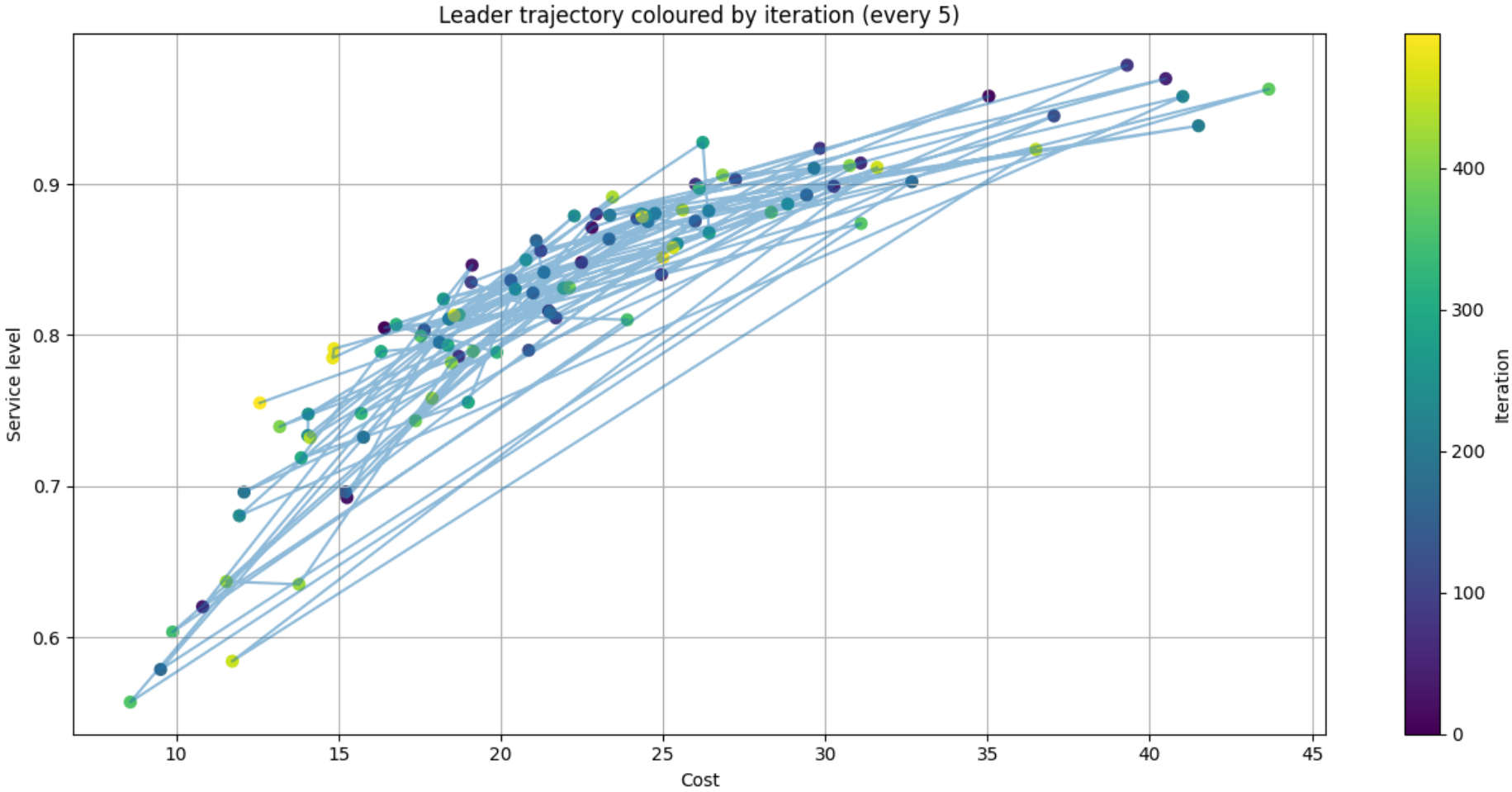
MOPSO: Iterative pareto-curves method

- Identify the current Pareto front of particles.
- Randomly choose two points from the pareto set, say **a** and **b**.
- For each particle $\mathbf{x_i}$, compute the angle between $\mathbf{x_i}$ and **a** and **b**.
- Update the velocity to move the particle toward the pareto point corresponding to the smallest angle.
- $v_i^{t+1} = r_1 * v_i^t + r_2(b - x_i^t)$ where r_1, r_2 are random numbers, **b** is the closest pareto point.
- Update the position of the particle then Iterate.

MOPSO: Ideal-point method

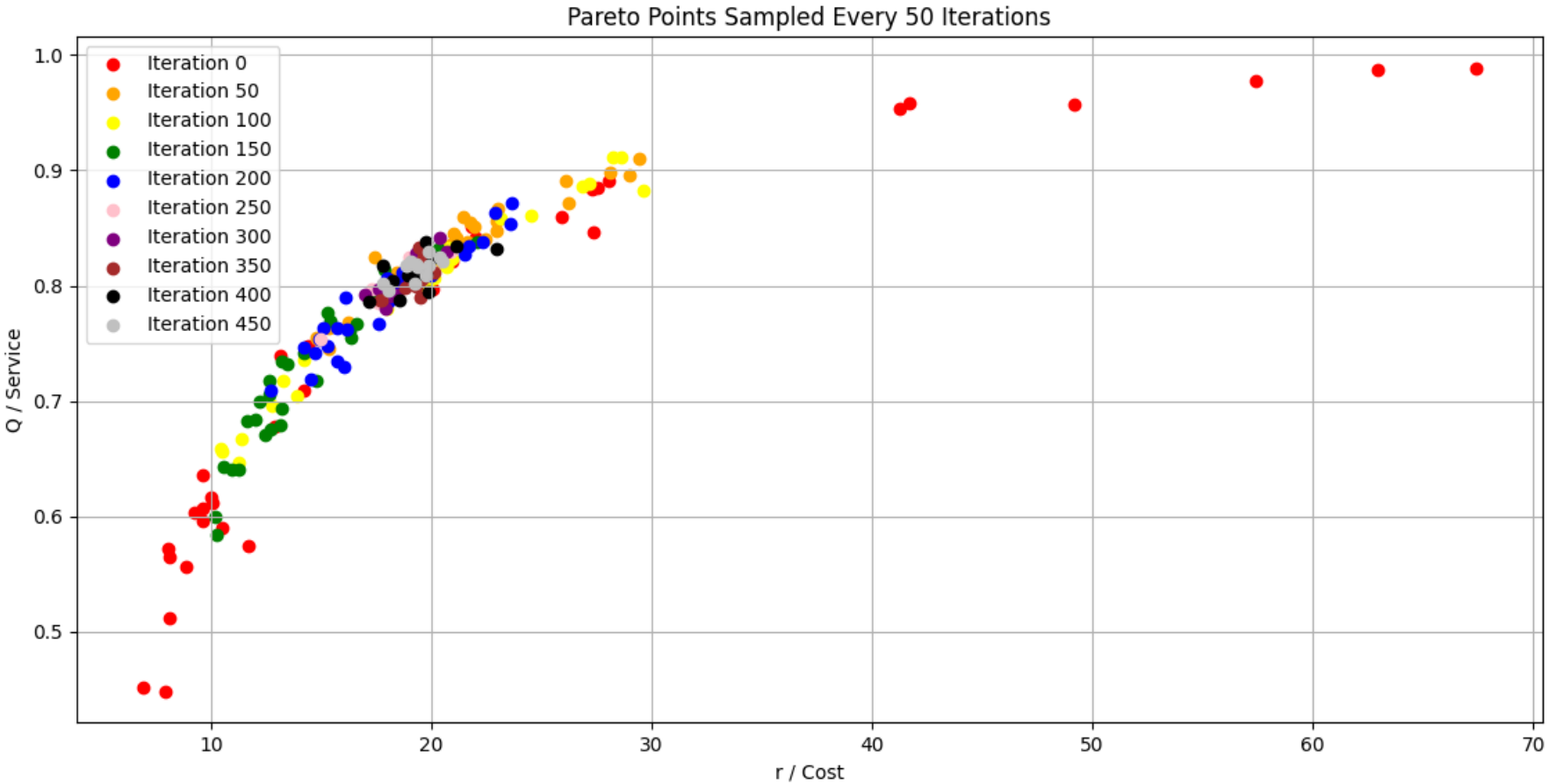
- Define an ideal point using the best value of each objective (in this case, (cost, service) = (0,1).)
- Compute each particle's euclidean distance to the ideal point.
- Closest particles are selected as leaders.
- Apply the normal PSO velocity update equation, with p_i^{best} and g^{best} determined as above.
- Drives convergence toward a single compromised solution.

Results: MOPSO with Dominance Filtering



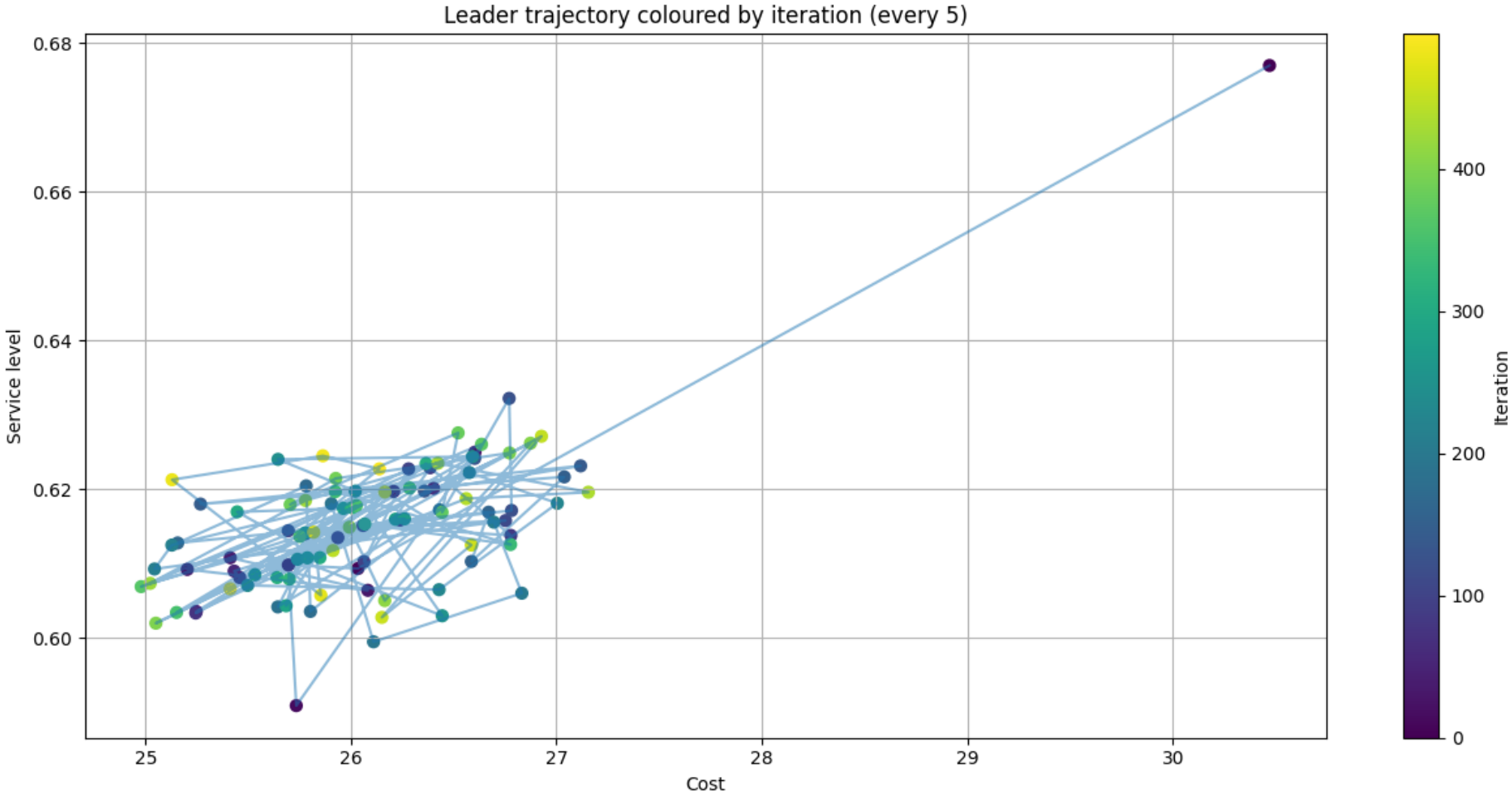
Parameters: 100 particles, 500 iterations, $r, Q \in \{1, 100\}$

Results: MOPSO with Pareto Curves



Parameters: 100 particles, 500 iterations, $r, Q \in \{1, 100\}$.

Results: MOPSO with Ideal Point Attraction



Parameters: 100 particles, 500 iterations, $r, Q \in \{1, 100\}$.

Interpretation and Analysis

- PSO with dominance-filtering seems to converge to a similar Pareto front as seen earlier.
- PSO with iterative pareto-curves clearly shows convergence of the curves to a smaller group, (incidentally the closest to the ideal point), and agrees with dominance-filtering results.
- PSO with ideal-point method converged to a different locus than the previous methods. Further adjustments are needed (e.g. additional constraints on cost) in order to weight the distance function used, such that it would converge to a similar/better locus.

Conclusion

- The inventory problem involving two conflicting objectives was successfully simulated.
- The pareto set was then determined.
- Three methods of MOPSO were explored, two of which were seen to successfully optimised the problem.

Limitations and Future Work

- Limitations:
  - more realistic constraints are needed for the original simulation (e.g. maximum inventory, maximum cost)
  - different/more nuanced conditions on demand satisfaction process (e.g. knapsack algorithm).
- Future work:
  - adaptive feedback models.
  - hyper-parameters on MOPSO models.
  - using weighted distance-to-ideal-point MOPSO method.

References

1. Thevenot, A., 2020. Particle Swarm Optimization (PSO) Visually Explained. Medium. [online] Available at: https://medium.com/data-science/particle-swarm-optimization-visually-explained-46289eeb2e14 [Accessed 9 January 2026].
2. Brea, F. Fachinotti, V.D., 2017. A computational multi-objective optimization method to improve energy efficiency and thermal comfort in dwellings. Energy and Buildings, 154, pp.283–294. doi:10.1016/j.enbuild.2017.08.002.
3. Mysore, A. et al., 2026. CMOPSO: Competitive Mechanism based Multi-objective Particle Swarm Optimizer. pymoo: Multi-objective Optimization in Python. [online] Available at: https://pymoo.org/algorithms/moo/cmopso.html [Accessed 9 January 2026].

Thank you!

Questions?