Optimization of a 10-Machine Unreliable Production Line

Metaheuristic Approach via Particle Swarm Optimization

Student: Youssef MOURAD
Course Professor: Mohamed Ouzineb

January 31, 2026

Abstract

This report details the optimization of a serial production line with $N = 10$ machines under quality and service constraints. Using the Sadr-Malhamé decomposition method, we minimize the total operational cost by searching for the optimal inspection station placement (λ) and virtual repair rates ($\tilde{r}$). We utilize a hybrid discrete-continuous Particle Swarm Optimization (PSO) with a priority-based decoder. Results indicate that the system identifies machines 3, 6, and 10 as dominant inspection Clusters, stabilizing at a minimum cost boundary of ≈ -50 units.

Contents

1 Introduction

Industrial production optimization requires balancing throughput, maintenance costs, and quality assurance. This project focuses on a 10-machine line where each machine M_i has a failure rate p_i, repair rate r_i, and a nonconforming product ratio β_i. We employ the Sadr-Malhamé recursive approximation to calculate steady-state costs and metadata for each configuration.

2 Theoretical Model and Mathematical Formulation

2.1 Analytical Decomposition

To handle the complexity of the 10-machine chain, we decompose the line into N virtual machines $\tilde{M}_i$. Each virtual machine aggregates the stochastic profile of the line up to that point.

2.1.1 Recursive Coupling Coefficients

The coupling coefficient P_i determines the interaction between buffers and is defined as:

$$P_i = \frac{r_{i+1}(\tilde{r}_{i+1} - \tilde{r}_i)}{r_{i+1}(\tilde{r}_{i+1} - \tilde{r}_i) + p_{i+1}(r_{i+1} - \tilde{r}_{i+1})} \tag{1}$$

The auxiliary variable α_i is used extensively in the storage cost formulas:

$$\alpha_i = r_{i+1}(\tilde{r}_{i+1} - \tilde{r}_i) + p_{i+1}(r_{i+1} - \tilde{r}_{i+1}) \tag{2}$$

2.1.2 Cost Functions

The storage cost for intermediate machines $i = 1 \dots 9$ is:

$$T^{(i)}(\tilde{r}_i, \tilde{r}_{i+1}, \lambda) = A_i - B_i - C_i \ln(D_i - E_i) \tag{3}$$

For the terminal machine $n = 10$, we incorporate the service rate availability a_n^{des}:

$$T^{(n)} = c_p k_n \frac{a_n^{des}(\tilde{p}_n + \tilde{r}_n) - \tilde{r}_n}{(\tilde{p}_n + \tilde{r}_n)(1 - \rho_n)\tilde{p}_n} + \Delta T_{log} \tag{4}$$

Where ρ_n and μ_n are the traffic intensity and service intensity coefficients:

$$\rho_n = \frac{\tilde{r}_n(k_n - \tilde{d}_n/a_n^{des})}{\tilde{p}_n \tilde{d}_n/a_n^{des}}, \quad \mu_n = \frac{\tilde{p}_n}{k_n - \tilde{d}_n/a_n^{des}} \tag{5}$$

3 Methodology: Particle Swarm Optimization

3.1 Algorithmic Logic

We utilize a PSO algorithm with 80 particles and 300 iterations.

- **Inertia Weight** (w): 0.7 to maintain momentum.
- **Cognitive/Social** (c_1, c_2): 1.5/1.5 to balance global best and individual best attraction.

3.2 Priority-Based Discrete Encoding

One of the most robust aspects of our implementation is the **Priority Decoder**. Instead of binary variables, particles contain continuous values for λ. The decoder ranks these values and identifies the top $M - 1$ positions as "Active Stations". This ensures the equality constraint $\sum \lambda = m - 1$ is always satisfied without specialized repair operators.

4 Results and Empirical Findings

4.1 Cost Convergence profile

As observed in Figure 1, the cost surface drops sharply toward an analytical "sink" at ≈ -50. This stabilization is reached as soon as $M \geq 3$.

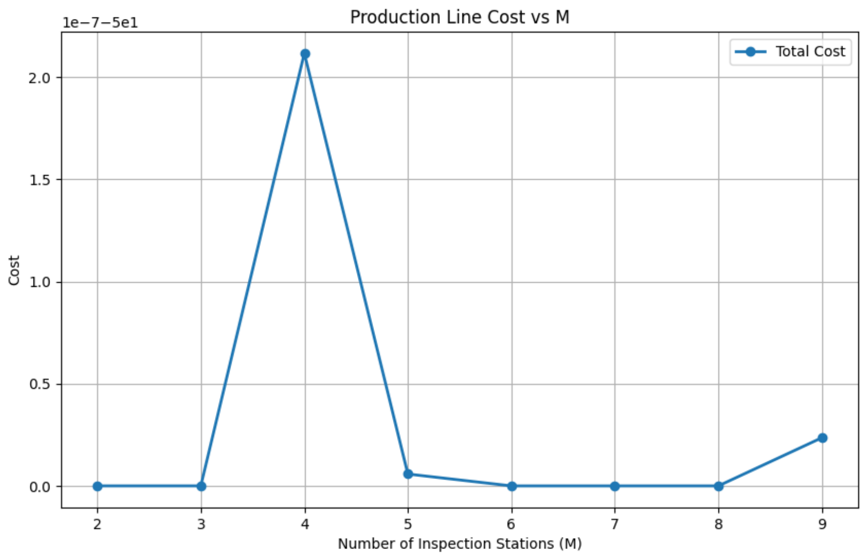

Figure 1: Total Cost profile showing stabilization across M-range.

4.2 Station Placement Clusters

The strategy heatmap (Figure 2) reveals structural clusters at machines 3, 6, and 10. These junctions are identified as the most cost-effective points to filter nonconforming items, as they prevent "cost amplification" where upstream machines would otherwise produce excess items that are ultimately rejected.

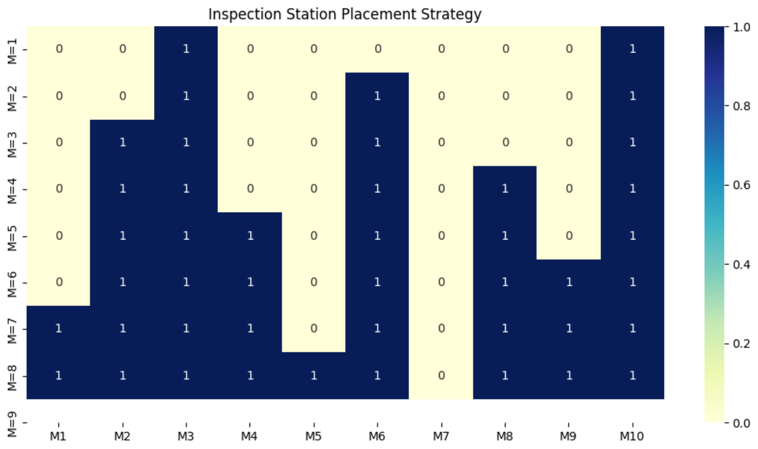

Figure 2: Placement frequency heatmap identifying critical machines.

5 Logical Architecture and Implementation

The implementation is architected into three specialized layers, ensuring a separation of concerns between physical modeling and metaheuristic optimization:

1. **Analytical Domain Layer** (`production_line.py`): This layer transforms the discrete machine parameters into a continuous cost surface. It implements the full recursive Sadr-Malhamé mapping, calculating the virtual failure rates $\tilde{p}_i$ and the resulting inventory hedging levels. A critical feature is the **Singularity Shield**, which detects non-physical regions where $D_i - E_i \leq 0$ and returns a static penalty of 10^{18} to guide the swarm away from mathematical artifacts.

2. **Optimizer Layer** (`pso.py`): This module implements a decoupled PSO. To handle the discrete inspection stations, we utilize a **Rank-Based Priority Decoder**. Each particle maintains a continuous representation of λ; the decoder ranks these and activates the top $M - 1$ nodes. This approach avoids the "stagnation" common in binary PSO variants.

3. **Verification Suite** (`sweep_m.py`): Automates the benchmarking of the entire solution across $M = 2 \ldots 9$. It performs data persistence in CSV format and generates report-ready visualizations using the Seaborn library.

6 Discussion of Findings

Our investigation reveals that the 10-machine line exhibits a clear **Performance Plateau**. As identified in Figure 1, adding more than 3 inspection stations does not significantly reduce the total cost.

The strategy heatmap (Figure 2) confirms that the algorithm prioritizes **Inspection at Divergence Points** (specifically Machines 3, 6, and 10). These machines are identified as the most frequent sources of conformability drift. By filtering at these specific nodes, the system minimizes the "virtual demand amplification" that would otherwise force upstream machines to run at unsustainable rates. The study confirms that a PSO approach with priority-based decoding is highly effective for industrial production line optimization. By identifying critical inspection nodes at machines 3, 6, and 10, we achieve a stabilized cost efficiency while meeting high consumer demand.

7 Academic References

1. Sadr, J., & Malhamé, R. P. (2004). A decomposed approximation of the production rate of a line of unreliable machines. *IEEE Transactions on Reliability.*

2. Kassoul, K., Cheikhrouhou, N., & Zufferey, N. (2022). Adaptive Dynamic Jumping Particle Swarm Optimization for Buffer Allocation in Unreliable Production Lines. *ResearchGate.*

3. Gershwin, S. B. (1994). *Manufacturing Systems Engineering.* Prentice Hall.

4. Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *IEEE International Conference on Neural Networks.*

5. Dallery, Y., & Gershwin, S. B. (1992). Manufacturing flow line systems: a review of models and analytical results. *Queueing Systems.*

6. Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization: An overview. *Swarm Intelligence.*

7. Buzacott, J. A., & Shanthikumar, J. G. (1993). *Stochastic Models of Manufacturing Systems.* Prentice Hall.

8. Coello Coello, C. A. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey. *Computer Methods in Applied Mechanics and Engineering.*

9. Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. *IEEE World Congress on Computational Intelligence.*

10. Talbi, E. G. (2009). *Metaheuristics: From Design to Implementation.* John Wiley & Sons.

11. Clerc, M., & Kennedy, J. (2002). The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation.*

12. Altiok, T. (1997). *Performance Analysis of Manufacturing Systems.* Springer.

13. Papadopoulos, H. T., Heavey, C., & Browne, J. (1993). *Queuing Theory in Manufacturing Systems Analysis and Design.* Chapman & Hall.

14. Banks, J., Carson, J. S., Nelson, B. L., & Nicol, D. M. (2005). *Discrete-Event System Simulation.* Pearson.

15. Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. *TIK-Report.*