AAAI Press Anonymous Submission Instructions for Authors Using LaTeX

Anonymous submission

Abstract

AAAI creates proceedings, working notes, and technical reports directly from electronic source furnished by the authors. To ensure that all papers in the publication have a uniform appearance, authors must adhere to the following instructions.

Introduction

Technical Appendix

Plant Operations

Predictive Maintenance: The task of predicting the Remaining Useful Life (RUL) is formulated as a sequential decision-making problem within a Markov Decision Process (MDP) framework. The objective of the learning model is to learn a policy π that, given the current state s_t of the industrial equipment, can accurately estimate the remaining number of operational time steps before failure. The model is trained in a simulation environment constructed from historical run-to-failure data, which serves as a data-driven emulator of the equipment's degradation dynamics. At each time step t, the model observes a state vector s_t—comprising sensor readings, health indicators, and their temporal change rates—generates an RUL prediction a_t, and receives a reward r_t based on the prediction's accuracy. Through repeated interaction, the model learns a policy that maximizes long-term prediction accuracy or, equivalently, minimizes cumulative prediction error. The underlying MDP is defined by the following components. The state space ($\mathbf{s_t} \in \mathbf{S}$) represents the condition of the industrial equipment as a feature vector constructed by combining sensor data, observable health parameters that characterize degradation, and their discrete rates of change (degradation velocity) over a fixed interval. The action space ($\mathbf{a_t} \in \mathbf{A}$) is a continuous scalar value denoting the model's predicted RUL, expressed as $a_t = \text{RUL}_{\text{predicted}}(t)$. The reward function ($\mathbf{r_t}$) imposes an asymmetric penalty based on the normalized prediction error $\tilde{\epsilon}_t$:

$$r(s_t, a_t) = \begin{cases} \lambda_{\text{early}} \cdot \tilde{\epsilon}_t, & \text{if } a_t < \text{RUL}_{\text{true}}(t), \\ \lambda_{\text{late}} \cdot \tilde{\epsilon}_t, & \text{otherwise,} \end{cases}$$

where λ_{early} and λ_{late} are penalty coefficients corresponding to underestimation and overestimation, respectively. This asymmetric design allows the learning model to balance proactive risk mitigation—by penalizing late predictions more severely—and the avoidance of false alarms, by assigning higher penalties to early predictions when desired. We employ the Soft Actor-Critic (SAC) algorithm to solve this continuous control MDP. SAC is an off-policy actor-critic method that promotes sample efficiency and stable convergence by incorporating an entropy term into the reward objective to enhance exploration. The objective is to maximize:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[\gamma^t r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))\right]$$

where π is the policy, ρ_π the state–action distribution, γ the discount factor, $\mathcal{H}$ the policy entropy, and α a temperature parameter balancing reward against entropy. This encourages the model to optimize long-term prediction accuracy while exploring diverse degradation patterns, avoiding overconfident predictions. The SAC architecture comprises three key components: a stochastic actor (policy network, π_ϕ) that maps states to a Gaussian action distribution using reparameterization for differentiable sampling; twin critic networks ($\mathbf{Q}_{\theta_1}, \mathbf{Q}_{\theta_2}$) that independently estimate state–action values to reduce overestimation bias; and target networks ($Q_{\theta'_1}, Q_{\theta'_2}$) that stabilize learning via Polyak averaging. We enhance sample efficiency using a Prioritized Experience Replay (PER) buffer, which samples transitions with probability proportional to their temporal-difference (TD) error:

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

where p_i is the priority of transition i and α controls prioritization strength, focusing learning on informative experiences. Training proceeds episodically over run-to-failure trajectories. At each step t, the model observes state s_t, predicts RUL a_t, and receives accuracy-based reward r_t. Transitions are stored in the PER buffer, with mini-batches used to update networks: critics minimize Bellman error, while the actor maximizes entropy-augmented expected reward. The temperature parameter α is automatically tuned to balance exploration and exploitation. Model generalization is assessed on a validation set using MAE and MAPE between predicted and true RUL values, enabling accurate, data-driven RUL estimation for predictive maintenance. In

short, Predictive maintenance is traditionally framed as a supervised regression problem, focusing on estimating equipment's RUL. In contrast, reinforcement learning approaches frame maintenance planning and intervention as MDPs, enabling sequential, policy-driven decision-making. This allows RL agents to learn optimal strategies for when to perform maintenance and what specific actions to take, maximizing system uptime while minimizing costs and risks.

Plant-Wide Optimization and Control of Industrial Systems: Industrial plant optimization employs a hierarchical framework combining process optimization and control. At the supervisory level, process optimization determines optimal setpoints for controlled variables (CVs) by balancing economic objectives like profit or efficiency against process constraints and operating conditions. At the regulatory level, process control maintains CVs at their setpoints by adjusting manipulated variables, ensuring disturbance rejection and operational stability. This integrated approach enables plants to achieve both economic optimization and safe operational reliability in dynamic environments. This work addresses learning regulatory control policies directly from historical plant data, providing a data-driven alternative when accurate first-principles models are unavailable. The control task is formulated as a continuous-state, continuous-action Markov Decision Process (MDP), where an autonomous reinforcement learning agent learns a feedback control policy mapping process states to optimal manipulated variable (MV) adjustments. The MDP is defined by the tuple (S, A, P, R), where the state space ($\mathbf{s_t} \in S$) represents the plant's dynamic operating condition through measured and inferred variables. The action space ($\mathbf{a_t} \in A$) specifies manipulated variable values. The reward function encodes control objectives through a weighted penalty formulation:

$$r(s_t, a_t) = -\left[w_{\text{track}} \cdot \frac{|CV(s_t) - SP|}{\epsilon_{\max}} + w_{\text{act}} \cdot \|a_t - a_{t-1}\|^2 \right],$$

where $CV(s_t)$ is the controlled variable, SP its target setpoint, and w_{track}, w_{act} weight tracking precision against actuation smoothness. Our methodology employs an offline, model-based reinforcement learning approach with two sequential phases. First, offline system identification trains a deep neural network f_θ to approximate system dynamics $s_{t+1} = f(s_t, a_t)$ from historical data $\mathcal{D} = \{(s_t, a_t, s_{t+1})\}$ by minimizing:

$$\theta^* = \arg\min_\theta \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}} \left[\|f_\theta(s_t, a_t) - s_{t+1}\|^2 \right].$$

The trained model f_{θ^*} serves as a differentiable, data-driven simulator for risk-free policy training. Second, offline policy optimization uses the Soft Actor-Critic (SAC) algorithm within a maximum entropy framework:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} \left[\gamma^t r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right],$$

where π is the policy, ρ_π the state-action marginal, and α a temperature parameter. The agent architecture includes a stochastic actor (π_ϕ) parameterizing a Gaussian policy and twin critic networks ($Q_{\theta_1}, Q_{\theta_2}$) employing double-Q learning. Target networks and experience replay ensure stable, efficient off-policy learning. The training process is conducted episodically and entirely offline within the learned environment. The agent's networks are updated using mini-batches sampled from the replay buffer, where the critics minimize the Bellman error while the actor maximizes expected future reward and entropy. For evaluation, the trained policy is deployed in simulation and benchmarked against historical operational data. Performance is assessed based on the policy's ability to maintain the controlled variable at setpoint with minimal deviation while ensuring smooth manipulated variable adjustments. This offline model-based RL framework enables long-horizon planning and direct optimization of shaped reward functions that encode operational objectives, process constraints, and economic goals, thereby bridging traditional model-based control with deep reinforcement learning. In short, traditional plant-wide process control relies on model-based predictive control using predefined first-principles process models. In contrast, RL reframes control as a sequential decision-making problem, enabling adaptive, model-free policies that optimize long-term performance under uncertainty.

Knowledge Engineering via Offline Adversarial Imitation Learning We address the problem of distilling human operator expertise from a fixed dataset of historical operation trajectories into a control policy. Each trajectory $\tau_i = \{(s_t^i, a_t^i)\}_{t=1}^{T_i}$ consists of process states s_t (encompassing all measured, controlled, and dependent variables) and actions a_t (the manipulated variables adjusted by the operator). The goal is to learn a stochastic policy $\pi_\phi(a|s)$ that mimics the expert's decision-making from the dataset $\mathcal{D} = \{\tau_i\}_{i=1}^N$, preserving this tacit knowledge for safer and more intelligent operation. We model the control problem as a Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ with continuous states $\mathcal{S}$, continuous actions $\mathcal{A}$, transition dynamics $\mathcal{P}$, an unknown reward function r, and a discount factor γ. The objective is to recover a policy π_ϕ whose state-action occupancy distribution $\rho_{\pi_\phi}(s, a)$ matches that of the expert policy π_E. The occupancy measures the discounted time the policy spends in each state-action pair:

$$\rho_{\pi_\phi}(s, a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t P(s_t = s, a_t = a \mid \pi_\phi)$$

A matched occupancy implies the learned policy is statistically indistinguishable from the expert. Offline Generative Adversarial Imitation Learning (GAIL) frames imitation as a two-player game between a policy π_ϕ and a discriminator $D_\psi(s, a)$. The adversarial objective is:

$$\min_{\pi_\phi} \max_{D_\psi} \mathbb{E}_{\rho_{\pi_E}}[\log D_\psi(s, a)] + \mathbb{E}_{\rho_{\pi_\phi}}[\log(1 - D_\psi(s, a))]$$

The discriminator learns to distinguish expert pairs $(s, a) \sim \rho_{\pi_E}$ from policy-generated pairs $(s, a_\phi) \sim \rho_{\pi_\phi}$, while the policy learns to fool the discriminator. In the offline setting, we sample expert states $s_E \sim \mathcal{D}_E$ and generate corresponding actions with the current policy $a_\phi \sim \pi_\phi(\cdot|s_E)$. The discriminator loss includes a gradient penalty for stability:

$$\begin{aligned}\mathcal{L}_D = &- \mathbb{E}_{(s_E,a_E)\sim\mathcal{D}_E}[\log D_\psi(s_E,a_E)] \\ &- \mathbb{E}_{s_E\sim\mathcal{D}_E,a_\phi\sim\pi_\phi(\cdot|s_E)}[\log(1-D_\psi(s_E,a_\phi))] \\ &+ \lambda\mathbb{E}_{(\hat{s},\hat{a})}\left[(\|\nabla D_\psi(\hat{s},\hat{a})\|_2-1)^2\right]\end{aligned}$$

where $(\hat{s},\hat{a})$ is a random interpolation between expert and policy data: $(\hat{s},\hat{a}) = \alpha(s_E,a_E) + (1-\alpha)(s_E,a_\phi)$ with $\alpha \sim U(0,1)$. The trained discriminator provides a surrogate reward signal: $r_\psi(s,a) = \log\sigma(D_\psi(s,a))$, where $\sigma(\cdot)$ is the sigmoid function. This reward guides the policy to produce expert-like behavior. We use Implicit Q-Learning (IQL) for safe, offline policy optimization. IQL uses the surrogate reward to learn a policy across three conservative learning stages: (a) *Value Function Learning:* The value function $V_\theta(s)$ is learned via asymmetric expectile regression:

$$\mathcal{L}_V = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_2^\tau\left(Q_\omega(s,a) - V_\theta(s)\right)\right]$$

The expectile loss $L_2^\tau(u)$, with $u = Q_\omega(s,a) - V_\theta(s)$, is defined as:

$$L_2^\tau(u) = \begin{cases}\tau\cdot u^2 & \text{if } u \geq 0 \\ (1-\tau)\cdot u^2 & \text{if } u < 0\end{cases}$$

For $\tau > 0.5$, this emphasizes positive advantages, enabling implicit maximization over actions without explicitly querying the policy. (b) *Q-Function Learning:* The Q-function is updated using the Bellman equation with the value target:

$$\mathcal{L}_Q = \mathbb{E}_{(s_E,a_E,s'_E)\sim\mathcal{D}_E}\left[\left(Q_\omega(s_E,a_E) - (r_\psi(s_E,a_E) + \gamma V_\theta(s'_E))\right)^2\right]$$

(c) *Policy Extraction:* The policy is improved via advantage-weighted regression:

$$\begin{aligned}\mathcal{L}_\pi = &- \mathbb{E}_{s_E\sim\mathcal{D}_E}\left[\exp(\beta A(s_E,a_\phi))\log\pi_\phi(a_\phi|s_E)\right], \\ &A(s_E,a_\phi) = Q_{\min}(s_E,a_\phi) - V_\theta(s_E)\end{aligned}$$

where $a_\phi \sim \pi_\phi(\cdot|s_E)$, $Q_{\min} = \min(Q_{\omega_1}, Q_{\omega_2})$ for twin Q-networks, and $\beta > 0$ is an inverse temperature parameter. The complete Offline GAIL–IQL objective integrates adversarial imitation with constrained policy optimization:

$$\begin{aligned}\min_{\pi_\phi,Q_\omega,V_\theta}\max_{D_\psi} \quad &- \mathbb{E}_{(s_E,a_E)\sim\mathcal{D}_E}[\log D_\psi(s_E,a_E)] \\ &- \mathbb{E}_{s_E\sim\mathcal{D}_E,a_\phi\sim\pi_\phi(\cdot|s_E)}[\log(1-D_\psi(s_E,a_\phi))] \\ &+ \lambda\mathcal{L}_{\text{grad-pen}} + \mathcal{L}_V + \mathcal{L}_Q + \mathcal{L}_\pi\end{aligned}$$

GAIL and IQL serve complementary roles: GAIL's discriminator defines *what* to learn via adversarial reward shaping, while IQL defines *how* to learn it safely via conservative offline policy optimization. Policy performance is evaluated using Mean Squared Error between expert and learned actions on held-out states: MSE $= \frac{1}{N}\sum_{i=1}^N \|a_E^{(i)} - a_\phi^{(i)}\|^2$. This framework distills operational expertise into robust control policies without online interaction.

Knowledge Engineering Via Inverse Reinforcement Learning The fundamental goal is to automate complex human skills by extracting and embedding an expert process plant operator's implicit knowledge into a robust, AI-driven control policy. We aim to capture this tacit understanding from historical data, distilling it into an autonomous agent capable of safe and intelligent operation, just like its human counterpart. We frame the problem of learning from expert demonstrations using Adversarial Inverse Reinforcement Learning (AIRL). Given a dataset of expert trajectories $\mathcal{D} = \{\tau_i\}_{i=1}^N$, where each trajectory $\tau_i = \{(s_t,a_t,s_{t+1})\}_{t=1}^{T_i}$ contains state-action-next state tuples, our goal is to recover the expert's underlying reward function $r_\psi(s,a)$ and a policy $\pi_\phi(a|s)$ that mimics their behavior. AIRL uses an adversarial game between a discriminator D_ψ and a policy π_ϕ:

$$\min_{\pi_\phi}\max_{D_\psi}\ \mathbb{E}_{(s,a)\sim\rho_{\pi_E}}[\log D_\psi(s,a)] + \mathbb{E}_{(s,a)\sim\rho_{\pi_\phi}}[\log(1-D_\psi(s,a))]$$

The policy learns to generate state-action pairs that the discriminator cannot distinguish from expert data. The discriminator has a specific structure to enable reward recovery:

$$D_\psi(s,a,s') = \frac{\exp(f_\psi(s,a,s'))}{\exp(f_\psi(s,a,s'))+1}$$

where $f_\psi(s,a,s')$ is an advantage-like function:

$$f_\psi(s,a,s') = r_\psi(s,a) + \gamma V_\psi(s') - V_\psi(s)$$

This decomposition allows the reward function $r_\psi(s,a)$ to be disentangled from the dynamics. We use separate neural networks for the reward $r_\psi(s,a) = \text{NN}_{\text{reward}}(s,a;\psi_r)$ and value function $V_\psi(s) = \text{NN}_{\text{value}}(s;\psi_v)$. The discriminator's loss includes a gradient penalty for stable training:

$$\begin{aligned}\mathcal{L}_D = &- \mathbb{E}_{(s_E,a_E)\sim\mathcal{D}_E}[\log D_\psi(s_E,a_E,s'_E)] \\ &- \mathbb{E}_{s_E\sim\mathcal{D}_E,a_\phi\sim\pi_\phi}[\log(1-D_\psi(s_E,a_\phi,s'_E))] \\ &+ \lambda\mathbb{E}_{(\hat{s},\hat{a},\hat{s}')}[(\|\nabla f_\psi(\hat{s},\hat{a},\hat{s}')\|_2-1)^2]\end{aligned}$$

where $(\hat{s},\hat{a},\hat{s}')$ are interpolated between expert and policy samples. For policy optimization, we use Implicit Q-Learning (IQL) to learn offline from the fixed dataset. The value function is learned via asymmetric expectile regression:

$$\mathcal{L}_V = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_2^\tau(Q_\omega(s,a) - V_\theta(s))\right]$$

where $L_2^\tau(u) = \begin{cases}\tau u^2 & \text{if } u \geq 0 \\ (1-\tau)u^2 & \text{if } u < 0\end{cases}$ and $\tau \in (0.5, 1)$ weights the upper tail of Q-values. The Q-function is trained using temporal difference learning:

$$\begin{aligned}y &= r_\psi(s,a) + \gamma V_\theta(s') \\ \mathcal{L}_Q &= \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[(Q_\omega(s,a) - y)^2\right]\end{aligned}$$

The policy is optimized via advantage-weighted regression:

$$\mathcal{L}_\pi = -\mathbb{E}_{s\sim\mathcal{D}}\left[\exp(\beta A(s,a))\log\pi_\phi(a|s)\right], \quad a\sim\pi_\phi(\cdot|s)$$

where $A(s,a) = Q_{\min}(s,a) - V_\theta(s)$ is the advantage estimate and β is an inverse temperature parameter. The policy is a Gaussian $\pi_\phi(a|s) = \mathcal{N}(\mu_\phi(s),\sigma_\phi(s)^2)$, using the reparameterization trick $a = \mu_\phi(s) + \sigma_\phi(s)\cdot\epsilon,\ \epsilon\sim\mathcal{N}(0,I)$

for gradient estimation. The complete optimization combines adversarial imitation learning with offline reinforcement learning:

$$\begin{aligned}\min_{\pi_\phi, Q_\omega, V_\theta} \max_{D_\psi} \quad & -\mathbb{E}_{\mathcal{D}_E}[\log D_\psi(s,a,s')] \\ & -\mathbb{E}_{s\sim\mathcal{D}_E, a\sim\pi_\phi}[\log(1-D_\psi(s,a,s'))] \\ & +\lambda\mathcal{L}_{\text{grad-pen}} + \mathcal{L}_V + \mathcal{L}_Q + \mathcal{L}_\pi\end{aligned}$$

This framework simultaneously learns a reward function that explains expert behavior, a value function that estimates long-term returns, and a policy that acts according to the learned reward. Training is entirely offline. For evaluation, the trained policy is deployed on held-out expert states, with performance measured by the Mean Squared Error between predicted and true expert actions and qualitative trajectory comparisons.

Supply Chain Optimization

Route Planning & Optimization We address chemical distribution logistics through route planning and optimization for multi-vehicle operations. Vehicles transport products from depots to customers, with each depot serving as both origin and destination. Vehicles depart fully loaded, deliver to clients, and may collect by-products or empty containers before returning within strict time windows and capacity constraints. Each customer is served exactly once. We propose a deep reinforcement learning framework based on Proximal Policy Optimization (PPO) for learning efficient routing strategies. The problem is formulated on a complete directed graph $G = (V, E)$, where $V = D \cup C$ comprises depots D and customers C, and E represents travel connections. Each depot $d \in D$ operates within time window $[e_d, l_d]$. Each customer $c \in C$ has delivery demand q_c, pickup quantity p_c, service time s_c, and time window $[e_c, l_c]$. Each edge $(i, j) \in E$ has travel time t_{ij} and distance d_{ij} from real-world routing data(Google Maps API). The heterogeneous fleet K consists of vehicles $k \in K$, each with capacity Q_k, start/end depots d_k^s, d_k^e, start time τ_k, maximum duration $T_k^{\max}$, variable cost c_k^{km}, and fixed cost c_k^{fixed}. The objective constructs feasible routes $\{R_1, \ldots, R_{|K|}\}$, where $R_k = (d_k^s, c_1, \ldots, c_m, d_k^e)$, minimizing total system cost:

$$Z = \sum_{k\in K}\left(c_k^{\text{fixed}} \cdot y_k + c_k^{\text{km}} \cdot \sum_{(i,j)\in R_k} d_{ij}\right),$$

where $y_k = 1$ if vehicle k is used, 0 otherwise. Constraints include: (i) capacity limits Q_k, (ii) time-window feasibility (allowing early arrival waiting), (iii) route duration limits $T_k^{\max}$, (iv) depot windows $[e_{d_k^e}, l_{d_k^e}]$, and (v) each customer visited exactly once. We model this as a Markov Decision Process where a PPO agent sequentially constructs routes. The *state* s_t encodes node features, vehicle attributes (load, time, location), and a mask excluding infeasible actions. The *action* a_t selects the next node to visit. The *reward function* penalizes travel distance and waiting:

$$r_t = -\left(d_{ij} \cdot c_k^{\text{km}} + \beta \cdot \text{Wait Time}\right),$$

where β weights the waiting penalty. The PPO agent uses an *Actor–Critic architecture* with a shared Transformer encoder for context-aware embeddings. The *Actor* produces masked action probabilities ensuring feasibility; the *Critic* estimates state-value $V(s_t)$. Policy updates use PPO's clipped objective:

$$L^{\text{CLIP}} = \mathbb{E}_t\left[\min\left(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right],$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio and A_t is the advantage estimate. The total loss combines value function and entropy terms:

$$L^{\text{TOTAL}} = L^{\text{CLIP}} - c_1 L^{VF} + c_2 H(\pi(\cdot|s_t)),$$

where $L^{VF} = \mathbb{E}_t\left[(V_\theta(s_t) - V_{\text{target}})^2\right]$ and V_{target} is the GAE-based return estimate. The agent iteratively refines its policy through gradient-based updates. Performance is evaluated against heuristic benchmarks (Nearest Neighbor, Savings, Random Insertion) using metrics including total cost, distance, fleet utilization, and unserved customers (Table 1).

Table 1: Performance Comparison of Vehicle Routing Problem Solvers for Chemical Distribution

Metric	PPO RL Agent	Nearest Neighbor	Savings (C&W)	Random Insertion
Cost ($)	904.77	997.02	1,287.07	1,589.04
Distance (km)	280.42	331.68	553.53	641.13
Vehicles	2	2	3	3
Unserved	1	1	0	0

Network & Route Optimization in Chemical Distribution Logistics We address the Network & Route Optimization problem for a multi-plant distribution system, formulated as a Multi-Depot Vehicle Routing Problem with Sourcing Costs (MDVRP-SC). The system consists of multiple production plants acting as depots, each operating a fleet of vehicles that deliver products to customers. Customer demands may be split into delivery jobs assignable to different plants. The optimization jointly determines (i) plant–customer assignments and (ii) vehicle routes that start and end at the corresponding plants. These decisions must respect key operational constraints: plant production capacities, vehicle load limits, customer time windows, maximum route durations, plant operating hours, and fixed unloading times at customer sites. The objective is to minimize total system cost, including both production costs at plants and transportation costs for deliveries. The system is represented by a complete directed graph $G = (V, E)$, where $V = P \cup C$ consists of production plants $P = \{p_1, ..., p_m\}$ and customers $C = \{c_1, ..., c_n\}$. Each plant $p \in P$ has production capacity Q_p (tons), unit production cost α_p (\$/ton), and operational time window $[e_p, l_p]$. Each customer $c \in C$ has demand d_c (tons), transfer time u_c (seconds), and delivery time window $[e_c, l_c]$. Each vehicle $k \in K$ has capacity q_k (tons), home plant $h_k \in P$, start time τ_k, maximum route duration $T_k^{\max}$, variable transportation cost β_k (\$/km), and fixed dispatch cost γ_k (\$). The integrated optimization involves

two coupled decisions: (i) the *sourcing decision* via binary variables $y_{p,c} = 1$ if plant p serves customer c, and (ii) the *routing decision* determining vehicle routes $\{R_1, ..., R_{|K|}\}$. The total cost is:

$$Z = \sum_{p \in P} \sum_{c \in C} \alpha_p \cdot d_c \cdot y_{p,c} + \sum_{k \in K} \left(\gamma_k \cdot \delta_k + \beta_k \cdot \sum_{(i,j) \in R_k} \text{dist}_{ij} \right),$$

where $\delta_k = 1$ if vehicle k is used and dist_{ij} is the travel distance between locations i and j. The objective function (Z) minimizes the sum of production costs (where to make the chemicals) and transportation costs (how to deliver them). Key constraints include:

$$\sum_{c \in C} d_c \cdot y_{p,c} \leq Q_p \quad \forall p \in P \quad \text{(Plant Capacity)},$$

$$\sum_{p \in P} y_{p,c} = 1 \quad \forall c \in C \quad \text{(Demand Satisfaction)},$$

along with vehicle capacity limits, hard time-window constraints $e_c \leq \text{start}_c \leq l_c$ for each customer c, where start_c is the service start time, and l_c = latest allowable service start time for customer c. If a vehicle arrives before (e_c), it must wait; if it cannot arrive by (l_c), the customer cannot be served by that vehicle. Route duration constraints ensuring that each vehicle k completes its route within maximum duration $T_k^{\max}$, where $\text{duration}(R_k) \leq T_k^{\max}$ accounts for total travel and service time. To manage the computational complexity of this NP-hard problem, we employ a two-stage decomposition framework. The first stage assigns customers to plants using a hybrid metaheuristic. It begins with greedy initialization based on composite cost $\text{Cost}_{p,c} = \alpha_p \cdot d_c + \text{dist}(p, c) \cdot \bar{\beta}$, where $\bar{\beta}$ is the average transportation cost rate. This is refined via local search using scoring function $\text{Score} = \sum_{p \in P} \text{Cost}_p + \lambda \cdot \text{Unserved}_p$, where λ is a penalty factor and Unserved_p denotes unserved customers at plant p, Cost_p is the production and estimated transportation costs for serving all customers assigned to that particular plant p. The second stage solves routing subproblems in parallel using Reinforcement Learning (RL) based on Proximal Policy Optimization (PPO). Each subproblem is a single-depot VRP modeled as a Markov Decision Process, where state s_t captures node features, vehicle status, and an action mask ensuring feasibility. Action a_t selects the next customer. The reward function is:

$$r_t = -\left(\text{dist}_{ij} \cdot \beta_k + \omega \cdot \text{wait\_time}\right),$$

where ω is a penalty coefficient and wait_time denotes idle waiting duration. The policy uses an Actor-Critic architecture with a shared Transformer encoder generating context-rich graph embeddings. The **Actor** produces probabilities over feasible actions via masked attention; the **Critic** estimates value function $V(s_t)$. Training minimizes the PPO clipped objective:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right],$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio, $\hat{A}_t$ is the advantage estimate, and ϵ is the clipping hyperparameter. The total loss is:

$$L^{TOTAL} = L^{CLIP} - c_1 L^{VF} + c_2 H(\pi(\cdot|s_t)),$$

where L^{VF} is the value function loss, H is policy entropy, and c_1, c_2 are weighting coefficients. Performance is benchmarked against a baseline Greedy heuristic using metrics including Total Cost Z, Vehicles Used, and Unserved Customers, demonstrating the robustness of the two-stage framework for large-scale chemical distribution logistics (see Table 2 and Figure 1).

Table 2: Performance comparison between Reinforcement Learning and Greedy heuristic solvers for the Multi-Depot Vehicle Routing Problem. Both algorithms achieved identical results, finding the optimal solution that minimizes total costs while serving all construction sites within operational constraints.

Metric	PPO_RL	GREEDY
Total Network Cost	$19,470.72	$19,470.72
Production Cost	$11,605.00	$11,605.00
Transport Cost	$7,865.72	$7,865.72
Trucks Dispatched	10	10
Unserved Sites	0	0

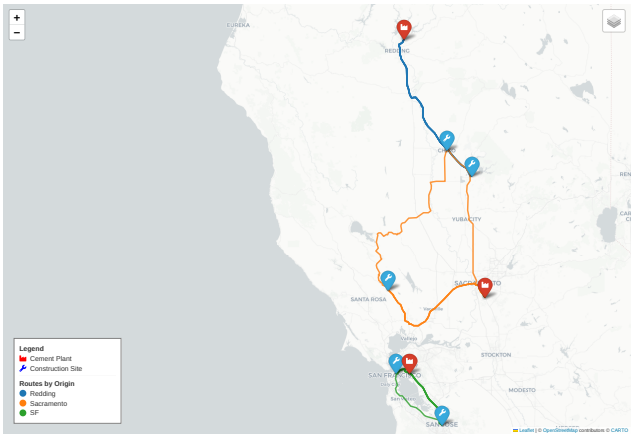

Figure 1: Geographic visualization of the optimized cement distribution network showing plants (red), construction sites (blue), and color-coded routes by originating facility.

Logistics Optimization

Single-Robot Multi-Order Picking and Placement The proposed framework models single-robot multi-order picking and placement in chemical plants as a Hierarchical Reinforcement Learning (HRL) task. The decision-making process is decomposed into two levels: a high-level policy that prioritizes material transfer requests based on processing

priorities and payload capacity limitations, and a low-level policy that executes navigation, pickup, and delivery actions while avoiding obstacles. This temporal abstraction enables the agent to make macro-decisions while executing micro-actions to achieve sub-goals, allowing goal-conditioned behaviors that generalize across varying process complexities. The framework jointly optimizes both levels to maximize operational efficiency—measured by throughput and process completion rates—while respecting payload capacity constraints, processing priorities, and obstacle avoidance requirements. The robotic picking and placement task is modeled as a Hierarchical Markov Decision Process (HMDP) defined by $(\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma))$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $P(s'|s,a)$ the transition dynamics, $R(s,a)$ the reward function, and $\gamma \in (0,1)$ the discount factor. The warehouse is a two-dimensional grid with obstacles, items, and packing stations. Each item $i \in \mathcal{I}$ is located at $l_i = (x_i, y_i)$ with weight w_i, and each order $o \in \mathcal{O}$ contains a set of items $\mathcal{I}_o$, a packing station l_o^p, and a priority $p_o \in [0,1]$. A robot located at $l_r = (x_r, y_r)$ has a maximum carrying capacity C_r and a current load $W_r \leq C_r$. The hierarchy separates high-level order selection from low-level motion and manipulation. The high-level state $s_r^{(H)}$ includes order progress, priority, estimated delivery cost, and current load:

$$s_r^{(H)} = \Big[(c_o, p_o, \hat{d}_o)_{\forall o \in \mathcal{O}}, \frac{W_r}{C_r}\Big],$$

where c_o is the order completion ratio, p_o the normalized priority, and $\hat{d}_o$ the estimated normalized distance. The low-level state $s_r^{(L)}$ captures local context, including the robot's position relative to items and stations, normalized load, and number of items currently carried n_c:

$$s_r^{(L)} = \Big[\frac{l_r}{L}, \frac{(l_i - l_r)}{L}, \frac{(l_o^p - l_r)}{L}, \frac{W_r}{C_r}, n_c\Big],$$

where L denotes the grid size. The high-level action space determines which order to execute next, while the low-level action space $\mathcal{A}^{(L)} = \{\text{UP}, \text{DOWN}, \text{LEFT}, \text{RIGHT}, \text{PICK}, \text{DROP}\}$ defines the robot's motion and manipulation primitives. The reward function combines multiple components:

$$R(s,a) = r_{\text{step}} + r_{\text{move}} + r_{\text{pick}} + r_{\text{drop}} + r_{\text{completion}},$$

where $r_{\text{step}} < 0$ penalizes longer task durations, r_{move} rewards movement toward the target and penalizes invalid or backward moves, r_{pick} and r_{drop} reward successful item handling, and $r_{\text{completion}}$ provides a large terminal reward when an order is completed. In capacity-constrained scenarios, the robot can only pick an item if the total weight after pickup remains within its capacity ($W_r + w_i \leq C_r$). When multiple items are delivered in a single trip, the drop reward scales proportionally to the number of items successfully delivered during that delivery cycle. The objective is to learn hierarchical policies $\pi_H(a_H|s_H)$ and $\pi_L(a_L|s_L, g_H)$ that jointly maximize the expected cumulative reward:

$$J(\pi_H, \pi_L) = \mathbb{E}\Big[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)\Big].$$

Both controllers employ Deep Q-Learning with prioritized experience replay and soft target updates. The high-level Q-network $Q^{(H)}(s^{(H)}, a^{(H)}; \theta_H)$ selects orders, and the low-level Q-network $Q^{(L)}(s^{(L)}, a^{(L)}; \theta_L)$ handles detailed control. Each minimizes the temporal-difference loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')}\Big[\big(r + \gamma \max_{a'} Q_{\text{target}}(s', a'; \bar{\theta}) - Q(s,a;\theta)\big)^2\Big],$$

where the target parameters are updated via $\bar{\theta} \leftarrow \tau\theta + (1-\tau)\bar{\theta}$. Prioritized replay samples transitions proportional to TD-error magnitude (α), with importance sampling weights (β) compensating for bias. Exploration follows an epsilon-greedy policy decay, while gradient clipping and periodic target updates stabilize training. Performance is evaluated across episodes using metrics such as order completion rate, average steps per order, total distance traveled, success rate, and cumulative reward. In unconstrained environments, the agent learns efficient sequential routing and order handling. In capacity-constrained environments, the policy learns to plan item pickups based on individual weights, ensuring the total carried load never exceeds the robot's capacity. The resulting framework achieves balanced and adaptive control between strategic order assignment and precise motion execution for efficient warehouse operations.

Multi-Robot Multi-Order Picking and Placement The single-robot formulation is extended to a *multi-robot, multi-order* setting, where multiple robots collaborate within a shared grid-based warehouse containing static obstacles, items, and packing stations. Each robot operates under its own payload capacity constraint and interacts with others through spatial coordination and collision avoidance. The overall objective remains to maximize throughput and order completion rates while ensuring safe and efficient cooperative operation. Unlike the single-robot case, the environment now maintains a set of robots $\mathcal{R} = \{r_1, r_2, \ldots, r_K\}$, each characterized by a position $l_{r_k} = (x_{r_k}, y_{r_k})$, a capacity C_{r_k}, and a current load $W_{r_k} \leq C_{r_k}$. The state and action spaces are extended to encode multi-agent interaction, enabling concurrent decision-making and decentralized coordination. The system is modeled as a *Multi-Agent Hierarchical Markov Decision Process (MA-HMDP)*:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma),$$

where $\mathcal{S}$ and $\mathcal{A}$ represent the *joint* state and action spaces over all robots, $P(s'|s,a)$ captures multi-agent transition dynamics, and $R(s,a)$ defines the cooperative global reward. Each robot's high-level state incorporates not only its own progress and load but also the relative states of other robots:

$$s_{r_k}^{(H)} = \Big[(p_o, c_o, \hat{d}_{o,i})_{\forall(o,i)},\ \frac{W_{r_k}}{C_{r_k}},\ \big(\frac{l_{r_j}}{L}\big)_{\forall j \neq k}\Big],$$

allowing spatial awareness and coordination for collision avoidance and distributed task selection. The low-level state further includes the positions of neighboring robots:

$$s_{r_k}^{(L)} = \Big[\frac{l_{r_k}}{L},\ \frac{(l_i - l_{r_k})}{L},\ \frac{(l_o^p - l_{r_k})}{L},\ \frac{W_{r_k}}{C_{r_k}},\ \big(\frac{l_{r_j} - l_{r_k}}{L}\big)_{\forall j \neq k}\Big],$$

enabling each robot to plan collision-free paths while navigating toward its assigned targets. All robots act concurrently at each time step, forming a joint action set $\mathcal{A} = \{a_{r_1}, a_{r_2}, \ldots, a_{r_K}\}$, where each $a_{r_k} \in \mathcal{A}^{(L)} = \{\text{UP}, \text{DOWN}, \text{LEFT}, \text{RIGHT}, \text{PICK}, \text{DROP}\}$. The environment enforces mutual exclusion such that no two robots can occupy the same grid cell, and collisions with obstacles or other robots incur penalties. The global reward aggregates individual contributions from all robots:

$$R(s,a) = \sum_{r_k \in \mathcal{R}} \Big(r_{\text{step}} + r_{\text{move}} + r_{\text{pick}} + r_{\text{drop}} + r_{\text{completion}} + r_{\text{collision}} \Big),$$

where $r_{\text{collision}} < 0$ penalizes robot–robot and robot–obstacle collisions. In the capacity-constrained setting, each robot can pick an item only if $(W_{r_k} + w_i \leq C_{r_k})$; otherwise, a penalty is applied, encouraging load-feasible item selection and implicit cooperation among robots through task redistribution. All robots share parameters for both the high-level and low-level Q-networks, enabling decentralized execution with centralized training. The learning objective remains:

$$J(\pi_H, \pi_L) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)\right],$$

but gradient updates are performed using experience tuples aggregated from all robots' interactions, promoting cooperative learning through shared replay buffers. Performance is evaluated through multi-agent metrics such as mean order completion rate, total travel distance, collision frequency, and cumulative reward. In unconstrained environments, robots learn efficient division of labor and spatial coordination, while in capacity-constrained scenarios, they develop load-aware task allocation strategies that maintain balanced utilization and collision-free cooperation.

Bin Packing Optimization for Chemical Plant Logistics In large-scale chemical plant logistics and warehouse operations, efficient packing and shipment planning are vital for minimizing handling costs, maximizing space utilization, and maintaining safety compliance. The task involves selecting suitable containers and arranging diverse items under strict geometric, weight, and stacking constraints. To address this challenge, we develop a Reinforcement Learning (RL)–based bin packing framework that autonomously learns efficient packing strategies. The framework models the problem as a Markov Decision Process (MDP) and employs a Dueling Deep Q-Network (DQN) with Prioritized Experience Replay (PER) to optimize packing efficiency and cost under realistic industrial conditions. The framework seeks to balance cost, utilization, and constraint compliance through a multi-objective reward formulation. The overall objective is expressed as

$$\max_{\pi} J(\pi) = \mathbb{E}_{\pi}[w_1(-C_{\text{total}}) + w_2 U_{\text{avg}} - w_3 N_b - w_4 \mathcal{V}_{\text{constraint}}],$$

where C_{total} denotes the total logistics cost, U_{avg} represents the mean container utilization, N_b is the number of boxes or tanks used, and $\mathcal{V}_{\text{constraint}}$ measures the cumulative constraint violation. The coefficients w_1, w_2, w_3, w_4 are user-defined weights that adjust the trade-off between cost efficiency, space utilization, container minimization, and safety compliance. The environment is modeled as a Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action set, $\mathcal{T}$ the transition function, $\mathcal{R}$ the reward function, and γ the discount factor. Each state $s_t \in \mathcal{S}$ encodes the condition of all available containers, the packing configuration, and the items yet to be placed. The state vector is represented as

$$s_t = [x_{\text{avail}},\, u_i,\, \tfrac{w_i}{w_i^{\max}},\, \text{fit}_i,\, \tfrac{c_i}{c_{\text{div}}},\, \tfrac{V_i}{V_{\text{div}}}],$$

where x_{avail} indicates whether an item remains unpacked, u_i is the utilization ratio of container i, and $\frac{w_i}{w_i^{\max}}$ expresses the packed-to-maximum weight ratio. The binary variable $\text{fit}_i \in \{0, 1\}$ denotes if the next item can fit in container i, while c_i and V_i represent its cost and volume, normalized by divisors c_{div} and V_{div} for scale stability. The action $a_t \in \mathcal{A}$ selects an item for placement into an existing or new container, and the transition $\mathcal{T}(s_t, a_t)$ updates the state according to geometric fit and residual space, giving $s_{t+1} = f(s_t, a_t)$. The immediate reward r_t balances cost and utilization:

$$r_t = \begin{cases} -\lambda_c C_b, & \text{if a new container is opened,} \\ \lambda_v \frac{V_i}{V_{\text{div}}} + \lambda_z \left(1 - \frac{z_i}{H_b}\right), & \text{if an item is placed successfully,} \\ -\lambda_f, & \text{otherwise.} \end{cases}$$

Here, C_b is the container cost, V_i the item volume, and z_i its placement height within a container of height H_b. The coefficients $\lambda_c, \lambda_v, \lambda_z, \lambda_f$ regulate penalties for cost, spatial efficiency, height placement, and failures. The cumulative episodic reward sums all r_t values and applies terminal bonuses when utilization and cost targets are met. Physical feasibility is ensured by the constraints

$$\sum_j w_j \leq W_b^{\max}, \qquad \sum_j V_j \leq V_b,$$

which prevent weight and volume overflow. For fragile items, if an object below is fragile, no heavier object may be placed above it, expressed as

$$\text{if fragile(below)} = 1 \Rightarrow w_{\text{above}} = 0.$$

Rotations are allowed only for items with flexible orientations, i.e., $r_i \in \text{Rotations}(d_i)$ when the orientation flag is unset. The overall optimization objective is

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} \gamma^t r_t\right], \quad \text{subject to } \mathcal{C}(s_t, a_t) = 0,$$

where π^* is the optimal packing policy and $\mathcal{C}$ encodes geometric and safety constraints. The agent employs a Dueling Deep Q-Network (DQN) that decomposes the Q-value into state value $V(s_t)$ and advantage $A(s_t, a_t)$:

$$Q(s_t, a_t) = V(s_t) + \Big(A(s_t, a_t) - \tfrac{1}{|\mathcal{A}|}\sum_{a'} A(s_t, a')\Big).$$

The target update is

$$y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-),$$

and the loss minimized is a weighted Huber loss,

$$\mathcal{L} = \mathbb{E}[w_i \cdot \text{Huber}(Q(s, a; \theta) - y_t)]\,,$$

where w_i is the importance-sampling weight correcting prioritized replay bias. Gradient clipping and scheduled learning-rate decay stabilize convergence. After training, the learned policy π^* is evaluated without exploration noise. Performance is compared with heuristic baselines commonly used in logistics: First Fit Decreasing (FFD), Best Fit Decreasing (BFD), and a Genetic Algorithm (GA). The FFD method sequentially assigns items to the first feasible container, while BFD minimizes residual volume after placement. The GA baseline evolves item orderings using a fitness function

$$f = C_{\text{total}} + \alpha(1 - U_{\text{avg}}) + \beta N_{\text{unpacked}},$$

where C_{total} is the total cost, U_{avg} the average utilization, and N_{unpacked} the count of unplaced items. A solution is accepted if

$$N_b \le N_b^{\max}, \quad C_{\text{total}} \in [C_{\min}, C_{\max}], \quad U_{\text{avg}} \ge U_{\min}.$$

Empirical evaluation shows that the RL agent achieves higher utilization and lower cost than heuristic methods by learning spatially aware and cost-efficient packing strategies that generalize across diverse container and item configurations in chemical logistics operations.

Dynamic Pricing and Revenue Optimization for Process Plants In chemical production plants, determining optimal product pricing is a complex decision-making task influenced by volatile market demand, fluctuating raw material and energy costs, and evolving inventory conditions. Conventional static or cost-plus pricing strategies remain insensitive to market variability, motivating a reinforcement learning (RL) formulation that learns adaptive, data-driven pricing decisions. The pricing process is formulated as a sequential decision-making problem using Proximal Policy Optimization (PPO) to learn policies that maximize long-term profit while ensuring operational and inventory compliance. The policy $\pi_\theta(a_t|s_t)$ determines the optimal daily selling price a_t given the current system state s_t, balancing short-term profitability with long-term operational stability. The PPO-based agent exhibits dynamic, context-aware behavior: when production costs rise, it increases prices to preserve profit margins, while lower costs incentivize price reductions to stimulate demand. Under favorable hedging conditions—when futures prices fall below spot feedstock costs—the agent maintains or slightly decreases prices to expand market share, whereas unfavorable conditions prompt higher prices to safeguard profitability. The pricing process is modeled as a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the continuous state space encoding operational and market variables (forecasted demand, inventory, production costs, feedstock and energy prices, temporal features), $\mathcal{A}$ is the discrete action space of feasible price levels, P represents transition dynamics, R is the reward function, and $\gamma \in (0, 1)$ is the discount factor for future rewards. The action space $\mathcal{A} = \{p_0, p_1, \ldots, p_K\}$ represents discrete price levels centered around a nominal base price p_{base} with markup factors μ_k: $p_k = p_{\text{base}}(1 + \mu_k)$. The policy $\pi_\theta(a_t|s_t)$ determines the optimal daily selling price a_t given the current system state s_t. The immediate reward is the normalized profit margin:

$$r_t = \frac{1}{\eta}\big(R_t - C_t - P_t^{\text{inv}}\big),$$

where R_t is total revenue, C_t is total cost, P_t^{inv} is an inventory penalty, and η is a scaling factor for numerical stability. Daily revenue combines spot and contract sales:

$$R_t = Q_t^{\text{spot}} p_t + Q_t^{\text{contract}} p_c,$$

where $p_c = p_{\text{base}}(1 - \delta)$ is the discounted contract price with discount rate δ. Spot demand follows a price-elastic relationship:

$$Q_t^{\text{spot}} = (D_t(1 - \rho_c)) \left(\frac{p_t}{p_{\text{base}}}\right)^{-\epsilon},$$

where D_t is predicted market demand, $\rho_c \in [0, 1]$ is the contractual demand fraction, and $\epsilon > 0$ is the price elasticity coefficient (higher prices exponentially reduce demand). Total cost combines material, catalyst, energy, and hedging costs:

$$C_t = Q_t^{\text{total}}(c_t^{\text{raw}} + c^{\text{add}} + c_t^{\text{energy}}) - H_t + Q_t^{\text{total}} c^{\text{hedge}},$$

where $Q_t^{\text{total}} = Q_t^{\text{spot}} + Q_t^{\text{contract}}$ is total sales volume. The hedging benefit $H_t = (c_t^{\text{raw}} - f_t^{\text{futures}})\psi Q_t^{\text{total}}$ is realized when futures price f_t^{futures} is below the raw-material price c_t^{raw}, with $\psi \in [0, 1]$ representing hedge effectiveness. Inventory evolves via a mass-balance equation:

$$I_{t+1} = I_t + P^{\text{rate}} - Q_t^{\text{total}},$$

where P^{rate} is the fixed daily production rate. Inventory deviations beyond operational thresholds incur penalties:

$$P_t^{\text{inv}} = \begin{cases} \alpha_{\text{low}}(I_{\text{low}} - I_t), & \text{if } I_t < I_{\text{low}} \text{ (stock-out risk)}, \\ \alpha_{\text{high}}(I_t - I_{\max}\phi), & \text{if } I_t > I_{\max}\phi \text{ (storage constraint)}, \\ 0, & \text{otherwise}, \end{cases}$$

where I_{low} is minimum safety stock, $I_{\max}$ is storage capacity, $\phi \in (0, 1)$ controls soft upper limits, and $\alpha_{\text{low}}, \alpha_{\text{high}} > 0$ are penalty weights. The learning objective maximizes expected cumulative discounted reward:

$$J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right].$$

The training pipeline begins with feature engineering: cost, demand, and energy time series are aggregated across temporal windows, computing statistical descriptors (mean, variance, extrema, polynomial trends) over multiple lags. A gradient boosting regression model $\hat{D}_t = f_\phi(s_t)$ forecasts short-term demand. The RL agent employs PPO with an actor-critic architecture. The actor outputs a categorical distribution over price levels; the critic estimates the state-value function $V_\theta(s_t)$. PPO maximizes the clipped surrogate objective:

$$L^{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min \left(\rho_t(\theta)\hat{A}_t,\ \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right],$$

where $\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio and $\hat{A}_t$ is the generalized advantage estimate computed via temporal-difference updates with decay parameter λ. The critic minimizes value loss $L_V = \frac{1}{2}\|V_\theta(s_t) - \hat{R}_t\|^2$. Gradient clipping ensures numerical stability during training across multiple episodes with randomized starting indices. The PPO-based agent exhibits dynamic, context-aware behavior: when production costs rise, it increases prices to preserve margins; lower costs prompt price reductions to stimulate demand. Under favorable hedging (futures below spot costs), the agent maintains or reduces prices to expand market share; unfavorable conditions prompt higher prices to safeguard profitability. For evaluation, three strategies—Static, Cost-Plus, and PPO-based Adaptive Pricing—are benchmarked over a full operational horizon using cumulative profit, average daily margin, and average realized price. The PPO-based agent learns to dynamically adjust pricing according to market elasticity and inventory conditions, lowering prices under oversupply and raising them during scarcity. This adaptive policy achieves superior long-term profitability and smoother inventory utilization compared to static and heuristic benchmarks.

Inventory Optimization System for Chemical Manufacturing Industrial manufacturing plants must determine optimal production rates that balance inventory holding costs, demand fulfillment, and operational expenses under uncertainties from market fluctuations and production variability. Maintaining equilibrium between production, consumption, and storage constitutes a complex control problem. To address this challenge, the task is formulated as a sequential decision-making problem where the objective is to learn adaptive control policies that minimize long-term operational costs while ensuring compliance with production and inventory constraints. A reinforcement learning (RL) framework is employed, integrating a data-driven world model that captures production-demand dynamics with a policy optimization agent that learns optimal production rate decisions. This problem is formulated as a Markov Decision Process (MDP) defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ denotes the state space comprising inventory levels, production rates, and demand features, $\mathcal{A}$ represents the discrete action space of feasible production rate setpoints, $P(s_{t+1}|s_t, a_t)$ captures transition dynamics, and $R(s_t, a_t)$ defines the immediate reward. A reinforcement learning framework integrates a data-driven world model that captures production-demand dynamics with a policy optimization agent to enable safe training in simulation before real-world deployment. The objective is to derive an optimal control policy $\pi_\theta(a_t|s_t)$, parameterized by θ, that determines production decision a_t given system state s_t to maximize the long-term expected return:

$$J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right],$$

where $r(s_t, a_t)$ represents the immediate reward (negative of total operating cost), $\gamma \in [0, 1)$ is the discount factor balancing short- and long-term objectives, and T is the planning horizon. The reward function penalizes four cost components:

$$r_t = -(c_h I_t + c_u U_t + c_p P_t + c_r R_t),$$

where I_t, U_t, P_t, and R_t correspond to inventory holding, unmet demand, production, and rate-change costs, with $c_h, c_u, c_p, c_r > 0$ representing their respective cost coefficients. The process dynamics evolve according to $s_{t+1} = f(s_t, a_t; \omega) + \epsilon_t$, where $f(\cdot)$ denotes the underlying system evolution function parameterized by ω and ϵ_t represents stochastic process noise. The training framework adopts a hybrid model-based reinforcement learning approach, combining supervised learning for process prediction with policy optimization via Proximal Policy Optimization (PPO). The predictive world model approximates the process dynamics using a function approximator $\hat{f}_\omega$ trained on historical observations. Given an input feature vector $\mathbf{x}_t \in \mathbb{R}^n$ constructed from lagged demand, production rates, catalyst health, and time-dependent features, the model predicts the next system state y_t by minimizing mean-squared error:

$$\omega^* = \arg\min_\omega \frac{1}{N} \sum_{t=1}^{N} \|y_t - \hat{f}_\omega(\mathbf{x}_t)\|^2.$$

This trained model enables the creation of a simulated environment in which the PPO agent learns robust control policies efficiently without requiring extensive real-world interactions. The PPO algorithm refines the control policy by constraining policy updates for training stability through its clipped surrogate objective:

$$L^{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right],$$

where $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$ is the probability ratio between the updated and previous policies, and ϵ controls the extent of policy deviation. The advantage term $\hat{A}_t$ is computed using Generalized Advantage Estimation (GAE):

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}, \quad \text{where} \quad \delta_t = r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t),$$

with $V_\theta(s_t)$ being the critic's value function and $\lambda \in [0, 1]$ balancing bias and variance. The total loss

combines actor, critic, and entropy components to balance exploitation, value estimation accuracy, and exploration. The evaluation phase benchmarks multiple decision strategies—including reactive, heuristic, and reinforcement learning-based agents—under consistent simulated demand conditions across a simulation horizon. Each policy's cumulative performance is evaluated using the total cost function:

$$C_{\text{total}} = \sum_{t=0}^{T} (c_h I_t + c_u U_t + c_p P_t + c_r R_t),$$

with comparisons made in terms of total cost, cost breakdown, and stability of control decisions, demonstrating that the optimized control policy achieves reduced operating costs and improved dynamic stability compared to baseline strategies.

Risk-Aware Hedging and Portfolio Optimization In large-scale chemical process plants, profitability is strongly influenced by fluctuations in feedstock costs, energy prices, and product market values. Chemical plants rely on volatile commodity inputs and sell products in fluctuating markets. These price variations create significant financial risk, where sudden market changes can erode profit margins despite stable operations. To mitigate this risk, plants employ hedging strategies—securing future feedstock prices through forward contracts (private agreements to buy or sell at fixed prices on a future date) or futures contracts (standardized, exchange-traded versions with daily settlement to reduce counterparty risk)—and portfolio optimization, which allocates production capacity among multiple products to maximize expected profit while maintaining acceptable risk levels. Together, hedging and portfolio optimization form a dynamic financial–operational decision-making framework that couples production planning with market uncertainty management. The problem is formulated as a Markov Decision Process (MDP) defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space representing market features (commodity prices, trends, volatility), $\mathcal{A}$ is the action space encoding hedging ratios and portfolio allocations, $P(s_{t+1}|s_t, a_t)$ represents transition dynamics, $R(s_t, a_t)$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor. At each time step t, the agent observes state s_t, selects action $a_t \sim \pi_\theta(a_t|s_t)$ from a policy parameterized by θ, receives reward $R(s_t, a_t)$, and transitions to s_{t+1}. The objective is to learn an optimal policy π_θ^* maximizing the expected cumulative risk-adjusted profit:

$$J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t \left(\frac{\Pi_t}{\alpha} - \lambda \frac{\sigma_t}{\alpha}\right)\right],$$

where Π_t is instantaneous profit, σ_t is the rolling standard deviation of profits capturing volatility, λ is the risk-aversion coefficient, and α is a normalization constant. The first term encourages profit maximization while the second penalizes volatility. Instantaneous profit is

$$\Pi_t = \sum_p Q_{p,t} P_{p,t} - \sum_i C_{i,t} Q_{i,t} + H_t - F,$$

where $Q_{p,t}$ and $P_{p,t}$ are production quantity and price of product p, $C_{i,t}$ and $Q_{i,t}$ are feedstock prices and consumption of input i, H_t is hedge gain or loss, and F is fixed operating costs. A Proximal Policy Optimization (PPO) algorithm learns this optimal policy by iteratively updating parameters using clipped surrogate objectives for stable learning. The PPO objective is

$$L^{\text{PPO}}(\theta) = \mathbb{E}_t\Big[\min\Big(r_t(\theta)\,\hat{A}_t,\ \text{clip}\big(r_t(\theta),\,1-\epsilon,\,1+\epsilon\big)\hat{A}_t\Big) - c_v\left(V_\theta(s_t) - \hat{R}_t\right)^2 + c_e\,\mathcal{H}\big(\pi_\theta(\cdot|s_t)\big)\Big], \tag{1}$$

where $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$ is the probability ratio between new and old policies, $\hat{A}_t$ is the advantage estimate computed via Generalized Advantage Estimation (GAE), $V_\theta(s_t)$ is the critic's value estimate, $\hat{R}_t$ is the empirical return, and $\mathcal{H}$ is the policy entropy weighted by c_e for exploration. The clipping parameter ϵ prevents destabilizing policy updates. During training, the agent interacts with simulated market environments generated from historical data or Monte Carlo scenarios, learning to map price trajectories into optimal decisions. Feature engineering transforms raw market data into lagged and statistical representations—rolling means, variances, and polynomial trends—capturing temporal dependencies. For evaluation, the trained agent is benchmarked against baseline strategies under out-of-sample scenarios. Performance is assessed using expected total profit, standard deviation of returns, Value-at-Risk (VaR), and Profit-at-Risk (PaR):

$$\text{VaR}_\alpha = \inf\{x \mid P(\Pi < x) \geq \alpha\}, \qquad \text{PaR}_\alpha = \mathbb{E}[\Pi] - \text{VaR}_\alpha,$$

where VaR measures the minimum loss at confidence level α and PaR quantifies expected shortfall beyond VaR. A superior policy achieves higher average profit with lower volatility and tighter tail risk, demonstrating robust performance under stochastic market dynamics through adaptive, risk-aware control.

Chemical Industrial Demand Forecasting and Production Planning Industrial manufacturing requires integrated demand forecasting and production planning to determine optimal daily production rates under uncertainty. We formulate this as a Markov Decision Process and develop a Proximal Policy Optimization (PPO) agent that maximizes long-term profitability while simultaneously managing inventory, maintaining product quality specifications, ensuring environmental compliance below regulatory thresholds, and stabilizing operations by minimizing production rate changes. The system addresses fluctuating market demand, variable raw material and energy costs, maintenance schedules, and stochastic process dynamics affecting quality and environmental performance. The production planning task is modeled as a Markov Decision Process (MDP) defined by

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma),$$

where $\mathcal{S}$ denotes the system state (e.g., production rate, inventory level, energy use, product quality, and emissions), $\mathcal{A}$

represents feasible production setpoints, $P(s_{t+1}|s_t, a_t)$ captures stochastic process transitions, $R(s_t, a_t)$ is the immediate reward, and $\gamma \in [0, 1)$ is the discount factor weighting long-term profitability over short-term gain. The objective is to learn an optimal policy $\pi^*(a_t|s_t)$ that maximizes the expected cumulative discounted reward:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t) \right].$$

The reward function jointly optimizes economic performance and operational compliance:

$$R(s_t, a_t) = \Pi(s_t, a_t) - \mathcal{P}_{\text{quality}} - \mathcal{P}_{\text{emissions}} - \mathcal{P}_{\text{inventory}} - \mathcal{P}_{\text{stability}},$$

where $\Pi(s_t, a_t)$ denotes net profit (revenue minus feedstock, catalyst, energy, and storage costs), and the penalty terms $\mathcal{P}_{\text{quality}}$, $\mathcal{P}_{\text{emissions}}$, $\mathcal{P}_{\text{inventory}}$, and $\mathcal{P}_{\text{stability}}$ represent losses due to quality deviations, emission violations, inventory imbalance, and excessive production-rate fluctuations, respectively. The training process is divided into two tightly coupled phases: supervised forecasting and reinforcement learning–based optimization. The forecasting model $\hat{f}(\cdot)$ approximates multi-step-ahead predictions of demand, inventory, product quality, and emissions using regression over engineered time-series features, thus serving as a differentiable surrogate environment. The RL agent interacts with this surrogate to learn a policy $\pi_\theta(a_t|s_t)$ that generalizes across dynamic operating conditions. Policy optimization is performed using Proximal Policy Optimization (PPO), an actor–critic algorithm that maximizes a clipped surrogate objective:

$$L^{\text{PPO}}(\theta) = \mathbb{E}_t \Big[\min \Big(r_t(\theta)\hat{A}_t,\ \text{clip}\big(r_t(\theta),\, 1-\epsilon,\, 1+\epsilon\big)\hat{A}_t \Big) - c_v \big(V_\phi(s_t) - \hat{R}_t\big)^2 + c_e\, \mathcal{H}(\pi_\theta(\cdot|s_t)) \Big].$$

where $r_t(\theta)$ is the ratio between new and old policy probabilities, $\hat{A}_t$ is the advantage estimate from Generalized Advantage Estimation (GAE), and the last two terms represent value-function regularization and entropy maximization to balance learning stability and exploration. During training, the agent iteratively samples trajectories from the simulated environment, computes discounted returns and advantages, and updates the actor–critic parameters through mini-batch gradient ascent. Normalization and gradient clipping are applied to ensure numerical stability and prevent policy divergence. The surrogate forecasting model provides rapid feedback for evaluating the long-term impact of production decisions, enabling efficient off-policy learning without interacting with the real plant. The trained policy is benchmarked against baseline strategies such as static and reactive controllers using metrics including cumulative profit, on-spec production rate, environmental violations, and operational smoothness. Superiority is defined by consistently achieving higher profitability while maintaining compliance, demonstrating that the integrated forecasting–reinforcement approach enables adaptive and sustainable decision-making for industrial systems under uncertainty.

References