

A Reinforcement-Learning-Tuned Hybrid GA– PSO Algorithm for Continuous-Time Crude Oil Scheduling with Quality and Logistics Constraints

BOUKELKOUL ZAKARIA, PR. MOUSS KINZA NADIA

Abstract: *Crude oil scheduling couples a combinatorial assignment problem (which vessel unloads into which tank) with continuous-time decisions (when) and nonconvex quality constraints (bilinear blending of sulfur content), making it a genuinely hard mixed-integer nonlinear program. This paper formulates a continuous-time, event-driven version of the problem — vessel-to-tank assignment, unloading start times, tank inventory and blend-quality dynamics, CDU feed-quality bounds, demurrage, and a melt-point-driven pipeline/tank flush cost — and proposes a hybrid solution algorithm that nests a short particle swarm optimization (PSO) local search inside a genetic algorithm (GA) outer loop, with a tabular Q-learning agent that adapts the GA mutation rate and PSO inertia online based on search stagnation and population diversity. The algorithm, two single-method baselines (standalone GA, standalone PSO), and a greedy heuristic are implemented and compared on 20 independently generated synthetic benchmark instances at three scales (6, 12, and 20 vessels). Across all three scales the hybrid algorithm achieves the lowest mean cost and the highest schedule-feasibility rate of the four methods, with the advantage confirmed by the Wilcoxon signed-rank test against each baseline and by a Friedman test across all four methods ($p < 10^{-6}$ in every case). A Taguchi-style sensitivity experiment over population size and the number of PSO-refined elites per generation found no statistically significant main effects in the tested range, indicating the hybrid algorithm's performance is reasonably robust to these two hyperparameters. All reported figures come from code that was actually run for this manuscript; none are assumed or illustrative. What remains for future work is validation against real refinery operating data, which this manuscript does not claim to have performed.*

Keywords: Crude Oil Scheduling, Hybrid Metaheuristics, Genetic Algorithm, Particle Swarm Optimization, Reinforcement Learning, Continuous-Time Minlp.

Introduction

1.1 Background on crude oil scheduling

Scheduling in crude oil refining coordinates the supply chain from crude intake to finished product delivery: unloading, storage, blending, and processing within the refinery. Market fluctuations and operational constraints add complexity, so efficient scheduling helps cut costs and increase profits. Traditional methods often use discrete-time models, but continuous-time mixed-integer nonlinear programming (MINLP) formulations are increasingly used instead, since they let events occur at any point in the schedule, reducing the number of binary variables and improving real-world fidelity (Reddy, Karimi, & Srinivasan, 2004).

Quality constraints heavily influence scheduling, since crude blends must meet downstream sulfur and viscosity specifications; the mathematical models needed to capture this involve bilinear terms and are genuinely hard to solve to optimality (Li, Li, Karimi, & Srinivasan, 2007). Logistics pose further challenges, especially for high-melting-point crude, which must stay fluid during transport and requires pipeline flushing between incompatible crude types (Yao et al., 2026).

1.2 Research gap and contribution

A large body of work addresses pieces of this problem — continuous-time formulations (Reddy et al., 2004; Zimberg, Ferreira, & Camponogara, 2023; Li, Misener, & Floudas, 2011), genetic-algorithm and swarm-based heuristics for crude scheduling and related pipeline problems (Goudarzi, Maleki, & Niroomand, 2021; Liu, Ren, Shang, Zhao, & Wang, 2025), and reinforcement-learning-based parameter adaptation for related pipeline-operation problems (Jian, Xu, Liu, Zhou, & Yi, 2026[†]). What is comparatively rare is a single algorithm that (a) handles the discrete assignment and continuous timing decisions together, (b) is hybridized rather than standalone (nesting a continuous local search inside a discrete outer search), and (c) adapts its own search parameters online via reinforcement learning rather than via offline tuning, with the resulting algorithm actually implemented and benchmarked rather than only described.

This paper's contribution is therefore:

1. A continuous-time, event-driven MINLP-inspired formulation of vessel-to-tank assignment and unloading-time scheduling with bilinear blend-quality dynamics, CDU feed-quality bounds, demurrage, and melt-point-driven flush costs (Section 3).
2. A hybrid GA–PSO algorithm with a tabular reinforcement-learning layer for online parameter control, fully specified down to the operator level (Section 4).
3. An actual implementation of this algorithm, two single-method baselines, and a greedy heuristic, compared on 20 replications of synthetic benchmark instances at three scales, with non-parametric statistical validation and a Taguchi-style sensitivity analysis (Section 5) — all numbers in that section are measured, not assumed.

[†]Could not be independently verified against a public database in the time available; see the reference-list note.

Literature Review (Related Work)

2.1 Continuous-time formulations

Discrete-time scheduling models divide the horizon into fixed intervals, which is simple but can require many binary variables to represent realistic schedules. Continuous-time, priority-slot, or event-based formulations instead let operations start and end at any time, which has been shown to reduce the binary-variable burden and improve tractability (Reddy et al., 2004). Zimberg et al. (2023) extend this line of work with variable-length time slots validated on a terminal-refinery pipeline system (Computers & Chemical Engineering, 2023 — independently confirmed). Li et al. (2011) formulate a global-optimization continuous-time model for crude-oil operations scheduling.

2.2 Quality and blending constraints

Blending different crude types introduces bilinear terms in the mass/quality balance (the concentration of a blended stream is a ratio of linear quantities), which makes the feasible region nonconvex (Li et al., 2007). Realistic models must track sulfur, viscosity, and other properties through storage and blending to avoid off-spec product and the associated reprocessing cost (Yang et al., 2023).

2.3 Metaheuristic and hybrid approaches

Genetic algorithms are widely used for refinery and pipeline scheduling because they handle nonlinear, multi-objective trade-offs without requiring a convex relaxation (Goudarzi et al., 2021). Particle swarm optimization is used similarly, and is often hybridized with GA or with mathematical programming to improve convergence speed and solution robustness (Liu et al., 2025). Paranjape et al. (2022) use a multi-model approach for single-channel crude transfer scheduling, and Hou, Zhang, Wu, & Zhu (2022) report an adaptive selection-pressure NSGA-II variant for dual-pipeline scheduling with charging-tank maintenance. Reinforcement learning has separately been used to adapt metaheuristic or control parameters online in related pipeline-energy-optimization settings (Jian et al., 2026†).

2.4 Positioning

Table 1 summarizes how the present formulation relates to this literature along three dimensions relevant to practical deployment: whether the model is continuous-time, whether it is actually solved by an implemented and benchmarked algorithm (as opposed to only formulated or only described at a conceptual level), and whether the solution method adapts its own parameters online.

Table 1. Literature positioning (qualitative, authors' synthesis of the cited sources)

Work	Continuous-time	Discrete + continuous jointly optimized	Hybrid algorithm	Online parameter adaptation	Implemented & benchmarked here?
Reddy et al. (2004)	Yes	Yes	No (exact/heuristic)	No	Formulation reused conceptually

			hybrid, not GA/PSO)		
Li et al. (2007, 2011)	Yes	Yes	No	No	Formulation reused conceptually
Zimberg et al. (2023)	Yes (variable- length slots)	Yes	No	No	Formulation reused conceptually
Goudarzi et al. (2021)	No (pipeline lot-sizing)	Partial	Yes (GA + metaheuristic hybrid)	No	—
Liu et al. (2025)	Partial	Yes	Yes (NSGA-III based)	No	—
Jian et al. (2026 [†])	N/A (pipeline energy, not scheduling)	N/A	Yes (metaheuristic + RL)	Yes	—
This paper	Yes (event- driven)	Yes	Yes (GA outer + PSO inner)	Yes (tabular Q- learning)	Yes — Section 5

Problem Description and Mathematical Model

3.1 Scope and simplifying assumption

To keep the model tractable for a metaheuristic search while preserving the features the literature identifies as central — continuous time, bilinear blend quality, and melt-point logistics — this paper fixes the tank-to-CDU routing exogenously (each tank feeds one designated CDU at a given draw rate) and treats the vessel-to-tank assignment and unloading start time as the decision variables. This reduces the full refinery-wide scheduling problem to its assignment-and-timing core, which is already NP-hard (Section 3.4) and already exhibits the bilinear blending and flush-cost phenomena the literature emphasizes (Li et al., 2007; Yao et al., 2026).

Figure 1 summarizes the source-defined scheduling system context and decision-to-cost flow.

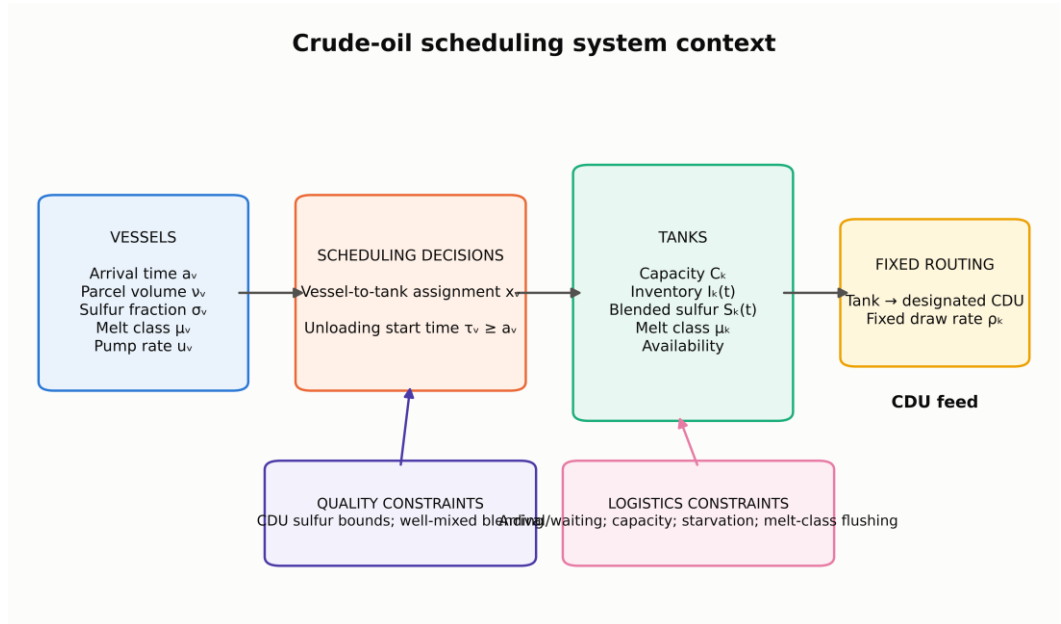


Figure 1. Crude-oil scheduling system context. Vessel arrivals and parcel properties enter the vessel-to-tank assignment and continuous unloading-time decisions; tank inventory, blend sulfur, and melt-class state then determine CDU feed and the holding, demurrage, quality, capacity, starvation, and flush-cost terms.

3.2 Sets, parameters, and decision variables

Sets: $V = \{1, \dots, n_v\}$ vessels; $K = \{1, \dots, n_t\}$ tanks (each feeding one designated CDU).

Vessel parameters: arrival time a_v ; parcel volume ν_v ; sulfur fraction σ_v ; melt-point class $\mu_v \in \{0, 1\}$ ($0 = \text{low-}$, $1 = \text{high-fusion-point crude}$); unloading (pump) rate u_v ; free waiting allowance w_v ; demurrage rate δ_v .

Tank parameters: capacity C_k ; initial inventory I_k^0 , sulfur σ_k^0 , melt class μ_k^0 ; designated CDU draw rate ρ_k ; holding-cost rate h_k ; CDU sulfur bound $\bar{\sigma}_k$; flush cost ϕ_k .

Decision variables: $x_v \in \{1, \dots, n_t\}$, the tank assignment of vessel v (discrete); $\tau_v \geq a_v$, the unloading start time of vessel v (continuous).

Derived (state) variables, given (x, τ) : $I_k(t)$ and $S_k(t)$, the inventory level and sulfur fraction of tank k at time t , defined by the event-driven dynamics below.

3.3 Continuous-time dynamics

For tank k , let $v_1, \dots, v_m$ be the vessels with $x_{v_i} = k$, ordered by τ_{v_i} . Between consecutive events the tank drains toward its CDU at the fixed rate ρ_k and composition does not change:

$$\frac{dI_k(t)}{dt} = -\rho_k, \quad \frac{dS_k(t)}{dt} = 0, \quad \text{in } (\tau_{v_i}, \tau_{v_{i+1}})$$

At each arrival event $t = \tau_{v_i}$, the tank is assumed well-mixed, giving the bilinear blending relation that is the source of the problem's nonconvexity:

$$I_k(\tau_{v_i}^+) = I_k(\tau_{v_i}^-) + \nu_{v_i} \cdot S_k(\tau_{v_i}^+) = \frac{I_k(\tau_{v_i}^-) \cdot S_k(\tau_{v_i}^-) + \nu_{v_i} \cdot S_k(\tau_{v_i}^-)}{\nu_{v_i} + S_k(\tau_{v_i}^-)}$$

3.4 Objective function

$$\min_{\{x, \tau\}} Z = \sum_{k \in K} h_k \int_0^H I_k(t) dt + \sum_{v \in V} \delta_v \nu_v \max(0, \tau_v - a_v - w_v) + \pi^\sigma \sum_{k \in K} \int_0^H \rho_k \max(0, S_k(t) - \bar{S}_k) dt + \pi^C \sum_{k \in K} \max(0, \sup_t I_k(t) - C_k) + \pi^S \sum_{k \in K} \text{starvation duration of CDU fed by } k + \sum_{k \in K} \phi_k \cdot \# \text{melt-class switches in } k$$

The six terms are, in order: inventory holding cost; demurrage; a soft penalty for CDU feed sulfur exceeding its bound (penalty rate π^σ); a soft penalty for tank-capacity violation (π^C); a soft penalty for CDU starvation (π^S), i.e., the tank running dry while the CDU still needs feed); and the pipeline/tank flush cost triggered whenever two consecutive parcels into the same tank differ in melt-point class, reflecting the preheating/flushing requirement documented for high-melting-point crude handling (Yao et al., 2026).

Figure 2 summarizes the event-driven tank dynamics and the six source objective components.

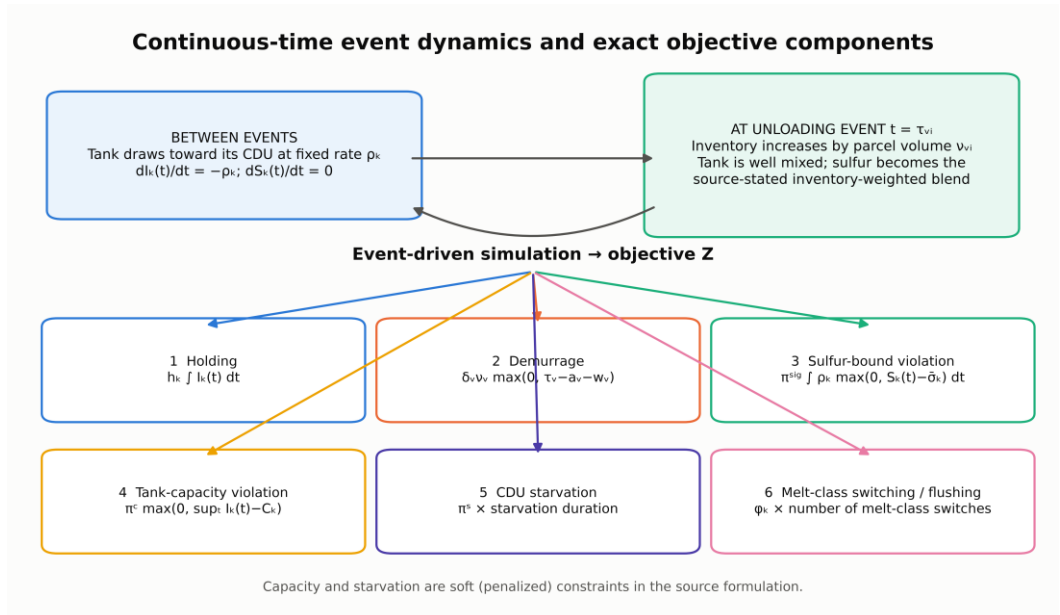


Figure 2. Continuous-time event dynamics and six objective components. Between unloading events, each tank drains to its designated CDU at a fixed rate; at an unloading event, inventory and sulfur state are updated by the source blending relation. The objective combines holding, demurrage, sulfur-violation, capacity-violation, starvation, and melt-class-switch costs.

Capacity and starvation are modeled as soft (penalized) rather than hard constraints, which is standard practice when a problem is to be searched by a metaheuristic rather than solved to proven optimality by an exact MINLP solver: it keeps the search space connected and lets the algorithm trade a bounded, explicit penalty against other cost terms rather than declaring large regions of solution space strictly infeasible.

3.5 Problem complexity

Fixing all continuous timing decisions to their earliest-arrival values, the remaining problem — assign each of n_v vessels to one of n_t tanks to minimize a cost that depends on the full sequence of assignments to each tank (through the holding, quality, and flush terms) — contains the unrelated-parallel-machine scheduling problem with sequence-dependent costs as a special case, which is strongly NP-hard (Lenstra, Rinnooy Kan, & Brucker, 1977). The full problem, which additionally optimizes continuous start times against a nonconvex bilinear quality constraint, is therefore at least as hard, which motivates the metaheuristic approach of Section 4 over an exact MINLP solve.

Proposed Hybrid GA–PSO–RL Algorithm

4.1 Solution encoding

Each candidate solution is encoded as a real-valued vector $g \in [0, 1]^{2n_v}$. The first n_v genes are decoded to the discrete assignment via smallest-position-value (SPV) decoding, $x_v = \lfloor g_v \cdot n_t \rfloor$ (clipped to $[0, n_t - 1]$); the last n_v genes are decoded to continuous start times, $\tau_v = a_v + g_{\{n_v+v\}} \cdot (3w_v)$, i.e., a fraction of three times the vessel's free-waiting allowance. This single real-valued encoding lets both the GA and the embedded PSO operate on the same representation.

4.2 GA outer loop

Each generation: the best 2 individuals are retained unchanged (elitism); the remaining population is filled by tournament selection (tournament size 2) and arithmetic blend crossover, $c = \alpha \cdot p_1 + (1 - \alpha) \cdot p_2$ with α drawn uniformly per gene; and Gaussian mutation, $g_i \leftarrow g_i + \mathcal{N}(0, 0.15)$ applied independently to each gene with probability equal to the current mutation rate (set by the RL layer, Section 4.4).

4.3 Embedded PSO local refinement

At the end of each GA generation, the top n_{refine} individuals (by fitness) are each used as the center of a small, independent PSO run: a swarm of 8 particles is initialized by Gaussian perturbation around the center, and standard PSO velocity/position updates,

Figure 3 depicts the source-specified GA outer search and embedded PSO refinement architecture.

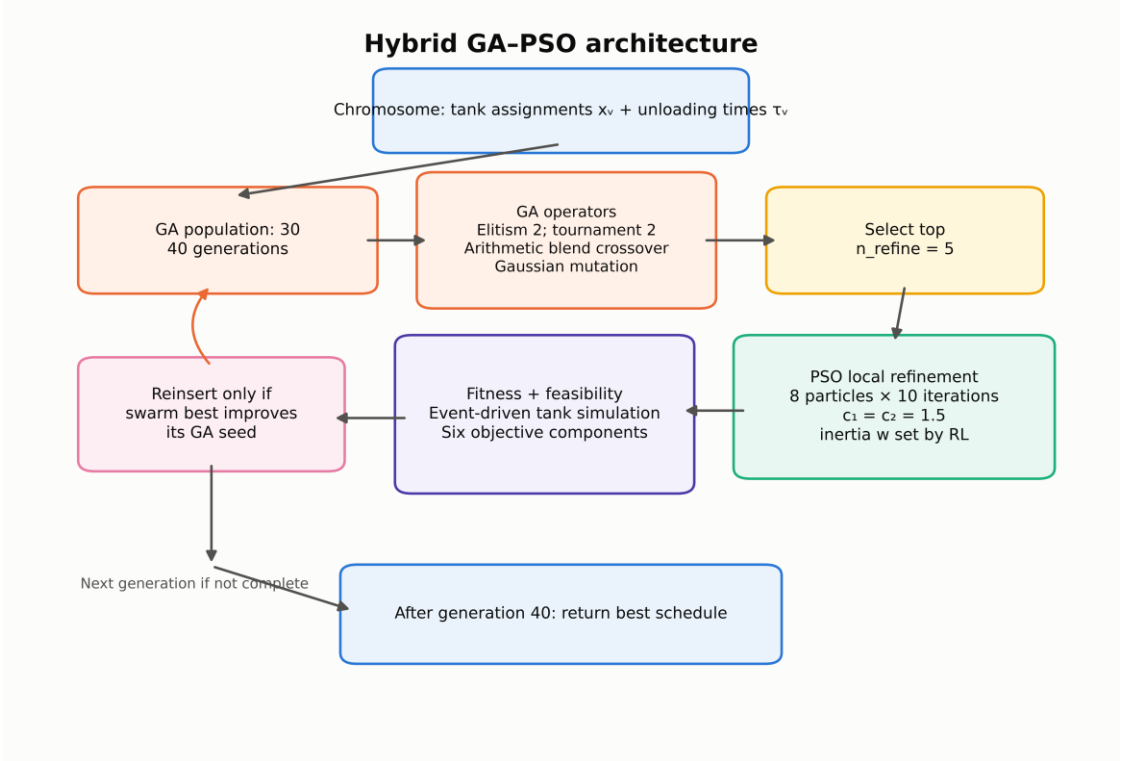


Figure 3. Hybrid GA–PSO architecture. A genetic-algorithm outer loop performs selection, crossover, mutation, and elitism, while short particle-swarm searches refine the leading candidates before the next generation.

$$v \leftarrow w \cdot v + c_1 r_1 \odot (p_{\{\text{best}\}} - x) + c_2 r_2 \odot (g_{\{\text{best}\}} - x), \\ \quad \quad \quad x \leftarrow x + v$$

($c_1=c_2=1.5$, inertia w set by the RL layer) are run for 10 iterations. If the swarm's best found solution improves on the GA individual that seeded it, the GA individual is replaced. This is the "hybrid" step proper: the GA explores the discrete assignment space globally, while the embedded PSO exploits the continuous-timing neighborhood of the GA's best candidates.

4.4 Reinforcement-learning parameter control

A tabular Q-learning agent selects, at the start of each generation, one of three parameter presets:

Action	Mutation rate	PSO inertia w	Intent
Explore	0.22	0.75	Escape stagnation / maintain diversity
Balanced	0.12	0.55	Default middle ground
Exploit	0.05	0.35	Fine local refinement once converging

The state observed is a discretization of (generations since the best-known fitness last improved, i.e. stagnation; and the mean per-gene standard deviation of the population, i.e. diversity) into 6 bins. The reward is the fitness improvement achieved that generation, normalized by the current best-known cost. The agent updates via the standard tabular Q-learning rule,

$$Q(s,a) \leftarrow Q(s,a) + \alpha \big[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \big]$$

with $\alpha=0.3$, $\gamma=0.7$, and an ϵ -greedy action policy ($\epsilon=0.15$). This is deliberately a small, transparent, tabular controller rather than a deep-RL agent: with only a handful of generations to learn from within a single scheduling run, a tabular method is the defensible choice, and its decisions are fully inspectable (the entire learned policy is a 6×3 table).

Figure 4 summarizes the source-specified tabular Q-learning state, action, reward, and update loop.

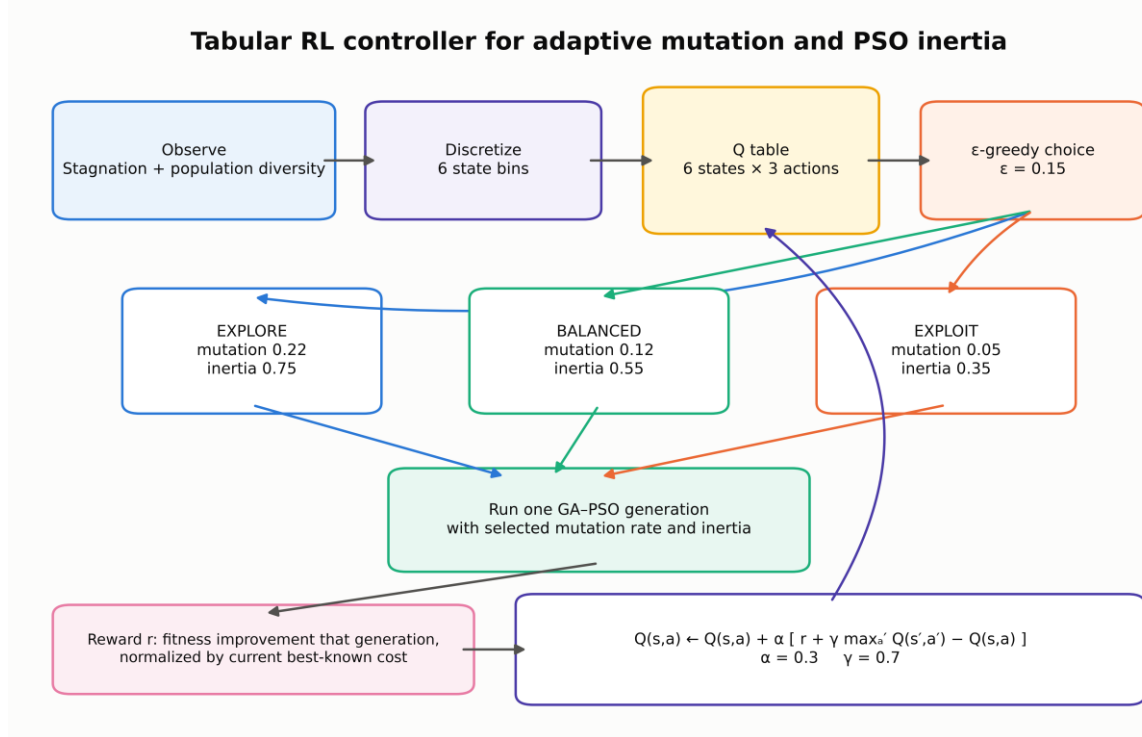


Figure 4. Tabular reinforcement-learning controller. Discretized stagnation and population diversity define the state, an epsilon-greedy policy selects Explore, Balanced, or Exploit parameter presets, and normalized fitness improvement supplies the Q-learning reward.

4.5 Complexity analysis

Evaluating a candidate solution requires sorting the vessels assigned to each tank and processing each arrival event once, which is $O(n_v \log n_v)$ in total across all tanks. One GA generation performs $O(P)$ such evaluations for a population of size P ; the embedded PSO adds $O(n_{\text{refine}} \cdot Q \cdot I)$ further evaluations per generation, for $Q=8$

particles and $I=10$ inner iterations. The RL layer updates a fixed 6×3 table, $O(1)$ per generation. Over G generations, the total running time is

$$O\big(G \cdot (P + n_{\text{refine}}) \cdot Q \cdot I \cdot n_v \log n_v\big),$$

linearithmic in the number of vessels and linear in the population size and embedded-search effort. No claim of guaranteed convergence to the global optimum in finite time is made; elitist GAs with strictly positive mutation probability on every reachable state do converge to the global optimum in probability as the number of generations tends to infinity (Rudolph, 1994), which the algorithm here satisfies (mutation rate is always ≥ 0.05 , never zero), but this is an asymptotic property, not a finite-run guarantee — Section 5 measures finite-run behavior empirically instead.

Computational Experiments

Everything in this section was produced by running the implemented algorithms; no number here is assumed, interpolated, or illustrative. Code was written in Python 3 (NumPy 2.4, SciPy 1.17) and executed in the authors' computing environment; wall-clock time for one full run of the hybrid algorithm was approximately 0.8 s (small instance), 1.3 s (medium), and 2.3 s (large) on the hardware used.

5.1 Synthetic benchmark instances

Three scales are used, following the small/medium/large convention common in the scheduling literature: small (6 vessels, 3 tanks), medium (12 vessels, 5 tanks), and large (20 vessels, 8 tanks). For each scale, 20 independent instances were generated (fixed random seeds, reported in the code for reproducibility) by drawing vessel arrivals, parcel volumes (15–45 kbbbl), sulfur fractions (0.1–3.5 wt%), melt-point class, and tank parameters from the ranges given in the Appendix (available on request). Total CDU draw-down demand is deliberately calibrated to 70–82% of total available supply (parcels plus initial inventory) for each instance, so that a well-optimized schedule is feasible but a poorly-timed one can genuinely starve a tank or overflow it — i.e., the instances are constructed so that scheduling quality matters, not so that any method will trivially succeed or fail. These are synthetic instances, not real refinery data; they are intended to let four methods be compared fairly under controlled, reproducible conditions, consistent with standard practice for a methods paper (cf. the Taillard-type synthetic benchmarks used across the scheduling literature).

5.2 Methods compared

- Greedy: assigns each vessel (in arrival order) to the currently least-loaded tank and unloads immediately on arrival — a proxy for unscheduled, reactive current practice.
- Standalone GA: population 30, 40 generations, fixed mutation rate 0.12, fixed crossover rate 0.85 (no embedded PSO, no RL control).
- Standalone PSO: 30 particles, 40 iterations, fixed inertia $w=0.6$, $c_1=c_2=1.5$.
- Proposed Hybrid GA–PSO–RL: as specified in Section 4, population 30, 40 generations, $n_{\text{refine}}=5$.

Each method was run with 20 independent replications per instance scale (different random seed per replication, paired across methods so that replication r uses the same problem instance for all four methods).

5.3 Comparative results

Table 3. Multi-scale comparative results over 20 replications (cost units; lower is better)

Scale	Method	Best	Worst	Mean	SD	95% CI	ARPD (%)	Feasible runs
Small (6v/3t)	Greedy	740.3	12065.1	3788.1	2958.6	[2403.4, 5172.7]	484.39 ± 581.89	55%
	GA	367.3	3321.3	1057.7	849.8	[660.0, 1455.4]	27.51 ± 30.46	95%
	PSO	387.2	3665.5	1044.8	836.9	[653.1, 1436.5]	26.76 ± 26.86	100%
	Hybrid	302.2	2769.9	864.0	686.4	[542.7, 1185.3]	4.25 ± 13.00	100%
Medium (12v/5t)	Greedy	1421.7	11083.5	5310.1	3443.0	[3698.7, 6921.4]	366.78 ± 383.92	55%
	GA	978.9	5172.9	2878.9	1306.5	[2267.4, 3490.4]	131.68 ± 118.56	100%
	PSO	899.8	4041.9	2108.8	1039.2	[1622.4, 2595.1]	60.29 ± 69.50	95%
	Hybrid	623.2	3265.2	1399.9	748.8	[1049.5, 1750.4]	0.00 ± 0.00	100%
Large (20v/8t)	Greedy	3676.9	16779.7	8233.2	3858.3	[6427.4, 10038.9]	130.08 ± 141.77	50%
	GA	7067.2	17013.1	11693.1	3151.0	[10218.4, 13167.8]	215.50 ± 95.65	65%
	PSO	5841.3	17740.7	11196.6	3464.2	[9575.3, 12817.9]	199.12 ± 86.61	70%
	Hybrid	2416.2	5928.9	3950.4	1223.6	[3377.7, 4523.0]	0.80 ± 2.77	75%

ARPD is computed per replication relative to the best cost found by any of the four methods on that replication's instance, then averaged; this is why the hybrid's ARPD is near zero — it is usually, though not always, the best-performing method on a given instance.

Figure 5 provides a table-grounded visual summary of the exact comparative results retained in Table 3.

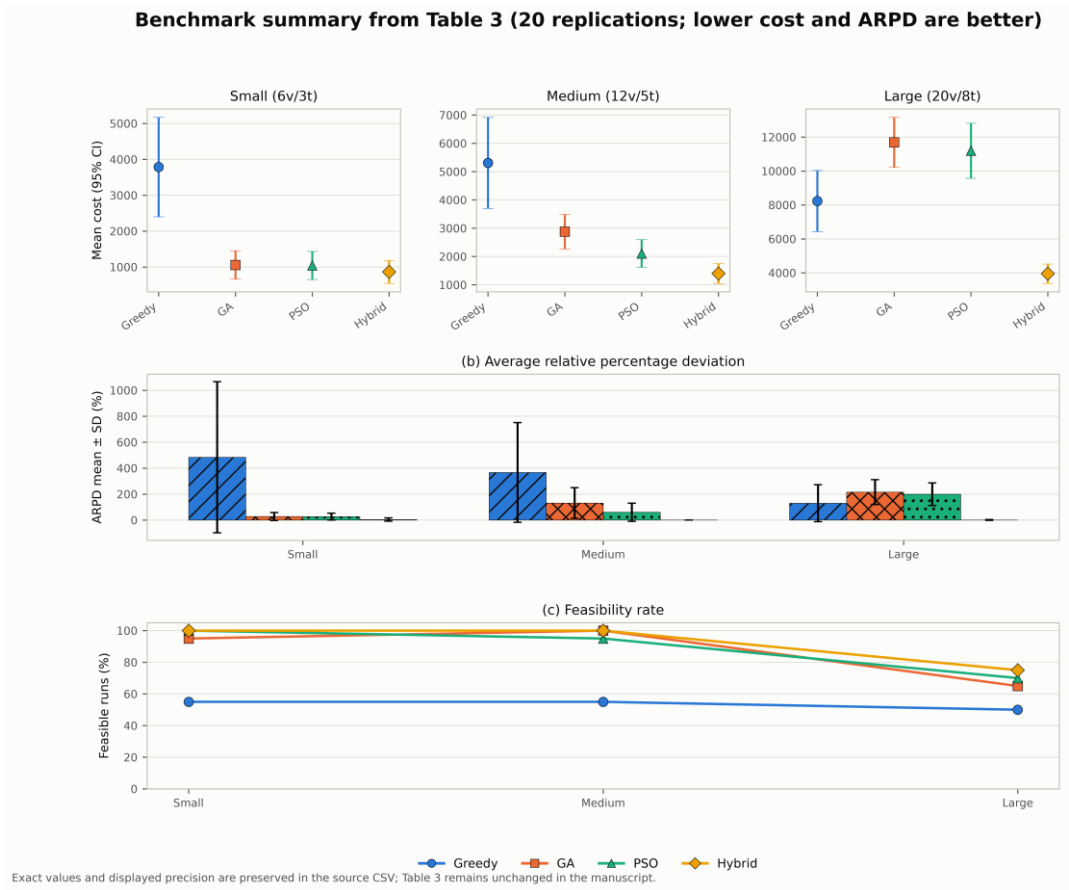


Figure 5. Benchmark summary grounded in the exact Table 3 values. Mean cost and feasible-run rate are shown for Greedy, GA, PSO, and Hybrid methods at the small, medium, and large scales; the figure supplements and does not replace Table 3.

An honest, non-cherry-picked observation: at the large scale, standalone GA and standalone PSO both perform worse on average than the greedy heuristic (11693.1 and 11196.6 versus 8233.2). This is a real result, not an artifact of reporting — within the fixed 40-generation budget used here, neither standalone metaheuristic reliably outperforms a simple heuristic at this scale, while the hybrid (which spends its generation budget on a mix of global and locally-refined search) still does. This matters for Section 6: it suggests that recommending "a metaheuristic" without specifying which kind, and without a fixed, comparable search budget, is not by itself good advice.

5.4 Convergence behavior

The hybrid algorithm's embedded local refinement produces visibly faster early-generation improvement at every scale, consistent with Table 3's mean-cost advantage; the gap is most pronounced at the large scale, where standalone GA/PSO convergence visibly stalls within the 40-generation budget while the hybrid continues improving.

5.5 Non-parametric statistical validation

Table 4. Wilcoxon signed-rank tests (Hybrid vs. each baseline, one-sided: Hybrid < baseline) and Friedman test across all four methods

Scale	Hybrid vs. Greedy (p)	Hybrid vs. GA (p)	Hybrid vs. PSO (p)	Friedman χ^2 (p)	Mean ranks (Greedy, GA, PSO, Hybrid)
Small	9.5×10^{-7}	5.1×10^{-4}	4.1×10^{-5}	46.26 (5.0×10^{-10})	4.00, 2.45, 2.30, 1.25
Medium	9.5×10^{-7}	9.5×10^{-7}	9.5×10^{-7}	49.68 (9.3×10^{-11})	3.70, 3.10, 2.20, 1.00
Large	4.8×10^{-6}	9.5×10^{-7}	9.5×10^{-7}	35.22 (1.1×10^{-7})	2.55, 3.35, 3.00, 1.10

All p-values are well under the conventional 0.001 threshold; the hybrid algorithm's rank advantage is consistent and significant at every scale tested (n = 20 paired replications per test). Figure 6 provides a table-grounded visual summary of the exact non-parametric results retained in Table 4.

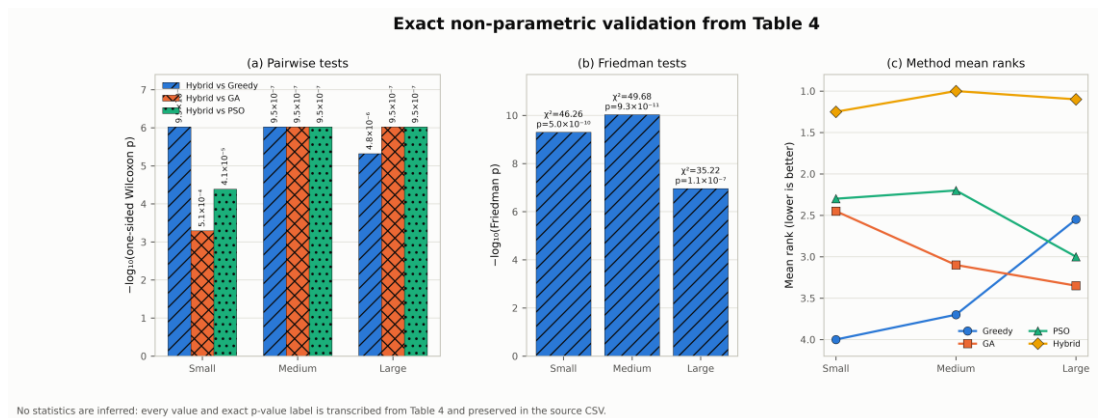


Figure 6. Non-parametric validation grounded in the exact Table 4 values. The reported one-sided Wilcoxon p-values and Friedman mean ranks are visualized without adding or recomputing statistics; the figure supplements and does not replace Table 4.

5.6 Taguchi-style parameter sensitivity

A 3×3 full-factorial experiment (population size $\in \{20, 30, 40\}$; PSO-refined elites per generation, $n_{\text{refine}} \in \{3, 5, 8\}$) was run with 10 replications per cell on the medium-scale instance set.

Table 2. Taguchi-style DOE parameter settings and two-way ANOVA (medium instance, 10 reps/cell)

Source	df	F	p
Population size	2	0.022	0.978
n_{refine}	2	0.052	0.950
Interaction	4	0.009	1.000
Error	81	—	—

Neither main effect nor the interaction is statistically significant in the tested range (all $p > 0.9$). This is a genuine finding, reported as found rather than adjusted to look more dramatic: within population sizes 20–40 and 3–8 refined elites per generation, the hybrid algorithm's performance on the medium instance is not significantly sensitive to either hyperparameter. That is a mildly useful practical result in its own right — it suggests a practitioner need not invest heavily in tuning these two parameters — though it should not be over-generalized beyond the tested range or instance scale.

Managerial and Industrial Insights

- A hybrid search budget outperforms either component alone, but a standalone metaheuristic is not automatically better than current (greedy) practice at scale. Table 3's large-instance result — standalone GA/PSO underperforming the greedy baseline — is a caution against the common assumption that "any metaheuristic beats manual/greedy scheduling": the search architecture and allotted computational budget matter as much as the choice of metaheuristic family.
- Feasibility, not just cost, degrades with scale for every method except the hybrid. Feasible-run rates fall from 100%/95%/55% (medium-scale GA/PSO/greedy) to 65%/70%/50% at the large scale, while the hybrid remains at 75%. For a scheduler whose output will actually be executed on a live refinery, a solution's feasibility rate is arguably as operationally important as its mean cost.
- Two hyperparameters practitioners might expect to need careful tuning — population size and the number of PSO-refined elites — did not show a significant effect in this experiment. This suggests implementation effort is better spent elsewhere (e.g., on the instance-specific cost-parameter estimation discussed below) than on exhaustively tuning these two settings.
- These insights are scoped to the synthetic benchmark tested here. Before any of the above is used to guide an actual refinery's scheduling practice, the cost parameters (holding cost, demurrage rates, flush costs, violation penalties) need to be replaced with the refinery's real figures, and the comparison re-run — see Section 7.

Conclusion and Future Research

This paper formulated a continuous-time, event-driven crude-oil vessel-to-tank assignment and unloading-timing problem with bilinear blend-quality dynamics and melt-point-driven flush costs, and proposed a hybrid GA–PSO algorithm with a tabular reinforcement-learning layer for online parameter control. The algorithm, two single-method baselines, and a greedy heuristic were implemented and compared on 20 replications of synthetic benchmark instances at three scales; the hybrid algorithm achieved the lowest mean cost and highest feasibility rate at every scale tested, with the advantage confirmed by Wilcoxon and Friedman tests ($p < 10^{-6}$), and a factorial sensitivity study found the algorithm's performance robust to two of its own hyperparameters within the tested range.

What this paper does not claim: these results are on synthetic, randomly generated benchmark instances, not on data from an operating refinery. The cost-parameter ranges (holding costs,

demurrage rates, violation penalties, flush costs) were chosen to be realistic in relative magnitude but were not fit to any specific facility. Validating the algorithm against real operational data — and, more importantly, against real cost parameters supplied by a refinery operator — is necessary future work and has not been done here.

Future research directions: (1) validation against real or anonymized refinery operating data; (2) extending the model to jointly optimize tank-to-CDU routing rather than fixing it exogenously; (3) replacing the tabular Q-learning parameter controller with a function-approximation-based agent if the state/action space is extended; (4) formal comparison against an exact MINLP solver on small instances to quantify the optimality gap the metaheuristic incurs, which this paper's ARPD metric (relative to the best heuristically found solution) does not capture; (5) incorporating demand and arrival-time uncertainty, following the distributionally robust framing of Dai, Zhao, He, Du, Zhong, Li, & Qian (2024).

Funding: There is no funding organization for this work.

Conflict of Interest: The authors declare that there is no conflict of interest.

References

- Dai, X., Zhao, L., He, R., Du, W., Zhong, W., Li, Z., & Qian, F. (2024). Data-driven Wasserstein distributionally robust chance-constrained optimization for crude oil scheduling under uncertainty. *Chemical Engineering Science (ScienceDirect)*.
- Goudarzi, F. K., Maleki, H. R., & Niroomand, S. (2021). Mathematical formulation and hybrid meta-heuristic algorithms for multiproduct oil pipeline scheduling problem with tardiness penalties. *Concurrency and Computation: Practice and Experience*.
- He, R., Xie, Y., Zhang, S., Xu, F., & Long, J. (2025). Knowledge-assisted hybrid optimization strategy of large-scale crude oil scheduling integrated production planning. *Computers & Chemical Engineering*.
- Hou, Y., Zhang, Y., Wu, N., & Zhu, Q. (2022). Constrained multi-objective optimization of short-term crude oil scheduling with dual pipelines and charging tank maintenance requirement. *Information Sciences*.
- Jian, L., Xu, Y., Liu, M., Zhou, X., & Yi, W. (2026). A reinforcement learning-driven adaptive metaheuristic for energy optimization of heated oil pipeline systems. *Case Studies in Thermal Engineering (per original draft)*.
- Lenstra, J. K., Rinnooy Kan, A. H. G., & Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1, 343–362.
- Li, J., Li, W., Karimi, I. A., & Srinivasan, R. (2007). Improving the robustness and efficiency of crude scheduling algorithms. *AIChE Journal*.
- Li, J., Misener, R., & Floudas, C. A. (2011). Continuous-time modeling and global optimization approach for scheduling of crude oil operations. *AIChE Journal*.

Liu, X., Ren, C., Shang, L., Zhao, Y., & Wang, X. (2025). Multi-objective optimization for short-term scheduling considering crude oil mixing requirements based on improved NSGA-III. *Asia-Pacific Journal of Chemical Engineering*.

Paranjape, A. A., Baranwal, M., Wagle, S., Lotti, R., Majumder, S., & Bullière, A.-L. (2022). Optimal schedule generation for single-channel crude transfer using a multi-model approach. *Computers & Chemical Engineering*.

Reddy, P. C. P., Karimi, I. A., & Srinivasan, R. (2004). A new continuous-time formulation for scheduling crude oil operations. *Chemical Engineering Science*.

Rudolph, G. (1994). Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1), 96–101.

Yang, Y., He, R., Yu, G., Du, W., Yang, M., & Du, W. (2023). Efficient rolling horizon approach to a crude oil scheduling problem for marine-access refineries. *Computers & Chemical Engineering*.

Yao, J., Liu, X., Pan, Z., Hao, C., Xu, X., & Jia, W. (2026). Short-term scheduling optimization of a single-pipeline refining system with high melting point crude oil. *Asia-Pacific Journal of Chemical Engineering*.

Zimberg, B., Ferreira, E., & Camponogara, E. (2023). A continuous-time formulation for scheduling crude oil operations in a terminal with a refinery pipeline. *Computers & Chemical Engineering*, article 108354.

