Proceedings of the ASME 2026
International Design Engineering Technical Conference and
Computer and Information in Engineering Conference
IDETC/CIE2026
August 23–26, 2026, Houston, TX

DETC2026-194209

A PHASE-I BENCHMARK OF SELF-ORGANIZING MANUFACTURING CONTROL UNDER HETEROGENEITY AND VOLATILITY

Jian Ni[1], Shih-Chun Deng[1], Yan Jin[1,*]

[1]IMPACT Laboratory
Department of Aerospace and Mechanical Engineering
University of Southern California
Los Angeles, CA 90089
jianni@usc.edu, dengshih@usc.edu, yjin@usc.edu

ABSTRACT

Cross-method comparisons in manufacturing scheduling often suffer from mismatched protocols, inconsistent seed handling, and incomplete failure reporting, obscuring whether differences reflect controllers or benchmarking artifacts. This paper presents an aligned Phase-I benchmark comparing a shortest-processing-time (SPT) dispatching baseline with two proximal policy optimization (PPO)-trained controllers: a centralized joint-decision controller and a decentralized self-organizing multi-agent reinforcement learning (SO-MARL) controller. Under a controlled 2×2 heterogeneity–volatility scenario matrix, the protocol enforces a common training budget, predeclared checkpoint screening, deterministic final evaluation, and explicit admissibility labels for valid, partial, and invalid outcomes with failure reasons. Results show regime-dependent controller ranking: no method dominates all four scenarios. In the main fixed-instance comparison, Centralized PPO remains admissible across all four scenarios, whereas SO-MARL is competitive in three regimes but inadmissible under simultaneous high heterogeneity and high volatility. Diagnostic retuning probes show that the tested static SO-MARL settings remain regime-sensitive; bounded supplemental checks over three instance seeds and eight training seeds per core learned controller show continued SO-MARL scenario sensitivity and full Centralized PPO admissibility within that inventory. Together, these results establish a Phase-I benchmark centered on aligned evaluation, admissibility-aware reporting, and boundary finding across heterogeneity and volatility regimes.

Keywords: manufacturing scheduling, self-organizing multi-agent reinforcement learning, SmartSOM, benchmarking, heterogeneity, volatility, proximal policy optimization

*Corresponding author: yjin@usc.edu

1. INTRODUCTION

Manufacturing control in high-mix and personalized production must adapt to heterogeneous resource capabilities and time-varying disturbances [1, 2]. Under such conditions, cross-method conclusions can be unreliable when protocols, seed handling, and failure reporting are not aligned. In particular, many existing comparisons report only successful episodes or best-case performance, quietly excluding failed or non-terminating runs — yet for manufacturing deployment, operational reliability matters as much as average performance. The job shop scheduling problem (JSSP) is a classical combinatorial optimization problem in manufacturing systems [3], and multi-agent architectures have been widely explored for distributed manufacturing scheduling and control [4].

Against this background, this paper presents an aligned benchmark comparison of a fixed-mode shortest-processing-time (SPT) rule baseline, Centralized PPO, and self-organizing multi-agent reinforcement learning (SO-MARL) under a common evaluation protocol. To keep the main comparison controlled, the primary benchmark uses one fixed benchmark instance (seed101) and one selected training run per learning-based controller/scenario pair; Appendix B separately reports bounded supplemental robustness checks over benchmark-instance seeds 101, 202, and 303 and eight training seeds per core learned controller. The Phase-I scope is focused on the SO-MARL controller and benchmark protocol; higher-level adaptive mechanisms are reserved for future study. The study therefore contributes a scoped protocol for comparing controller behavior under aligned heterogeneity–volatility regimes and explicit admissibility reporting.

The contributions are threefold. First, an aligned benchmark protocol is established for comparing rule-based, central-

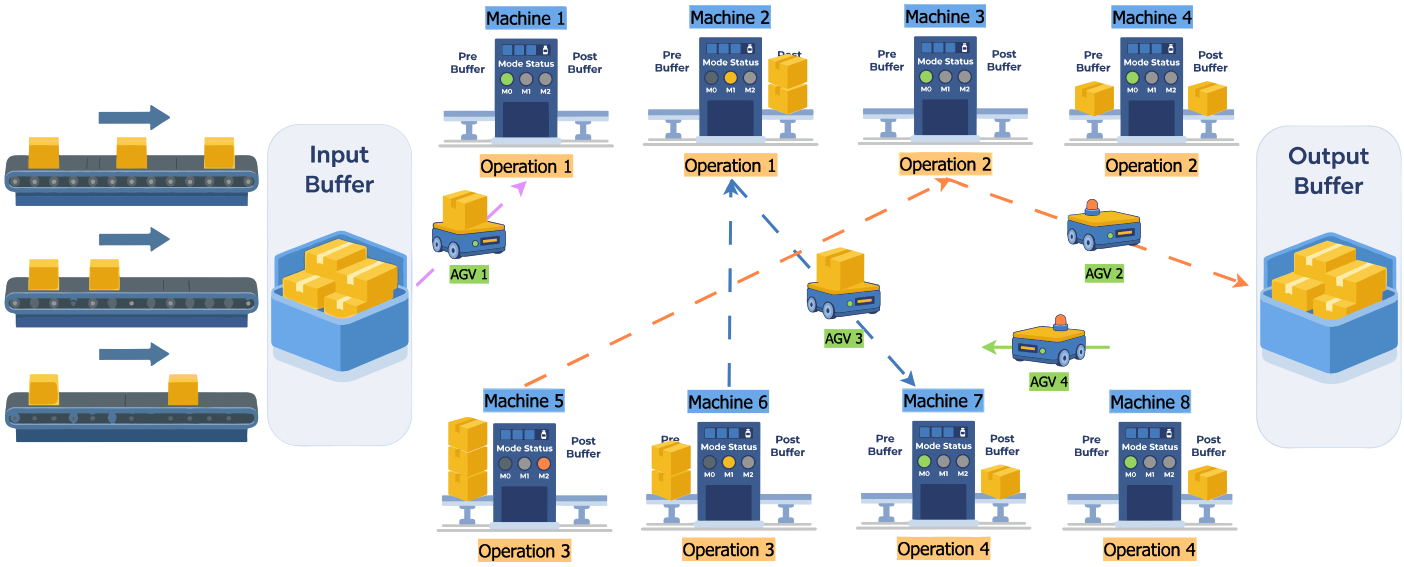

FIGURE 1: Benchmark shop-floor layout and routing structure. Jobs enter through the input buffer, are transported by automated guided vehicles (AGVs) among heterogeneous machines M1–M8 across operations O1–O4, and exit through the output buffer after final processing. The benchmark therefore extends a classical job shop scheduling problem into a buffered, AGV-coupled, heterogeneous manufacturing scheduling problem with machine-side mode decisions and transport-side coordination.

ized reinforcement learning (RL), and self-organizing multi-agent reinforcement learning (MARL) controllers under a controlled 2×2 heterogeneity–volatility scenario matrix, with common stop rules, predeclared checkpoint-screening procedures, and deterministic evaluation — elements that are commonly absent in existing manufacturing scheduling comparisons. Second, this protocol is instantiated as a reproducible controlled comparison under this setup, supplemented by bounded robustness checks over benchmark-instance seeds 101, 202, and 303 and eight training seeds for each core learned controller. The evidence reveals that controller ranking is regime-dependent within the tested setting: no method dominates across all four heterogeneity–volatility conditions, and diagnostic configuration probes indicate that the tested static SO-MARL configurations remain regime-sensitive, motivating future study of adaptive mechanism layers. Third, admissibility is made an explicit part of benchmark evaluation: makespan and passing rate are reported together with episode-level termination validity and failure modes, retaining partial and invalid outcomes rather than filtering out unsuccessful runs — a reporting discipline that is important for manufacturing control benchmarking but commonly absent in the literature.

2. RELATED WORK

Rule-based dispatching and scheduling methods remain core manufacturing baselines because they are interpretable, computationally light, and easy to deploy in practice. Their practical value makes them an essential reference point for any benchmark that evaluates robustness under changing shop-floor conditions.

Reinforcement learning (RL) and deep reinforcement learning (DRL) approaches, including centralized, joint, and multi-agent formulations, have been studied to improve adaptation beyond fixed dispatching priorities in production scheduling and control [5–10]. These methods can improve system-level performance. For instance, Gui et al. [5] applied multi-agent deep deterministic policy gradient (MADDPG) to dynamic flexible job-shop scheduling to minimize tardiness, while Qin et al. [6] developed a multi-agent dueling DRL approach targeting mass-personalization scheduling. Waseem and Chang [9] addressed flexible manufacturing via Nash-MADDPG, and Li et al. [10] combined attention mechanisms with physics-guided dynamics for multi-stage process control. Despite these advances, their outcomes are often sensitive to training dynamics, checkpoint selection, and evaluation protocol choices.

Self-organizing, distributed, and multi-agent reinforcement learning (MARL)-style control frameworks have been introduced to support decentralized coordination across heterogeneous resources [11–16]. In manufacturing settings, these approaches are promising because they align with local decision contexts. Li et al. [11] combined self-organizing resource matching with multi-agent DRL for manufacturing scheduling. Qin and Lu [12] proposed a Self-Organizing Manufacturing Network (SOMN) with self-configuration and self-healing functions as a paradigm for mass personalization. Park and Tran [13] demonstrated decentralized disturbance adaptation through a swarm of cognitive agents. However, these approaches suffer from the curse of dimensionality, heavy communication overhead, and unstable convergence in large heterogeneous agent societies [15, 16]. Benchmark evidence that compares them directly against rule-based and centralized RL baselines under aligned heterogeneity–volatility regimes remains limited.

Beyond control methods, the benchmarking methodology itself is relevant. Classical job shop scheduling benchmarks such as the Taillard instance set [17] and the DMU instance set [18] provide standardized problem instances for comparing optimization algorithms, but generally assume static deterministic instance definitions. RL-oriented environments such as

JobShopLab [19] extend job-shop scheduling toward simulation-based reinforcement-learning evaluation with practical shop-floor constraints such as transport logistics, buffers, setup times, machine disruptions, and stochastic processing conditions. However, they do not by themselves define a controlled benchmark protocol for separately varying resource-team heterogeneity and release/task-map volatility, nor do they require admissibility-aware reporting of invalid or partially terminating learned-control outcomes. Building on this environment, the present work adapts the simulation setting into an aligned stress-test protocol with a controlled 2×2 heterogeneity–volatility scenario matrix, common training budgets, predeclared checkpoint-selection rules, fixed evaluation seeds, and explicit failure reporting. The contribution is therefore a benchmark protocol and reporting contract for comparing rule-based, centralized RL, and self-organizing MARL controllers under aligned operating regimes.

3. BENCHMARK SETUP

3.1. Scenario Matrix

Figure 1 illustrates the benchmark shop-floor layout, including heterogeneous machines M1–M8, AGV transport, and input/output buffers. This benchmark uses a 2×2 heterogeneity–volatility matrix with explicit scenario mapping: S00 = low heterogeneity/low volatility, S01 = low heterogeneity/high volatility, S10 = high heterogeneity/low volatility, and S11 = high heterogeneity/high volatility. This structure provides aligned coverage of the four operating-regime corners, with S11 representing simultaneous high heterogeneity and high volatility.

3.2. Scenario Parameterization

Structural heterogeneity is defined over resource-team role counts as

$$H = -\sum_{r=1}^{F} p_r \ln p_r, \quad p_r = \frac{n_r}{N}, \quad \bar{H} = \frac{H}{\ln F} \tag{1}$$

where r indexes resource roles in $\{\mathrm{O1}, \mathrm{O2}, \mathrm{O3}, \mathrm{O4}, \mathrm{AGV}\}$, n_r is the number of resources assigned to role r, $N = \sum_r n_r$, and $F = 5$. Resources are grouped by functional role: machine-side roles are distinguished by operation capability over O1–O4, while AGVs form a separate homogeneous transport role. The AGV role is held fixed at four vehicles in both heterogeneity settings, but it is included because the benchmark is an AGV-coupled manufacturing system rather than a machine-only job shop. In this benchmark, low and high heterogeneity are relative scenario levels of the resource-team role distribution, not universal thresholds. The low-heterogeneity setting concentrates machine-side capability on O1 with counts (5, 1, 1, 1), making most operation types depend on a small number of eligible machines. The high-heterogeneity setting uses a more balanced machine-side role distribution, (2, 2, 2, 2), so the structural resource-team entropy is higher. The corresponding entropy values are summarized in Table 1.

Volatility is computed from the per-step normalized task-graph change plotted in Figure 2:

$$\Delta G_t = \frac{|G(t) - G(t-1)|}{N_G(t)}, \qquad V_{\mathrm{vol}} = \frac{1}{T}\sum_{t=1}^{T} \Delta G_t. \tag{2}$$

TABLE 1: Resource-team role counts used to compute the low- and high-heterogeneity scenarios. O1–O4 counts indicate how many machines can perform each operation type; AGV is a separate homogeneous transport role fixed at four vehicles and included in the structural entropy calculation.

Setting	O1	O2	O3	O4	AGV	H	$\bar{H}$
Low heterogeneity	5	1	1	1	4	1.35	0.84
High heterogeneity	2	2	2	2	4	1.56	0.97

Here $G(t)$ denotes the task graph at phase/time step t, $|G(t) - G(t-1)|$ is the amount of task-map editing between consecutive task graphs, and $N_G(t)$ is the task-graph vertex count used for normalization. Thus, the plotted vertical trace is ΔG_t, while the V_{vol} value shown in each Figure 2 panel title is the time average of that trace.

In the present benchmark, volatility is operationalized through a combined release-time and task-map-change profile. The release-time component controls how quickly jobs enter the system, while the task-map-change component controls how often the task graph changes across phases. Low volatility therefore represents a milder operating regime with lower release pressure and fewer graph edits; high volatility represents a more disturbed regime with higher release pressure and more frequent graph edits. In the low-volatility setting, the job-release intensity parameter is 0.25, task-map edit probabilities by phase are [0.04, 0.064, 0.088, 0.112, 0.136, 0.16], and the realized seed101 volatility value is $V_{\mathrm{vol}} = 0.0014$. In the high-volatility setting, the job-release intensity parameter is 0.75, task-map edit probabilities are [0.30, 0.39, 0.48, 0.57, 0.66, 0.75], and the realized seed101 volatility value is $V_{\mathrm{vol}} = 0.0042$.

3.3. Benchmark Environment and Resource Model

The benchmark environment adapts the JobShopLab simulation framework [19], a reinforcement-learning-oriented job shop scheduling environment. It retains the environment's buffered production and transport abstractions while adding the study-specific heterogeneity–volatility scenario design and mode-dependent processing/error behavior. Jobs are represented as ordered sequences of operations, where each operation is labeled by an operation type (O1–O4) and an associated processing time. Each job route has length 2–4. For a job j, the fixed benchmark instance records an ordered operation-type route, the corresponding operation-time vector, and a release time. The exact primary-instance job list is deterministically generated and remains fixed across compared methods. Machines are not fully interchangeable; instead, each machine is defined by a capability set over operation types.

The heterogeneity axis is therefore realized through the resource-team role distribution: machine-side roles vary across O1–O4, while the AGV transport role remains fixed and homogeneous. The volatility axis is realized through release-time dynamics and task-map edits. The job pool remains fixed within each seed. All benchmark instances use 100 jobs, 8 machines, 4 homogeneous AGVs, and four operation types (O1–O4).

AGV transportation is modeled at the node-to-node service level using a fixed transport-time matrix, a common abstraction

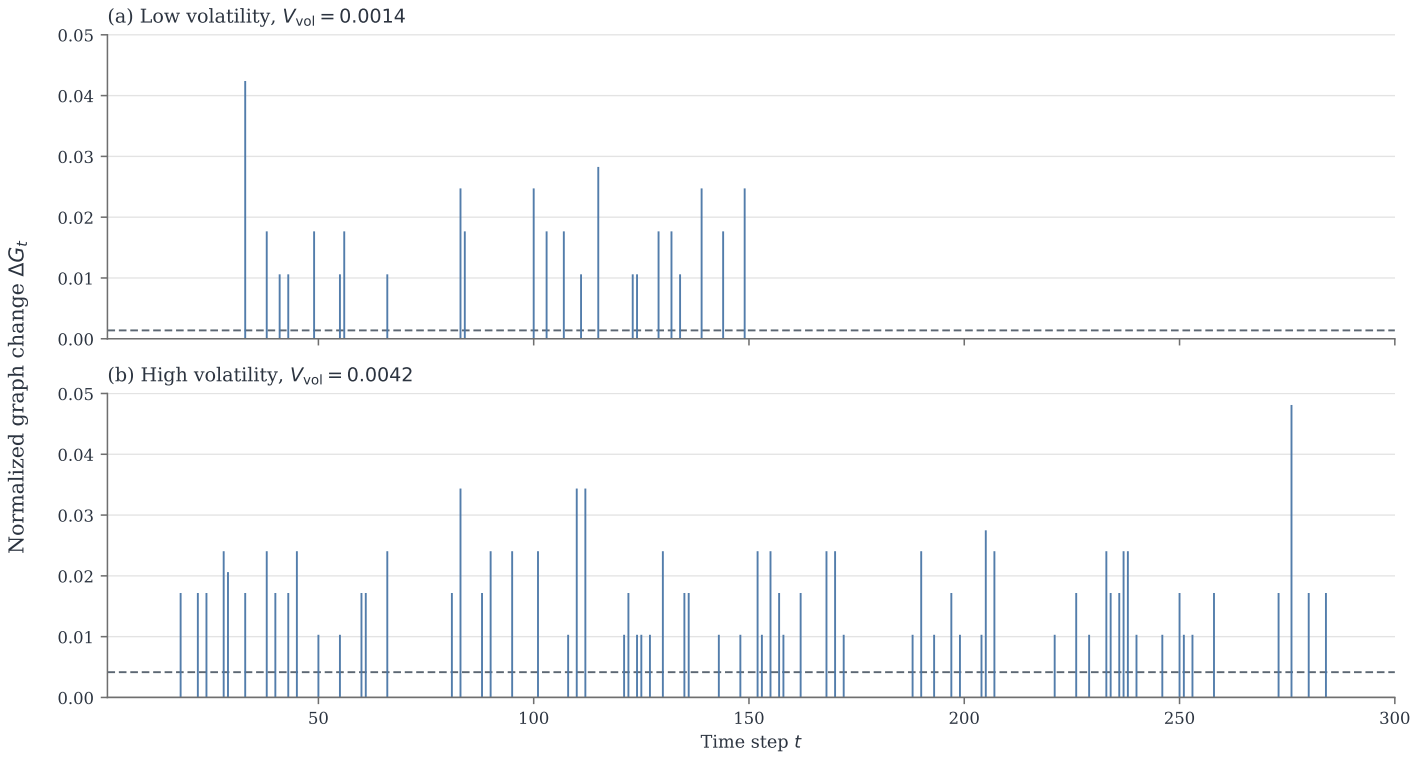

FIGURE 2: Representative realized volatility traces for the low- and high-volatility settings under seed101. The plotted vertical bars show the per-step normalized graph change ΔG_t; the dashed horizontal line and the value V_{vol} shown in each panel title report the time-average volatility level for that setting.

used in AGV system studies [20], over the input buffer, output buffer, and machine-side service nodes. When an AGV is assigned a transport task, the travel duration is obtained directly from this matrix. The present benchmark does not explicitly model route choice, collision avoidance, or arc-level traffic interaction; these effects are abstracted into the fixed node-to-node transport-time representation. The exact matrix is documented in Appendix A.

Within each seed, the job pool is held fixed and the defect-generation model is also held fixed across scenarios. Each machine-processing action uses one of three preset processing modes: M0 (slowest, lowest-error), M1 (nominal), and M2 (fastest, highest-error). These modes define a fixed speed-error trade-off. In this benchmark, all machines share the same three selectable processing modes (M0, M1, and M2), and the associated time-scale and error-rate parameters are identical across machines. The exact mode definitions are listed below.

Mode	Interpretation	time scale	error rate
M0	slow / lowest-error	1.2	1.0%
M1	nominal	1.0	1.8%
M2	fast / highest-error	0.8	3.0%

Quality is not modeled as a continuous score in this benchmark. Instead, each job carries a binary defect status. Once any processing step triggers an error, that job is permanently marked defective and cannot pass final inspection. Accordingly, a job passes final inspection if and only if no processing error occurs over its full route. Thus, job-level passing performance is determined by the accumulation of operation-level error events over the full route rather than by a continuous quality score. The preset modes therefore induce a speed-error trade-off at the operation level and a speed-passing-rate trade-off at the job level.

Because the same defect-generation rule and the same mode set are used across all scenarios, the heterogeneity-volatility comparison does not confound machine-capability changes with changes in the defect model. For rule-based boundary references, fixing SPT dispatch and fixing the mode to M0, M1, or M2 yields SPT-M0, SPT-M1, and SPT-M2, respectively.

3.4. Evidence Scope

In this paper, the primary benchmark denotes the controlled seed101 instance used for the main controller comparison: one selected training run for each learning-based controller/scenario pair and five deterministic final-evaluation episodes at the deployment-selected checkpoint. Appendix B reports bounded supplemental sensitivity checks over benchmark-instance seeds 101, 202, and 303 and eight training seeds for each core learned controller; these checks test sensitivity to instance and training seeds but are not pooled with the primary benchmark outcomes.

Benchmark scope and simplifications. The following scope choices are deliberate simplifications adopted to isolate the controller-comparison question from confounding infrastructure-level factors.

- First, AGV transportation is modeled at the node-to-node service level using a fixed transport-time matrix rather than through explicit path planning, collision avoidance, or arc-level

traffic simulation.

- Second, the main controller comparison fixes the job pool within the primary instance. Appendix B adds bounded sensitivity checks over seeds 101, 202, and 303, but this remains a limited instance-seed check rather than a broad instance-generation study.
- Third, all machines share the same three selectable processing modes (M0, M1, M2) with identical time-scale and error-rate parameters, removing machine-specific mode heterogeneity as a confound.
- Lastly, the benchmark uses 100 jobs, 8 machines, and 4 homogeneous AGVs across all scenarios; only the machine-side role distribution and the release-time/task-map volatility profile change between scenarios.

These choices are appropriate for a Phase-I benchmark focused on controller comparison under controlled conditions.

With the benchmark construction and evidence scope defined, the next section introduces the three compared control paradigms under this common setup.

4. COMPARED METHODS

This benchmark compares three aligned control paradigms: a fixed-mode SPT rule baseline family, Centralized PPO, and SO-MARL. The rule-based family uses shortest-processing-time dispatch together with a globally fixed processing mode at each machine-processing action. This yields three fixed-mode variants: SPT-M0, SPT-M1, and SPT-M2, corresponding to the slowest/lowest-error, nominal, and fastest/highest-error modes, respectively. Because these baselines do not adapt mode selection online, they serve as simple operating-policy references that expose the speed-error / passing-rate boundary induced by the preset mode settings. In the core fixed-configuration fair comparison, the primary rule-based baseline is SPT-M2, because M2 uses the fastest processing mode, making it the most efficiency-oriented rule-based reference against the learning-based controllers.

As shown in Figure 3, SPT-M0 and SPT-M1 also expose the full speed–passing-rate envelope of fixed-mode dispatching: SPT-M0 anchors the high-passing-rate / high-makespan end, SPT-M2 anchors the high-speed / low-passing-rate end, and SPT-M1 lies between them. Together, the three fixed-mode points define the rule-based speed–passing-rate envelope against which the learning-based controllers are assessed. Centralized PPO serves as the centralized learning-based controller, and SO-MARL serves as the self-organizing multi-agent controller.

Both learning-based controllers are trained with PPO [21] using the Stable-Baselines3 library [22] on the adapted benchmark environment. The comparison therefore keeps the learning-algorithm family common while contrasting two controller organizations: centralized joint decision making and decentralized self-organizing decision decomposition. Full observation, action, reward, and hyperparameter details are deferred to Appendix A.

Centralized PPO uses a single global policy over the joint decision context and selects from a global TopK+Mode action interface. SO-MARL uses a decentralized dual-PPO decomposition with separate machine-side and AGV-side policy modules, with parameter sharing within homogeneous machine and AGV agent classes. The machine-side policy uses the same candidate-by-mode action encoding as Centralized PPO, whereas the AGV-side policy selects among local transport candidates without mode selection. The environment wrapper filters infeasible joint-action combinations before execution so that independently proposed local actions remain consistent with the manufacturing state.

Accordingly, the reported comparison should be interpreted as controller-family benchmark evidence under a shared evaluation protocol: a single global PPO controller is compared with a decentralized dual-policy PPO controller. It is not a fully architecture-isolated ablation; an interface-matched ablation with aligned observation, reward, and decision interfaces is left for future work.

5. EXPERIMENTAL PROTOCOL AND METRICS

The main controller comparison is kept aligned by using the same primary benchmark instance for all compared methods, so job sets, release-time dynamics, and task-map volatility traces are identical across controllers. Each scenario and learning-based controller is reported from a single training run with a fixed training seed ($seed_{\text{train}} = 101$). All learning-based runs use the same stop rule: training terminates at 300k environment steps, and a checkpoint is recorded every five episodes. Rule-based baselines require no training and are evaluated directly under the same deterministic reporting pipeline. Supplemental sensitivity checks over benchmark-instance seeds 101, 202, and 303 and eight training seeds for each core learned controller are reported separately in Appendix B and are not pooled with the primary five-episode final-evaluation statistics. As discussed in Section 4, SPT-M2 serves as the primary rule-based baseline; SPT-M0 and SPT-M1 are retained as auxiliary boundary references.

Checkpoint handling is aligned across learning-based controllers. Periodic checkpoints are screened into a candidate set (size ≤ 5) to account for the fact that makespan and passing rate need not improve monotonically together. Each candidate checkpoint is evaluated under a fixed five-seed evaluation protocol (five fixed evaluation seeds and a fixed evaluation budget). A single deployment-selected checkpoint is then resolved by a predeclared trade-off rule that prefers lower makespan and higher passing rate. Specifically, candidates are first ranked by lowest makespan, and ties are broken by selecting the highest passing rate. The reported final result for each method/scenario pair is the deterministic five-episode outcome at the deployment-selected checkpoint: makespan as mean value and passing rate as mean ± standard deviation.

Admissibility and validity are defined at the final-evaluation episode level. A result is considered valid for primary cross-method comparison only if all 5/5 fixed final-evaluation episodes terminate successfully within the fixed evaluation budget. Results with partial termination (1–4/5 terminated episodes) or fully invalid final evaluation (0/5 terminated episodes) are not admissible for fair score comparison; they are retained and reported with explicit failure reasons rather than omitted.

For learning-based controllers, training uses a mixed shaped reward aligned with the benchmark objectives. At a high level, the reward combines terminal efficiency and passing-performance terms with shared step-level efficiency shaping and small local

shaping terms for machine-side and AGV-side behavior. Full observation, action, and reward details for SO-MARL are documented in Appendix A.

The reporting contract includes five quantities: (1) makespan — the episode completion time; (2) passing rate — the fraction of jobs passing final inspection; (3) termination rate — the fraction of evaluation episodes terminating within budget; (4) comparability status — a binary gate for admissibility; and (5) failure reason — recorded when a run is not admissible. Reward curves and checkpoint traces are treated as supplementary diagnostics rather than primary benchmark outcomes.

As defined in Section 3.3, each job carries a binary defect status determined by the accumulation of operation-level error events. Under the fixed benchmark instance and deterministic policy execution, evaluation seeds affect only these stochastic defect realizations, not scheduling trajectories. Accordingly, makespan is deterministic across evaluation episodes, while passing rate varies and is reported as mean ± standard deviation.

TABLE 2: Admissibility matrix across the four heterogeneity–volatility regimes ($k/5$ = terminated episodes out of five; shading is visual only).

	S00	S01	S10	S11
SPT-M0	5/5	5/5	5/5	5/5
SPT-M1	5/5	5/5	5/5	5/5
SPT-M2	5/5	5/5	5/5	5/5
Centralized PPO	5/5	5/5	5/5	5/5
SO-MARL	5/5	5/5	5/5	0/5
SO-MARL U01	5/5	0/5	5/5	5/5
SO-MARL U02	0/5	5/5	5/5	5/5

6. RESULTS

This section reports benchmark outcomes under the five-episode evaluation protocol defined in Section 5. Exact numerical values, admissibility summaries, and supplemental sensitivity checks are provided in Appendix B. Benchmark outcomes are interpreted jointly through performance and final-evaluation termination admissibility.

Admissibility-aware efficiency–passing-rate trade-off

Figure 3 provides the main efficiency–passing-rate view for the controlled comparison. The core fixed-configuration fair comparison is between SPT-M2, Centralized PPO, and SO-MARL. Additional SPT-M0/SPT-M1 points are shown as auxiliary rule-based boundary references. Lower makespan is better and higher passing rate is better; therefore, the preferred region is the upper-left.

Figure 3 also includes two diagnostic SO-MARL configuration probes — SO-MARL U01 and SO-MARL U02 — evaluated under the same reporting protocol (exact hyperparameter differences are listed in Table A2 in Appendix A). These probes examine whether modest static hyperparameter changes improve cross-regime admissibility; they are not part of the core fixed-configuration fair controller-to-controller comparison. As reported in Table 2, neither diagnostic variant is uniformly admissible: each improves some regimes but fails in others. These results support the narrower conclusion that the tested static SO-MARL configurations are regime-sensitive and that modest static retuning does not provide a uniformly admissible controller across the four regimes.

Across the four regimes, the core fixed-configuration comparison shows a regime-dependent trade-off rather than a universal winner. On S00, Centralized PPO achieves a slightly lower makespan (2947.4 vs. 3018.0) while matching SO-MARL's passing rate (0.950). On S01, SO-MARL outperforms Centralized PPO on both metrics, achieving a lower makespan (2977.0 vs. 3013.2) and a higher passing rate (0.948 vs. 0.912). On S10, SO-MARL offers the strongest trade-off among core comparison points, achieving the lowest makespan (2327.0) and the highest passing rate (0.950) among the three core methods. On S11, the selected SO-MARL checkpoint is inadmissible (0/5 termination) and therefore excluded from fair score comparison.

From a manufacturing deployment perspective, these trade-offs carry concrete operational implications. A controller that achieves lower makespan at the cost of reduced passing rate effectively trades throughput for higher defect rates — a trade-off whose acceptability depends on product criticality and rework costs. More importantly, an inadmissible controller — one that fails to terminate reliably — corresponds to a potential production stall on the shop floor. Admissibility is therefore treated as a prerequisite for controller selection throughout this benchmark.

Admissibility and comparable outcomes

Following the admissibility protocol defined in Section 5, Table 2 summarizes termination outcomes for all controllers across the four regimes. Full reporting-contract details, including status labels and failure reasons for non-admissible entries, are provided in Appendix B. Taken together, Figure 3 and Table 2 show that controller ranking is regime-dependent and that the tested static SO-MARL configurations expose different reliability boundaries rather than one uniformly admissible setting in the main comparison. In practice, controller choice should be guided by operating regime and performance priority: under S10 (high heterogeneity, low volatility), SO-MARL achieves the best makespan without sacrificing passing rate; Centralized PPO remains admissible across all four regimes, providing a reliable fallback under this reporting contract; and where maximum passing rate is paramount, SPT-M0 provides a deterministic upper bound.

Appendix Tables B.3 and B.4 report supplemental sensitivity checks over benchmark-instance seeds 101, 202, and 303. The admissibility heat table aggregates final-evaluation termination counts across all included instances and, for learned controllers, eight training seeds per instance/scenario; SPT baselines contribute deterministic final-evaluation episodes only. The supplemental checks reinforce the scope boundary: Centralized PPO remains fully admissible across the expanded learned-controller inventory (120/120 per scenario), while SO-MARL remains scenario-sensitive, with lower admissibility in S00 (75/120) and S01 (60/120) than in S10 and S11 (105/120). The performance heat table shows that ranking remains metric- and regime-dependent, so these checks provide robustness context rather than a pooled replacement for the main seed101 outcomes.

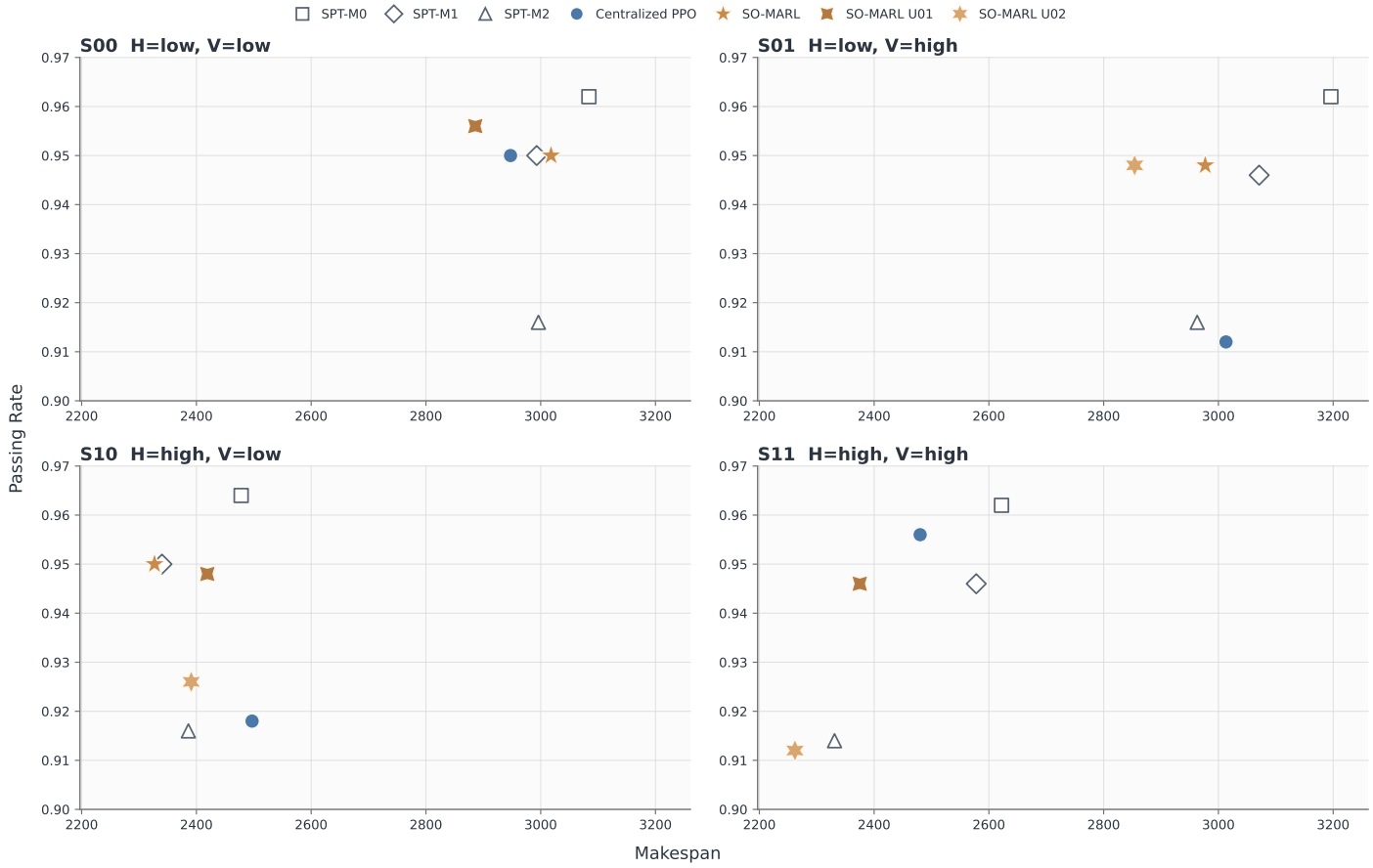

FIGURE 3: 2×2 efficiency–passing-rate trade-off on the fixed benchmark instance (seed101). Core fair comparison: SPT-M2, Centralized PPO, and SO-MARL. SPT-M0 and SPT-M1 are auxiliary rule-based boundary references; SO-MARL U01 and SO-MARL U02 are diagnostic configuration probes. All are reported under the same reporting protocol. Lower makespan and higher passing rate are preferred (upper-left).

Diagnostic configuration sensitivity and training diagnostics

Figure 4 shows supplementary single-run training diagnostics for SO-MARL U02 across the four scenarios. U02 is selected as the illustrative case because it is the only configuration that achieves admissibility on S11 while failing on S00, making its cross-regime convergence contrast the most visually informative. Each curve is plotted as a moving average over a fixed window, and the shaded band indicates the moving standard deviation within the same window. These curves are not used as primary benchmark outcomes and are not interpreted as cross-method comparable quantities.

7. DISCUSSION

The evidence reveals a method-sensitive landscape under aligned settings. In the main comparison, there is no universal winner: ranking changes with operating regime and metric emphasis. Centralized PPO remains consistently admissible but not uniformly superior; it does not dominate all scenarios. Within the core comparison, SO-MARL is competitive on three scenarios, yet the selected checkpoint does not provide a valid final outcome on S11. Controller assessment should therefore be interpreted jointly through makespan, passing rate, and final-evaluation termination validity. The S11 failure of that checkpoint is consistent with the expectation that, under simultaneous high heterogeneity and high volatility, the joint action space expands while environment non-stationarity increases, making it difficult for a static SO-MARL policy to converge to a valid operating regime within the fixed 300k-step budget. The supplemental heat tables in Appendix B qualify this point: across benchmark-instance seeds 101, 202, and 303 and eight training seeds, SO-MARL has valid outcomes in S11 for many learned runs but remains scenario-sensitive overall, with lower admissibility in S00 and S01 than in S10 and S11; Centralized PPO remains fully admissible across the supplemental learned-controller inventory. In SmartSOM terminology, the SO-MARL controller evaluated here is the base, mechanism-free configuration: it does not include higher-level mechanisms such as Task Constraint (TC), Social Learning (SL), or Organizational Learning (OL). Within the broader SmartSOM agenda, these results motivate testing such mechanisms as future regime-aware extensions, but that hypothesis remains to be evaluated directly.

The supplementary training diagnostics for SO-MARL U02 (Figure 4) further illustrate the regime-dependent nature of con-

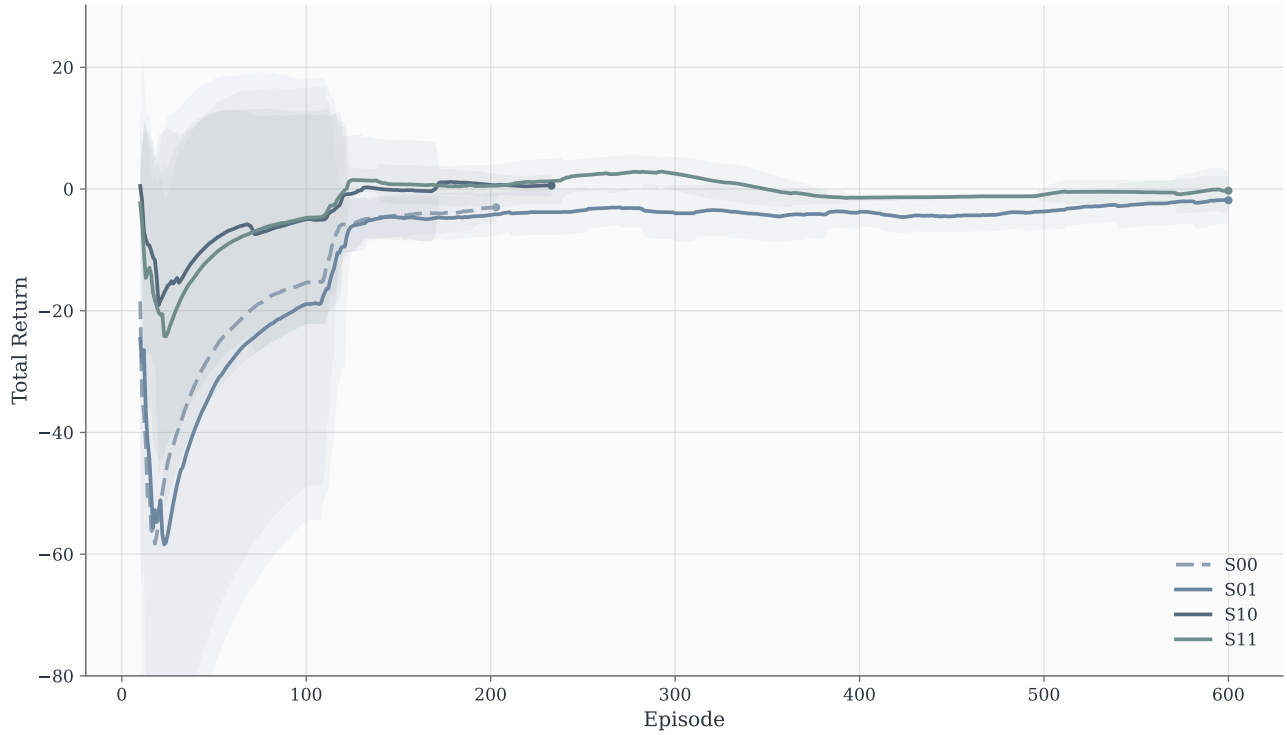

FIGURE 4: Supplementary single-run training diagnostics for SO-MARL U02 across the four scenarios. Lines denote moving-average returns (window = 100 episodes) and shaded regions denote ±1 standard deviation within the same window. This figure is diagnostic only and is not a primary benchmark-outcome figure.

vergence: although U02 achieves admissible outcomes on S11, its plotted return trace shows visibly different convergence behavior across scenarios, with S00 exhibiting the least stable pattern — consistent with U02's 0/5 admissibility failure on that regime. This reinforces the finding that convergence difficulty shifts across regimes rather than concentrating on a single hardest case, and that the tested static diagnostic configurations do not uniformly resolve all four regimes.

Several regime-specific patterns merit further discussion. First, on S10 (high heterogeneity, low volatility), Centralized PPO yields a higher makespan (2497.0) than both SPT-M2 (2386.0) and SO-MARL (2327.0), and a lower passing rate (0.918) than SO-MARL (0.950), while remaining comparable to SPT-M2 (0.916) in passing rate. One possible interpretation is that under high heterogeneity with stable task structure, decentralized machine-level decision-making can exploit local capability matching more effectively than a centralized policy that must jointly optimize over a larger decision space. Second, among the selected checkpoints, SO-MARL achieves a consistently high passing rate (0.948–0.950) across S00, S01, and S10, whereas Centralized PPO's passing rate varies more widely across regimes (from 0.912 on S01 to 0.956 on S11). This pattern suggests that the decentralized machine-side policy may implicitly learn a conservative mode-selection strategy that favors lower-error processing modes, though confirming this hypothesis would require per-step mode-usage analysis that is beyond the scope of the present benchmark study. Third, among the diagnostic variants, SO-MARL U02 achieves the lowest makespan on S11 (2262.0) but at the cost of the lowest passing rate (0.912) among admissible entries, illustrating that aggressive efficiency tuning amplifies the speed–passing-rate trade-off under simultaneous high heterogeneity and high volatility.

The diagnostic configuration matrix in Table 2 sharpens this point by showing that the three SO-MARL configurations fail in different regimes: the main SO-MARL checkpoint fails on S11, SO-MARL U01 fails on S01, and SO-MARL U02 fails on S00. This pattern is more informative than a simple winner–loser comparison because it points to regime-aware adaptation rather than reliance on one fixed static setting.

The present study therefore functions as a Phase-I boundary-finding benchmark: it identifies where the evaluated decentralized controller remains reliable and where additional adaptive mechanisms should be tested.

8. LIMITATIONS

The present study has several limitations that should be considered when interpreting the results. First, the primary benchmark is based on seed101, so the main controller comparison remains fixed-instance evidence rather than broad instance-to-instance generalization. Appendix B adds bounded supplemental sensitivity checks over benchmark-instance seeds 101, 202, and 303, with eight training seeds for each core learned controller and deterministic SPT evaluations. These checks mitigate the single-instance and single-training-seed concerns and show qualitatively similar regime-dependent behavior, but they are still bounded robustness checks rather than broad generalization evi-

dence. Second, as discussed in Section 4, Centralized PPO and SO-MARL share the PPO algorithm family but differ in controller organization and action-interface scope. The comparison should therefore be interpreted as controller-family benchmark evidence rather than a fully architecture-isolated ablation; an interface-matched ablation with aligned observation, reward, and decision interfaces is deferred to future work. Third, the analysis is benchmark-level rather than mechanism-level: explaining why specific controllers succeed or fail under different heterogeneity–volatility regimes will require additional trajectory, mode-usage, and interface-ablation studies. Finally, SmartSOM variants enhanced with TC, SL, or OL are not evaluated here. The diagnostic SO-MARL probes are included as configuration-sensitivity checks, not as part of the core fair comparison.

9. CONCLUSION

This paper presented an aligned Phase-I benchmark study for comparing fixed-mode SPT dispatching, Centralized PPO, and SO-MARL under a controlled 2×2 heterogeneity–volatility scenario matrix. The benchmark shows regime-dependent controller behavior: no method dominates all four scenarios, Centralized PPO provides the most consistent admissibility, and the tested SO-MARL configurations expose regime-specific reliability boundaries. Together, the main comparison and supplemental sensitivity checks support the paper's central contribution: a reproducible protocol for admissibility-aware boundary finding under aligned manufacturing-control regimes.

The contributions are threefold:

- First, it established an aligned benchmark protocol for comparing rule-based control, Centralized PPO, and SO-MARL under a controlled 2×2 heterogeneity–volatility scenario matrix. By aligning the stop rule, checkpointing schedule, candidate-checkpoint selection rule, and deterministic evaluation procedure, the study provides a disciplined basis for cross-method comparison.
- Second, it instantiated that protocol in a reproducible controlled comparison and added bounded supplemental sensitivity checks over benchmark-instance seeds 101, 202, and 303 with eight training seeds for each core learned controller.
- Third, it made admissibility an explicit part of benchmark evaluation by reporting makespan and passing rate together with episode-level termination validity and failure modes. By retaining partial and invalid runs with explicit failure reasons, the study supports a more rigorous assessment of manufacturing controllers in terms of both performance and operational feasibility.

Taken together, the main and supplemental results support the view that aligned benchmarking is necessary for drawing credible cross-method conclusions in smart manufacturing control, especially when both efficiency and product-quality outcomes matter. The evidence should be interpreted within the Phase-I scope summarized in the Limitations.

A natural next step is to extend this benchmark in three directions: (1) larger-scale multi-instance and multi-seed evaluation beyond the bounded supplemental checks reported here, (2) an interface-matched ablation study with aligned observation, reward, and decision interfaces across centralized and decentralized controllers to isolate the effect of control paradigm from interface design, and (3) explicit testing of SmartSOM variants enhanced with TC, SL, and OL for regime-aware adaptation. In that sense, the study should be viewed as a Phase-I benchmark foundation for a broader research program on self-organizing manufacturing control.

10. ACKNOWLEDGEMENTS

The authors acknowledge the support provided by a teaching assistantship from the Department of Aerospace and Mechanical Engineering at the University of Southern California. The authors also thank Dr. Bingling Huang of California State University, Fullerton, for her valuable discussions during the early stages of this research.

11. REFERENCES

[1] Hu, Y., et al. (2024). Industrial internet of things intelligence empowering smart manufacturing: A literature review. *IEEE Internet of Things Journal*, 11(11), 19143–19167.

[2] Weckenborg, C., Schumacher, P., Thies, C., and Spengler, T. S. (2024). Flexibility in manufacturing system design: A review of recent approaches from Operations Research. *European Journal of Operational Research*, 315(2), 413–441.

[3] Pinedo, M. (2016). *Scheduling: Theory, Algorithms, and Systems*. Springer.

[4] Shen, W., Hao, Q., Yoon, H. J., and Norrie, D. H. (2006). Applications of agent-based systems in intelligent manufacturing: An updated review. *Advanced Engineering Informatics*, 20(4), 415–431.

[5] Gui, Y., Zhang, Z., Tang, D., Zhu, H., and Zhang, Y. (2024). Collaborative dynamic scheduling in a self-organizing manufacturing system using multi-agent reinforcement learning. *Advanced Engineering Informatics*, 62, 102646.

[6] Qin, Z., Johnson, D., and Lu, Y. (2023). Dynamic production scheduling towards self-organizing mass personalization: A multi-agent dueling deep reinforcement learning approach. *Journal of Manufacturing Systems*, 68, 242–257.

[7] Park, I.-B., and Park, J. (2023). Scalable scheduling of semiconductor packaging facilities using deep reinforcement learning. *IEEE Transactions on Cybernetics*, 53(6), 3518–3531.

[8] Zhang, Y., et al. (2022). Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems. *Robotics and Computer-Integrated Manufacturing*, 102412.

[9] Waseem, M., and Chang, Q. (2024). From Nash Q-learning to Nash-MADDPG: Advancements in multiagent control for multiproduct flexible manufacturing systems. *Journal of Manufacturing Systems*, 74, 129–140.

[10] Li, C., Chang, Q., and Fan, H.-T. (2024). Multi-agent reinforcement learning for integrated manufacturing system-process control. *Journal of Manufacturing Systems*, 76, 585–598.

[11] Li, Y., Liu, Q., Li, X., and Gao, L. (2025). Manufacturing resource-based self-organizing scheduling using multiagent system and deep reinforcement learning. *Journal of Manufacturing Systems*, 79, 179–198.

[12] Qin, Z., and Lu, Y. (2021). Self-organizing manufacturing network: A paradigm towards smart manufacturing in mass personalization. *Journal of Manufacturing Systems*, 60, 35–47.

[13] Park, H.-S., and Tran, N.-H. (2012). An autonomous manufacturing system based on swarm of cognitive agents. *Journal of Manufacturing Systems*, 31(3), 337–348.

[14] Monostori, L., Vancza, J., and Kumara, S. R. (2006). Agent-based systems for manufacturing. *CIRP Annals*, 55(2), 697–720.

[15] Pulikottil, T., et al. (2021). Multi-agent based manufacturing: current trends and challenges. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–7.

[16] Pulikottil, T., et al. (2023). Agent-based manufacturing—review and expert evaluation. *The International Journal of Advanced Manufacturing Technology*, 127(5), 2151–2180.

[17] Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2), 278–285.

[18] Demirkol, E., Mehta, S., and Uzsoy, R. (1998). Benchmarks for shop scheduling problems. *European Journal of Operational Research*, 109(1), 137–141.

[19] Hoss, J., Schelling, F., and Klarmann, N. (2025). A production scheduling framework for reinforcement learning under real-world constraints. Accepted for presentation at the IEEE 21st International Conference on Automation Science and Engineering (CASE 2025), arXiv:2506.13566. JobShopLab repository: https://github.com/proto-lab-ro/jobshoplab.

[20] Vis, I. F. A. (2006). Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170(3), 677–709.

[21] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[22] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268), 1–8.

APPENDIX A. PROTOCOL DETAILS AND REPRODUCIBILITY SETTINGS

A.1 Training and checkpoint protocol

- Primary benchmark instance seed: seed101.
- Primary benchmark training seed for each learning-based controller: $seed_{\text{train}} = 101$ (single run per method/scenario).
- Supplemental sensitivity checks: benchmark-instance seeds 101, 202, and 303, with training seeds 11, 22, 33, 44, 55, 66, 77, and 88 for each core learned controller (Centralized PPO and SO-MARL) on each scenario/instance pair. Rule-based SPT baselines require no training and are evaluated deterministically on the same three benchmark-instance seeds.
- Stop rule: training terminates at 300k environment steps.
- Checkpoint frequency: one checkpoint every 5 episodes.
- Candidate set size: up to 5 checkpoints per run.
- SO-MARL configurations: SO-MARL, SO-MARL U01, and SO-MARL U02. All use the same training budget and checkpoint-selection protocol.

A.2 Candidate checkpoint formation

For each training seed, periodic training records are used to form a candidate checkpoint set. At each evaluation event, a checkpoint is admitted into the retained set only if it is valid for comparison (i.e., all five fixed final-evaluation episodes terminate successfully). The retained set is then pruned to at most five checkpoints by the same lexicographic order used for final deployment selection: makespan ascending, then passing rate descending. No manual selection is used. If no valid checkpoint exists for a given method/scenario pair, the retained set is empty and the entry is reported as inadmissible with no score.

A.3 Final deterministic evaluation and validity labels

- Deterministic evaluation seed list (fixed): {10000000, 10000001, 10000002, 10000003, 10000004}.
- Fixed final-evaluation budget: 4000.
- Each deployment-selected checkpoint is evaluated once on the same five fixed seeds under this fixed budget; the reported final result is this deterministic five-episode outcome.
- valid (admissible): 5/5 fixed final-evaluation episodes terminate successfully.
- partial: 1–4/5 fixed final-evaluation episodes terminate successfully.
- invalid: 0/5 fixed final-evaluation episodes terminate successfully.
- Partial and invalid outcomes are excluded from primary score comparison but retained in failure reporting.

A.4 AGV transport-time matrix

Table A1 provides the fixed AGV transport-time matrix described in Section 3.3.

TABLE A1: Fixed AGV transport-time matrix

Node	IB	OB	M1	M2	M3	M4	M5	M6	M7	M8
IB	0	24	6	10	12	16	18	22	24	28
OB	24	0	22	26	16	20	10	14	4	8
M1	6	22	0	4	6	10	12	16	18	22
M2	10	26	4	0	10	6	16	12	22	18
M3	12	16	6	10	0	4	6	10	12	16
M4	16	20	10	6	4	0	10	6	16	12
M5	18	10	12	16	6	10	0	4	6	10
M6	22	14	16	12	10	6	4	0	10	6
M7	24	4	18	22	12	16	6	10	0	4
M8	28	8	22	18	16	12	10	6	4	0

Rows and columns correspond to the input buffer, output buffer, and machine service nodes. Each entry gives the deterministic AGV travel time between node *i* and node *j*.

A.5 Implementation details

The implementation uses the same simulator configuration, job set, release-time dynamics, and task-map volatility trace for all compared methods in the primary benchmark. Additional implementation settings are summarized in the accompanying appendix tables. Centralized PPO used Stable-Baselines3 PPO with Adam (lr = 1×10^{-4}, batch_size = 64, ent_coef = 0.01, n_steps = 2048, gamma = 0.99, gae_lambda = 0.95, 10 PPO epochs), together with a global TopK+Mode interface (top_k = 8, num_modes = 3) and a total training budget of 300,000 timesteps. SO-MARL used two SB3 PPO modules with Adam and parameter sharing within homogeneous machine and AGV agent classes (lr = 3e-4, batch_size = 64, ent_coef = 0.0, gamma = 0.99, gae_lambda = 0.95, 10 PPO epochs, rollout_steps = 64), with machine-side local top_k = 8 and AGV-side local top_k = 4, under the common training budget of 300,000 environment steps. The diagnostic SO-MARL variants differ from the base configuration only in the hyperparameters listed in Table A2. All other settings are identical to the base configuration described above.

TABLE A2: SO-MARL configuration variants.

Config	lr	ent_coef	no_progress_step_limit
SO-MARL	3×10^{-4}	0.0	400
SO-MARL U01	5×10^{-4}	0.0	600
SO-MARL U02	3×10^{-4}	0.005	600

A.6 SO-MARL policy decomposition and reward design

This subsection documents the implementation-facing policy decomposition and reward design for SO-MARL. The intent is to make policy ownership, decision interfaces, and reward semantics explicit for reproducibility and review.

Machine-side observation. The machine-side PPO receives a dictionary observation. The fields are: job_running (J), job_executed_on_machine ($J \times M$), job_progression (J), machine_running (M), machine_progression (M), available_jobs (J), current_time (1), global_time (1), current_transition (3), local_state, machine_prebuffer_len (1), machine_postbuffer_len (1), machine_prebuffer_head_job (1), machine_postbuffer_head_job (1), machine_current_job (1), machine_remaining_time (1), global_input_buffer_fill (1), global_output_buffer_fill (1), starving_machine_ratio (1), candidate_mask (K_m), candidate_job (K_m), candidate_remaining_ops (K_m), and candidate_next_op_duration (K_m).

Binary status features remain 0/1. Time-like quantities are normalized by max_allowed_time, and buffer lengths are normalized by buffer capacity. Job identifiers are encoded as job_id/num_jobs, with -1 used for padding or missing values. candidate_remaining_ops is normalized by max_ops_per_job, and candidate_next_op_duration is normalized by max_operation_duration. The current_transition field encodes three normalized scalars: component_index/num_components, job_index/num_jobs, and component_type_code. The component-type code uses $m = 0$, $t = 0.33$, and $b = 0.66$.

Machine-side action. The machine-side PPO uses the same candidate-by-mode flattened action encoding as Centralized PPO. A candidate list of up to $K_m = 8$ base candidates is formed from currently valid local machine-side transitions, and the action space is Discrete($K_m \times$ num_modes + 1) = Discrete(25), where num_modes = 3. Action 0 denotes no-op. For actions 1–24, the action index is decoded as: candidate_slot = (action $-$ 1)//num_modes, mode_index = (action$-$1)%num_modes. This jointly selects which candidate transition to execute and which processing mode to use.

Machine-side feasibility handling. Only the first $K_m = 8$ currently valid transitions for that machine are exposed as base candidates. If no machine-side candidate exists, the training loop forces action 0 and does not add that agent's sample to PPO for that step. Because the action space is fixed-size, the decoded candidate_slot is further wrapped by candidate_slot mod len(candidates) to map to one current valid candidate.

AGV-side observation. The AGV-side PPO receives job_running (J), job_executed_on_machine ($J \times M$), job_progression (J), machine_running (M), machine_progression (M), available_jobs (J), current_time (1), global_time (1), current_transition (3), local_state, has_payload (1), agv_current_job (1), agv_remaining_time (1), global_input_buffer_fill (1), global_output_buffer_fill (1), starving_machine_ratio (1), candidate_mask (K_a), candidate_job (K_a), candidate_remaining_ops (K_a), and candidate_next_op_duration (K_a). The same normalization rules as the machine side are used. AGV local state is encoded categorically, and payload/current-job information is exposed explicitly.

AGV-side action. The AGV-side PPO uses Discrete(K_a+1) with $K_a = 4$ by default, i.e., Discrete(5). Action 0 denotes no-op. Actions 1–4 select one currently valid AGV transport candidate.

AGV-side feasibility handling. Only valid local AGV transitions are surfaced to the policy. After both policies act, the wrapper keeps at most one transition per job and greedily pre-validates joint transitions on a temporary state to prune inconsistent multi-agent combinations before stepping the base environment.

Parameter sharing / policy ownership. All machine agents share one PPO module (machine-side policy), and all AGV agents share another PPO module (AGV-side policy). Parameters are therefore shared within homogeneous agent classes, but machine-side and AGV-side policies remain distinct modules. The machine-side policy owns machine dispatch decisions, whereas the AGV-side policy owns transport-assignment and transport-execution decisions. There is no third joint coordinator policy.

Reward definition. For learning-based controllers, the training signal uses a shaped reward aligned with benchmark objectives.

At every step, all agents receive a shared shaping term $r_{\text{shared}} = 0.1\,(\Delta N_{\text{done}}/N_{\text{ops}} - \Delta t/T_{\max})$, where ΔN_{done} is the increase in completed operations and Δt is the event-time advance.

Each machine agent receives a local penalty when its postbuffer stays non-empty beyond a grace period. The trigger condition is: the postbuffer contains jobs and the accumulated wait exceeds 2 time units. The applied penalty on that step is $-0.01\,\Delta t$.

An AGV receives +0.05 when it was carrying a job before the step, unloads it in the current step, and that delivery fills the prebuffer of a machine that was starving in the previous state.

The SO-MARL wrapper adds an extra −0.1 to all agents if wrapper-level no-progress truncation fires. In the present benchmark configuration, the trigger is 400 consecutive wrapper steps without any increase in completed operations.

The training loop includes an explicit invalid-joint-action fallback penalty parameter, but the SO-MARL runs reported in this benchmark leave it at the default 0.0. Therefore, invalid joint actions cause reset/truncation handling without an additional nonzero penalty.

The following reward weights are identical across S00, S01, S10, and S11: sparse_bias = 1.0, dense_bias = 0.001, truncation_bias = −1.0, quality_weight = 1.0, machine_wait_penalty = 0.01, and machine_wait_grace_time = 2. The same configuration sets agv_starving_delivery_bonus = 0.05, global_shared_weight = 0.1, no_progress_step_limit = 400, no_progress_truncation_penalty = −0.1, and invalid_joint_action_penalty = 0.0. These weights were chosen so that terminal objectives (makespan and passing rate) dominate the reward signal, while step-level shaping terms remain at least an order of magnitude smaller to provide learning guidance without overriding the terminal objectives.

The reward structure and policy decomposition are identical across S00, S01, S10, and S11. What changes across scenarios are the compiled instance-derived normalization constants and the feasible candidate sets. In the present benchmark instance, S00 and S10 compile to num_jobs = 100, num_operations = 283, max_allowed_time = 8171, and lower_bound = 2377; S01 and S11 compile to num_jobs = 100, num_operations = 291, max_allowed_time = 8393, and lower_bound = 2494. Shared instance-scaling constants across all four scenarios are num_machines = 8, num_agvs = 4, max_ops_per_job = 4, max_operation_duration = 50, num_components = 12, machine candidate cap $K_m = 8$, and AGV candidate cap $K_a = 4$.

Accordingly, reward weights and policy structure are fixed across the four benchmark scenarios; only instance-derived scaling constants and feasible candidate sets vary with the compiled scenario instance.

A.7 Benchmark instance specification

Each job j in the fixed benchmark instance is represented as $(\text{route}_j, \mathbf{p}_j, r_j)$, where route_j is the ordered operation-type sequence, $\mathbf{p}_j$ is the operation-time vector along that route, and r_j is the release time. The instance is deterministically generated from seed101 and contains 100 jobs. For space reasons, the full per-job listing is not reproduced in the manuscript. A representative job may be written as $j = ([\text{O1}, \text{O2}, \text{O4}, \text{O3}], [p_1, p_2, p_3, p_4], r_j)$. The complete seed101 job list is provided in the supplementary material for reproduction.

APPENDIX B. EXACT NUMERICAL RESULTS AND FINAL-EVALUATION ADMISSIBILITY

TABLE B.1: Exact numerical results on the fixed benchmark instance (seed101). For learned-controller rows, the seed column reports the training seed.

SCN	Method	Train Seed	MSPAN	Pass Rate	Final Eval
S00	SPT-M0	N/A	3084.0	0.962 ± 0.018	5/5
S00	SPT-M1	N/A	2993.0	0.950 ± 0.016	5/5
S00	SPT-M2	N/A	2996.0	0.916 ± 0.038	5/5
S00	Centralized PPO	101	2947.4	0.950 ± 0.019	5/5
S00	SO-MARL	101	3018.0	0.950 ± 0.016	5/5
S00	SO-MARL U01	101	2886.0	0.956 ± 0.012	5/5
S00	SO-MARL U02	101	N/A	N/A	0/5
S01	SPT-M0	N/A	3196.0	0.962 ± 0.018	5/5
S01	SPT-M1	N/A	3071.0	0.946 ± 0.015	5/5
S01	SPT-M2	N/A	2963.0	0.916 ± 0.038	5/5
S01	Centralized PPO	101	3013.2	0.912 ± 0.040	5/5
S01	SO-MARL	101	2977.0	0.948 ± 0.016	5/5
S01	SO-MARL U01	101	N/A	N/A	0/5
S01	SO-MARL U02	101	2854.0	0.948 ± 0.016	5/5
S10	SPT-M0	N/A	2478.0	0.964 ± 0.018	5/5
S10	SPT-M1	N/A	2340.0	0.950 ± 0.016	5/5
S10	SPT-M2	N/A	2386.0	0.916 ± 0.038	5/5
S10	Centralized PPO	101	2497.0	0.918 ± 0.041	5/5
S10	SO-MARL	101	2327.0	0.950 ± 0.016	5/5
S10	SO-MARL U01	101	2419.0	0.948 ± 0.019	5/5
S10	SO-MARL U02	101	2391.0	0.926 ± 0.034	5/5
S11	SPT-M0	N/A	2622.0	0.962 ± 0.018	5/5
S11	SPT-M1	N/A	2578.0	0.946 ± 0.015	5/5
S11	SPT-M2	N/A	2331.0	0.914 ± 0.040	5/5
S11	Centralized PPO	101	2480.2	0.956 ± 0.023	5/5
S11	SO-MARL	101	N/A	N/A	0/5
S11	SO-MARL U01	101	2375.0	0.946 ± 0.019	5/5
S11	SO-MARL U02	101	2262.0	0.912 ± 0.038	5/5

Passing rate is reported as mean ± sample standard deviation over five fixed final-evaluation seeds.

Final eval is reported as the number of terminated episodes out of five fixed final-evaluation seeds (k/5).

Header legend: SCN = scenario, MSPAN = makespan, Pass rate = passing rate, and Final Eval = final terminated episodes out of 5.

TABLE B.2: Failure reporting for non-admissible entries

Method	SCN	Final Eval	Status	Failure Mode
SO-MARL	S11	0/5	invalid	no-progress truncation†
SO-MARL U01	S01	0/5	invalid	no-progress truncation‡
SO-MARL U02	S00	0/5	invalid	no-progress truncation‡

† Base configuration with `no_progress_step_limit = 400`.

‡ Diagnostic configuration with `no_progress_step_limit = 600`.

All three entries were invalid because all five deterministic final-evaluation episodes were truncated by the no-progress rule before successful termination.

TABLE B.3: Supplemental three-instance final-evaluation admissibility heat table. This robustness check is reported separately from the primary seed101 results in Table B.1. SPT rows aggregate three deterministic benchmark-instance evaluations (3 instance seeds × 5 final-evaluation episodes = 15). Learned-controller rows aggregate three benchmark instances, eight training seeds per instance/scenario, and five final-evaluation episodes per training seed (3 × 8 × 5 = 120). Invalid or truncated learned-controller runs remain visible in the admissibility count.

	S00	S01	S10	S11
SPT-M0	15/15	15/15	15/15	15/15
SPT-M1	15/15	15/15	15/15	15/15
SPT-M2	15/15	15/15	15/15	15/15
Centralized PPO	120/120	120/120	120/120	120/120
SO-MARL	75/120	60/120	105/120	105/120

TABLE B.4: Supplemental three-instance performance heat table. Makespan and passing rate are reported as mean ± standard deviation over the same three benchmark-instance sensitivity inventory. Learned-controller metric summaries use valid trained policies, while invalid or truncated policies are retained in the admissibility accounting in Table B.3. Cell colors rank methods within each scenario and metric only.

		S00	S01	S10	S11
MSPAN	SPT-M0	3033.0 ± 47.9	3281.0 ± 146.4	2545.7 ± 60.4	2619.0 ± 11.8
	SPT-M1	2990.3 ± 42.1	3046.7 ± 21.8	2402.7 ± 89.4	2554.0 ± 47.8
	SPT-M2	2968.3 ± 30.9	3048.3 ± 75.6	2335.7 ± 45.3	2321.3 ± 24.0
	Centralized PPO	3036.8 ± 15.1	3035.9 ± 36.8	2711.1 ± 5.3	2770.0 ± 52.0
	SO-MARL	3059.5 ± 70.9	3234.6 ± 178.3	2384.8 ± 22.2	2466.6 ± 58.7
Pass Rate	SPT-M0	0.963 ± 0.001	0.961 ± 0.001	0.963 ± 0.001	0.961 ± 0.001
	SPT-M1	0.951 ± 0.001	0.946 ± 0.002	0.951 ± 0.001	0.947 ± 0.002
	SPT-M2	0.917 ± 0.001	0.915 ± 0.002	0.917 ± 0.001	0.913 ± 0.004
	Centralized PPO	0.934 ± 0.007	0.926 ± 0.001	0.943 ± 0.003	0.934 ± 0.007
	SO-MARL	0.939 ± 0.011	0.940 ± 0.014	0.936 ± 0.001	0.939 ± 0.008