

Descent-Guided Policy Gradient for Scalable Cooperative Multi-Agent Learning

Shan Yang^{‡,*}, Yang Liu^{‡,*}

National University of Singapore

[‡]yang.shan@u.nus.edu, [‡]ceelya@nus.edu.sg

Abstract

Scaling cooperative multi-agent reinforcement learning (MARL) is fundamentally limited by cross-agent noise. When agents share a common reward, each agent’s learning signal is computed from a shared return that depends on all agents, so the stochasticity of the other agents enters the signal as cross-agent noise that grows with N . Fortunately, many engineering systems, such as cloud computing and power systems, have differentiable analytical models that prescribe efficient system states, providing a new reference beyond noisy shared returns. In this work, we propose Descent-Guided Policy Gradient (DG-PG), a framework that augments policy-gradient updates with a noise-free descent signal derived from differentiable analytical models. We prove that DG-PG reduces policy-gradient estimator variance from $\mathcal{O}(N)$ to $\mathcal{O}(1)$, preserves the equilibria of the cooperative game, and achieves agent-independent sample complexity $\tilde{\mathcal{O}}(1/\epsilon)$. On a heterogeneous cloud resource scheduling task with up to 1500 agents, DG-PG converges within 20 episodes on average, while MAPPO and IPPO fail to converge under identical architectures.

1 Introduction

Multi-agent reinforcement learning (MARL) has demonstrated remarkable success across a wide range of domains, from game playing [30] to cloud resource scheduling [15] and transportation management [5]. In these applications, an agent represents an individual decision-making entity, such as a player in a game, a job dispatcher in cloud scheduling, or a vehicle in a road network. When agents share a common objective and must coordinate their actions, the setting is referred to as cooperative MARL. The training difficulty of cooperative MARL grows rapidly with the number of agents N , stemming from multi-agent credit assignment: when all agents jointly contribute to the shared objective, no individual agent’s contribution can be cleanly identified.

Specifically, in cooperative MARL, all N agents share a single return, defined as the cumulative team reward, and compute a learning signal to update their policies. Regardless of the form of the signal, a gradient direction for policy-gradient methods or an advantage for actor-critic methods, the signal always carries cross-agent noise that grows with N . [11] formalize this in the policy-gradient setting, showing that the noise contributes $\mathcal{O}(N)$ variance to the per-agent gradient. Thus, to make cooperative MARL scalable, this cross-agent noise must be addressed.

A common approach to alleviating cross-agent noise is credit decomposition, which extracts per-agent contributions from the shared return, thereby reducing the variance of the per-agent gradient estimator. This is typically realized by subtracting a counterfactual baseline [6], factorizing the joint Q into per-agent Q -functions [20, 25], or estimating each agent’s marginal contribution [27]. However, in these methods, the cross-agent noise still grows with N , dominating the per-agent signal at scale.

*Corresponding authors.

Fortunately, many cooperative engineering systems admit differentiable analytical models that prescribe efficient system states. For example, queueing models characterize how workloads should be balanced across servers in cloud scheduling, while power-flow equations characterize how generation should meet demand subject to network constraints in power systems. However, these prescriptions remain system-level descriptions and do not directly yield local decision rules for individual agents. This raises a central question: can the differentiable structure of analytical references be converted into agent-local descent directions that reduce the variance of cooperative policy-gradient learning?

Building on this insight, we propose Descent-Guided Policy Gradient (DG-PG), a framework that integrates such analytical models into the multi-agent policy gradient. Our key contributions are:

- We derive noise-free per-agent descent guidance by differentiating the deviation from the model-prescribed reference with respect to agent’s action. We augment the policy gradient with this guidance, accelerating learning without architectural changes to existing methods.
- We prove that: (i) the augmented gradient does not alter the Nash equilibria of the original cooperative game (Theorem 5.1); (ii) the variance of the per-agent gradient estimator reduces from $\mathcal{O}(N)$ to $\mathcal{O}(1)$, independent of the number of agents (Theorem 5.2); and (iii) the sample complexity to reach an ϵ -optimal solution is $\tilde{\mathcal{O}}(1/\epsilon)$ (Theorem 5.3).
- We evaluate DG-PG on heterogeneous cloud scheduling with up to 1500 agents, demonstrating scale-invariant convergence and outperforming carefully tuned MAPPO and IPPO baselines under matched architectures.

2 Related Work

Classical methods address the credit assignment problem in cooperative MARL, typically by extracting per-agent contributions from the shared return. Difference Rewards [27] isolate each agent’s marginal contribution via counterfactual simulation, but require access to the reward function or a simulator. Value decomposition methods, such as VDN [23], QMIX [20], and QPLEX [25], factorize the joint Q-function into per-agent value contributions under structural assumptions (additivity, monotonicity), but fail on tasks whose optimal joint value violates these assumptions. COMA [6] computes per-agent counterfactual advantages from a centralized critic, with the critic conditioned on the joint action of all agents. Crucially, these methods still extract per-agent signals from the shared return, leaving each signal carrying cross-agent noise that grows with N . DG-PG instead obtains a noise-free descent direction via an analytical reference model and augments each agent’s policy gradient with it, enabling stable per-agent learning that does not degrade as N grows.

Another line of work addresses scalability in cooperative MARL through structural simplifications, typically restricting either how agents update or how they interact. HAPPO [12] restricts agent updates to a sequential order, so that each agent optimizes against fixed teammate policies and can obtain its marginal advantage. This is effective but sacrifices parallel execution, as each agent must wait for all preceding updates. Mean-Field MARL [29] takes a different approach, replacing the N -agent interaction with a population-level approximation that simplifies each agent’s optimization problem. This achieves scalability but requires agent homogeneity, precluding application to heterogeneous settings. DG-PG, in contrast, requires no structural change to how agents update or interact.

Concurrently, other lines of work also incorporate prior knowledge into RL training, but in forms different from DG-PG. First, Guided Policy Search [14] and Residual RL [8] derive a base policy from domain knowledge and optimize from there. However, they still suffer from cross-agent noise at large N , so the policy cannot improve substantially beyond the base. Second, Physics-Informed Neural Networks [19] encode prior knowledge as a constraint in the loss, rather than as a basis for learning or a direction for optimization. In contrast, DG-PG harnesses prior knowledge of the system to construct noise-free gradient guidance, significantly reducing noise in the gradient direction and accelerating learning without altering the optimal policy. We refer the reader to Appendix A for a complete comparison.

3 Preliminaries

We consider a cooperative MARL setting, formalized as a *Cooperative Markov Game* (equivalently, a Dec-POMDP) [18]. The game is defined by the tuple $\mathcal{M} = \langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, where $\mathcal{N} =$

$\{1, \dots, N\}$ denotes the set of N agents, $\mathcal{S}$ is the global state space, and $\mathcal{A} = \times_{i \in \mathcal{N}} \mathcal{A}^i$ is the joint action space with $\mathcal{A}^i$ is agent i 's action space. At each timestep t , the system is in state $s_t \in \mathcal{S}$. Each agent i receives a local observation o_t^i (derived from s_t) and selects an action $a_t^i \in \mathcal{A}^i$ according to its policy $\pi_{\theta^i}(a_t^i | o_t^i)$, parameterized by θ^i . The environment then transitions to state s_{t+1} according to the dynamics $P(s_{t+1} | s_t, \mathbf{a}_t)$, where $\mathbf{a}_t = (a_t^1, \dots, a_t^N)$ denotes the joint action.

3.1 Cooperative Objective and Policy Gradient

In the cooperative setting, all agents share a common global reward function $r(s_t, \mathbf{a}_t)$. The shared objective is to maximize the expected discounted cumulative reward:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \mathbf{a}_t) \right], \quad (1)$$

where $\gamma \in [0, 1)$ is the discount factor and $\boldsymbol{\pi}$ denotes the joint policy. Standard policy gradient methods perform gradient ascent on $J(\boldsymbol{\theta})$. For agent i , the gradient with respect to its parameters θ^i is given by the Policy Gradient Theorem [24]:

$$\nabla_{\theta^i} J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t \nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i | o_t^i) R_t \right], \quad (2)$$

where $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the sampled return from time t .

3.2 The Variance Explosion Problem

While Equation 2 provides an unbiased gradient estimator, its variance poses a critical challenge in large-scale settings. Because the estimator relies on sampled returns that depend on the joint action $\mathbf{a}_t$ of all N agents, each agent's gradient estimate carries noise from every other agent's sampled actions. Following [11], this variance can be approximately decomposed as:

$$\text{Var}(\hat{g}^i) \approx \underbrace{\sigma_{\text{self}}^2}_{\text{Own Exploration}} + \underbrace{(N-1)\sigma_{\text{others}}^2}_{\text{Cross-Agent Noise}}. \quad (3)$$

The second term represents the noise from sampling other agents' actions. As N increases, this term dominates, causing the total estimation variance to scale as $\mathcal{O}(N)$. This scaling implies that maintaining a constant estimation error requires the batch size to grow with N , rendering *standard multi-agent policy gradient methods sample-inefficient for large-scale systems*.

4 Method: Descent-Guided Policy Gradient

In this section, we present DG-PG, a framework that addresses variance explosion by exploiting the analytical models available in many engineering systems. We first define the *reference state* as an efficient system state prescribed by analytical models and identify the conditions under which it yields a valid guidance term (Section 4.1). We then augment the policy gradient with a noise-free, per-agent signal derived from the reference (Section 4.2). Under mild assumptions, DG-PG does not alter the Nash equilibria of the cooperative game while achieving agent-independent variance.

4.1 Utilizing Analytical Priors: System State and Reference

In many engineering systems, analytical models from operations research and control theory, such as fluid-limit approximations for queueing systems or equilibrium analysis for traffic networks, provide principled characterizations of desirable system behavior. While not executable under real-world stochasticity, these models offer something valuable, namely a *reference* that describes what efficient system operation looks like under given conditions. Our key observation is that such a reference, if properly defined, can supply each agent with a clear, directed signal for policy improvement. This signal is far more informative than the noisy scalar reward r_t aggregated over all agents.

As noted above, domain-specific models can prescribe efficient system states, but these states must be expressed in a concrete representational space. We define a *system state* $\mathbf{x}_t \in \mathbb{R}^m$ as the vector of

system-level quantities jointly determined by agents' actions (e.g., per-server resource utilization, per-link flow, or queue lengths), where m is the number of monitored quantities. The *reference state* $\tilde{\mathbf{x}}_t \in \mathbb{R}^m$ is the efficient state that the analytical model yields under current conditions, computed independently of the learned policies. A key structural property of $\mathbf{x}_t$ is that each agent's action typically affects only specific dimensions of this vector, so that each agent's influence on the system state is localized. We exploit this property in the gradient computation (Section 4.2).

For the reference to yield a valid guidance signal, it should ideally satisfy two conditions:

Assumption 4.1 (Exogeneity). The reference state $\tilde{\mathbf{x}}_t$ does not depend on the policy parameters θ :

$$\nabla_{\theta} \tilde{\mathbf{x}}_t = \mathbf{0}. \quad (4)$$

Assumption 4.2 (Descent-Aligned Reference). Moving the system state toward the reference improves system performance. Formally, for any $\mathbf{x}_t \neq \tilde{\mathbf{x}}_t$:

$$\langle \nabla_{\mathbf{x}} J(\mathbf{x}_t), \tilde{\mathbf{x}}_t - \mathbf{x}_t \rangle > 0. \quad (5)$$

When both conditions hold, the reference provides a stable, directionally correct target that does not shift as policies update (Assumption 4.1), and approaching it is expected to improve the cooperative objective (Assumption 4.2). This means that any policy update that steers the system state toward $\tilde{\mathbf{x}}_t$ is aligned with the original optimization goal, a property we exploit in the next subsection to construct a low-variance gradient estimator. Both assumptions are broadly satisfied in structured engineering systems, and we verify them concretely for the cloud scheduling case in Appendix C

Importantly, $\tilde{\mathbf{x}}_t$ need not be reachable, because it may arise from idealized fluid-limit or steady-state models. DG-PG uses $\tilde{\mathbf{x}}_t$ only to define a descent direction, not as a target to be attained exactly.

4.2 Descent-Guided Policy Gradient

Given a reference $\tilde{\mathbf{x}}_t$ satisfying Assumptions 4.1–4.2, steering the system toward $\tilde{\mathbf{x}}_t$ at each step is directionally aligned with improving J . The challenge is to incorporate this information into the learning process in a way that (i) does not alter the original cooperative objective, and (ii) provides each agent with a gradient signal free of cross-agent noise. Our DG-PG achieves both by augmenting the policy gradient with a per-agent guidance term derived from the analytical model.

We construct DG-PG in three steps: define a deviation measure between the system state and the reference, differentiate it to obtain per-agent descent directions, and augment the standard policy gradient with these directions to eliminate cross-agent noise.

Deviation Measure. We first define a per-step deviation function to quantify how far the system state deviates from the reference:

$$d(\mathbf{x}_t, \tilde{\mathbf{x}}_t) = \frac{1}{2} \|\mathbf{x}_t - \tilde{\mathbf{x}}_t\|^2. \quad (6)$$

By Assumption 4.2, reducing d at each step is directionally consistent with improving J . However, simply incorporating d into the reward would yield a scalar signal that cannot distinguish each agent's individual contribution to the deviation. Since d is differentiable with respect to $\mathbf{x}_t$ and $\mathbf{x}_t$ depends differentially on each a_t^i , the gradient $\partial d / \partial a_t^i$ can be computed analytically. Thus, the gradient of d can serve as a guidance signal for each agent, aligned with improving J .

DG-PG Gradient. We augment the standard policy gradient with the gradient of d , yielding the DG-PG gradient:

$$\nabla_{\theta^i} J_{\alpha} \triangleq (1 - \alpha) \nabla_{\theta^i} J - \alpha \nabla_{\theta^i} \mathcal{G}, \quad (7)$$

where $\alpha \in (0, 1)$ is the guidance weight, and $\mathcal{G} \triangleq \mathbb{E}_{s \sim \nu^{\pi}} [d(\mathbf{x}_t, \tilde{\mathbf{x}}_t)]$ is the expected deviation under the state visitation distribution ν^{π} induced by the joint policy. The DG-PG gradient $\nabla_{\theta^i} J_{\alpha}$ thus retains the optimization direction of J while incorporating a guidance signal from the reference model. We prove that this augmentation preserves the Nash equilibria of the original game (Theorem 5.1).

To evaluate $\nabla_{\theta^i} \mathcal{G}$, we apply the chain rule through $\mathbf{x}_t$ to isolate each agent's contribution to the deviation:

$$\frac{\partial d}{\partial a_t^i} = \left\langle \nabla_{\mathbf{x}} d, \frac{\partial \mathbf{x}_t}{\partial a_t^i} \right\rangle = \langle \mathbf{x}_t - \tilde{\mathbf{x}}_t, \mathbf{z}_t^i \rangle, \quad (8)$$

where $\mathbf{z}_t^i \triangleq \frac{\partial \mathbf{x}_t}{\partial \mathbf{a}_t^i}$ is agent i 's *local influence vector*, the partial derivative of the system state with respect to its action. This coefficient measures the deviation projected onto the dimensions that agent i influences. It depends only on the current state $\mathbf{x}_t$, the exogenous reference $\tilde{\mathbf{x}}_t$, and the agent-local derivative $\mathbf{z}_t^i$, so it carries no cross-agent noise and can be computed analytically. The concrete form of $\mathbf{z}_t^i$ depends on the domain, and we derive it for cloud scheduling in Appendix D.

Substituting Equation 8 into the policy gradient theorem yields the gradient of the guidance term:

$$\nabla_{\theta^i} \mathcal{G} = \mathbb{E}_{s \sim \nu^\pi, \mathbf{a} \sim \pi} [\langle \mathbf{x}_t - \tilde{\mathbf{x}}_t, \mathbf{z}_t^i \rangle \nabla_{\theta^i} \log \pi^i(a_t^i | o_t^i)]. \quad (9)$$

Gradient Estimator. In practice, we estimate the DG-PG gradient $\nabla_{\theta^i} J_\alpha$ via the score function method. At each step t of a sampled trajectory, the gradient estimate for agent i is:

$$\hat{g}_{\text{DG},t}^i = (1 - \alpha) \hat{g}_{J,t}^i - \alpha \hat{g}_{\mathcal{G},t}^i, \quad (10)$$

where

$$\hat{g}_{J,t}^i = R_t \nabla_{\theta^i} \log \pi^i(a_t^i | o_t^i), \quad (11)$$

$$\hat{g}_{\mathcal{G},t}^i = \langle \mathbf{x}_t - \tilde{\mathbf{x}}_t, \mathbf{z}_t^i \rangle \nabla_{\theta^i} \log \pi^i(a_t^i | o_t^i). \quad (12)$$

Each agent then updates its policy parameters via gradient ascent with learning rate $\eta > 0$:

$$\theta_{k+1}^i = \theta_k^i + \eta \hat{g}_{\text{DG},t}^i, \quad (13)$$

Variance Properties. The two components of $\hat{g}_{\text{DG},t}^i$ have fundamentally different variance properties. The standard policy gradient sample $\hat{g}_{J,t}^i$ uses the return R_t , which aggregates all N agents' stochastic actions, yielding $\mathcal{O}(N)$ variance. The guidance sample $\hat{g}_{\mathcal{G},t}^i$ uses an analytically computed coefficient $\langle \mathbf{x}_t - \tilde{\mathbf{x}}_t, \mathbf{z}_t^i \rangle$, contributing $\mathcal{O}(1)$ variance independent of N . We formalize how the choice of α controls this variance trade-off in Section 5.

4.3 Implementation

Integration with MAPPO. DG-PG can be integrated into the MAPPO framework [30] by modifying the advantage estimator. After collecting trajectories and computing the standard GAE advantages $\hat{A}_{GAE}^i(t)$, they are augmented with the guidance term:

$$\hat{A}_{DG}^i(t) = (1 - \alpha) \hat{A}_{GAE}^i(t) - \alpha \cdot \langle \mathbf{x}_t - \tilde{\mathbf{x}}_t, \mathbf{z}_t^i \rangle, \quad (14)$$

The computational overhead is minimal, since the reference trajectory $\{\tilde{\mathbf{x}}_t\}$ is computed once per batch using the analytical model, and each guidance coefficient is just an inner product. The policy and critic are then updated using standard PPO objectives with the modified advantages.

Comparison with Existing Methods. Unlike counterfactual methods (e.g., COMA [6]) that require training N separate critics, or value decomposition methods (e.g., QMIX [20]) that need additional mixing networks, DG-PG requires *no architectural changes* to existing methods. The only modification is a single-line change to the advantage estimator, given in Equation 14.

5 Theoretical Guarantees

In this section, we establish the theoretical foundations of DG-PG. Our central result is that the DG-PG gradient estimator achieves *agent-independent variance* (Theorem 5.2), breaking the $\mathcal{O}(N)$ scaling barrier of standard policy gradients. We complement this with two supporting guarantees: (1) *consistency* (Theorem 5.1), ensuring that DG-PG preserves the Nash equilibria of the original game; and (2) *sample complexity* (Theorem 5.3), showing that the variance reduction yields agent-independent convergence rates under standard regularity assumptions (the complete dependency structure is given in Table 3, Appendix B).

5.1 Nash Invariance

A primary concern when augmenting the policy gradient is whether the additional guidance term shifts the Nash equilibria of the original cooperative game. We address this by showing that DG-PG preserves the stationary equilibria of the original objective.

Theorem 5.1 (Nash Invariance). *Let θ^* be a Nash equilibrium of the original cooperative game, i.e., $\nabla_{\theta^i} J(\theta^*) = \mathbf{0}$ for all $i \in \mathcal{N}$. Under Assumptions 4.1–4.2, the DG-PG gradient also vanishes at θ^* :*

$$\nabla_{\theta^i} J_\alpha(\theta^*) = \mathbf{0}, \quad \forall \alpha \in (0, 1), \forall i \in \mathcal{N}. \quad (15)$$

This invariance holds for any guidance weight $\alpha \in (0, 1)$, so α can be chosen to control variance without changing the equilibrium set of the original game.

Proof Sketch. We first show by contradiction that $\nabla_{\theta^i} \mathcal{G}(\theta^*) = \mathbf{0}$ at any Nash equilibrium θ^* (Lemma E.1). Since both $\nabla_{\theta^i} J$ and $\nabla_{\theta^i} \mathcal{G}$ vanish at θ^* , the DG-PG gradient $\nabla_{\theta^i} J_\alpha = (1 - \alpha) \cdot \mathbf{0} - \alpha \cdot \mathbf{0} = \mathbf{0}$. The detailed proof is in Appendix E. $\square$

5.2 Variance Reduction

Having established that DG-PG preserves the Nash equilibria of J (Theorem 5.1), we now formalize the variance properties of the DG-PG gradient estimator. The return R_t in the standard policy gradient aggregates all N agents' stochastic actions, so the per-agent gradient variance $\text{Var}(\hat{g}_J^i) = \mathcal{O}(N)$. In contrast, the guidance coefficient $\langle \mathbf{x}_t - \bar{\mathbf{x}}_t, \mathbf{z}_t^i \rangle$ depends only on agent-local quantities (Section 4.2), yielding guidance gradient variance $\text{Var}(\hat{g}_G^i) = \mathcal{O}(1)$ independent of N .

Since the DG-PG estimator combines $\hat{g}_J^i$ and $\hat{g}_G^i$, its variance depends on their correlation $\rho = \text{Corr}(\hat{g}_J^i, \hat{g}_G^i)$. Under Assumption 4.2, improving J reduces the deviation $\mathcal{G}$, so $\hat{g}_J^i$ and $\hat{g}_G^i$ are negatively correlated ($\rho < 0$). The following theorem characterizes the optimal variance as a function of ρ and the variances of two gradient estimators.

Theorem 5.2 (Agent-Independent Variance). *Under Assumptions 4.1–4.2, let $\sigma_J^2 \triangleq \text{Var}(\hat{g}_J^i)$, $\sigma_G^2 \triangleq \text{Var}(\hat{g}_G^i)$, and $\rho \triangleq \text{Corr}(\hat{g}_J^i, \hat{g}_G^i) \in (-1, 0)$. The minimum variance over the guidance weight α is*

$$\min_{\alpha \in (0, 1)} \text{Var}(\hat{g}_{\text{DG}}^i) = \frac{\sigma_J^2 \sigma_G^2 (1 - \rho^2)}{\sigma_J^2 + \sigma_G^2 + 2\rho \sigma_J \sigma_G}. \quad (16)$$

Since $\sigma_J^2 = \mathcal{O}(N)$ and $\sigma_G^2 = \mathcal{O}(1)$, this minimum variance is $\mathcal{O}(1)$, independent of N .

Proof Sketch. The variance of the DG-PG estimator $\sigma_{\text{DG}}^2(\alpha)$ is a convex quadratic in α with cross-term $\rho \sigma_J \sigma_G$. Minimizing in closed form yields

$$\sigma_{\text{DG}, \min}^2 = \sigma_J^2 \sigma_G^2 (1 - \rho^2) / (\sigma_J^2 + \sigma_G^2 + 2\rho \sigma_J \sigma_G).$$

Substituting $\sigma_J^2 = \mathcal{O}(N)$ and $\sigma_G^2 = \mathcal{O}(1)$, the denominator is dominated by σ_J^2 and is thus $\mathcal{O}(N)$, while the numerator is $\mathcal{O}(N)$, giving $\sigma_{\text{DG}, \min}^2 = \mathcal{O}(1)$ for all N . $\square$

The role of α . Appendix F shows that the optimal weight α^* is a function of σ_J^2 , σ_G^2 , and ρ . Since only σ_J^2 grows with N , this optimizer shifts toward the guidance signal and satisfies $\alpha^* \rightarrow 1$ in the large- N limit. At finite scales, α remains a tuning choice that balances variance reduction with the original policy-gradient signal, and we validate this choice empirically in Section 6.

5.3 Convergence Rate

The variance reduction in Theorem 5.2, combined with standard regularity assumptions, yields agent-independent sample complexity.

Theorem 5.3 (Sample Complexity). *Assume J is L -smooth and satisfies the Polyak-Łojasiewicz (PL) condition with PL constant $\mu \in (0, L]$ (see Assumption G.1 in Appendix G). Given the optimal variance from Theorem 5.2, DG-PG achieves an ϵ -optimal policy in:*

$$T = \mathcal{O} \left(\frac{L}{\mu^2 \epsilon} \log \frac{1}{\epsilon} \right) \quad (17)$$

iterations. This bound is independent of the number of agents N .

This contrasts with standard multi-agent policy gradient methods, which require $T = \mathcal{O}(N/\epsilon)$ iterations to compensate for $\mathcal{O}(N)$ variance. We evaluate the scalability in Section 6.4.

Proof Sketch. L -smoothness bounds the per-iteration descent in terms of $\|\nabla J\|^2$ and the gradient variance σ_{DG}^2 . The PL condition converts $\|\nabla J\|^2$ into the optimality gap $J^* - J(\theta_k)$. Substituting $\sigma_{\text{DG}}^2 = \mathcal{O}(1)$ from Theorem 5.2 yields the final bound. The derivation is provided in Appendix G. $\square$

Summary. Theorems 5.1–5.3 establish that DG-PG is (1) scalable: variance does not grow with N ; (2) safe: optimal solutions are preserved; and (3) efficient: sample complexity is agent-independent.

6 Experiments

We evaluate DG-PG on a heterogeneous cloud resource scheduling task with up to $N=1500$ agents, a challenging domain where queueing-theoretic priors are available and the cross-agent noise problem is particularly acute. Our experiments address three questions: (1) guidance-weight sensitivity; (2) controlled comparison with MAPPO and IPPO; and (3) scalability to large agent populations.

6.1 Setup

Environment. We evaluate DG-PG on a cloud scheduling simulator with N servers based on AWS instance types and N job dispatcher agents making placement decisions. The simulator combines heterogeneous server efficiency, heavy-tailed bimodal workloads, and non-stationary arrivals. Together, these factors create model mismatch by violating the stationarity assumptions behind the queueing-theoretic prior. The objective is to simultaneously reduce queueing delay and improve energy efficiency. Figure 1 summarizes the environment, and Appendix H provides the full specifications.

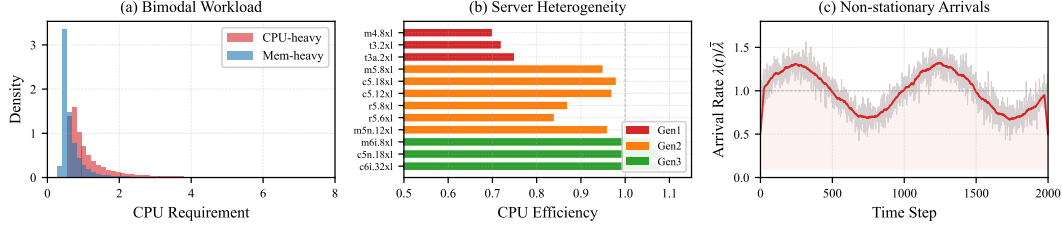


Figure 1: Environment characteristics. (a) Bimodal workloads with heavy-tailed demands. (b) Server heterogeneity across hardware generations. (c) Non-stationary arrivals with tidal fluctuation.

Baselines. We compare against (1) **MAPPO** [30], multi-agent PPO with a centralized value function, and (2) **IPPO**, independent PPO with local value functions. We additionally report (3) **Best-Fit**, a heuristic algorithm that operates with centralized dispatcher, as an upper reference, and (4) **Random** as a lower reference. Mean-Field methods [29] are excluded because they assume agent homogeneity. Training settings are detailed in Appendix I.

6.2 Guidance Weight Selection

This section evaluates how the guidance weight α affects convergence speed and final performance. We test four fixed values of α at both $N=20$ and $N=50$.

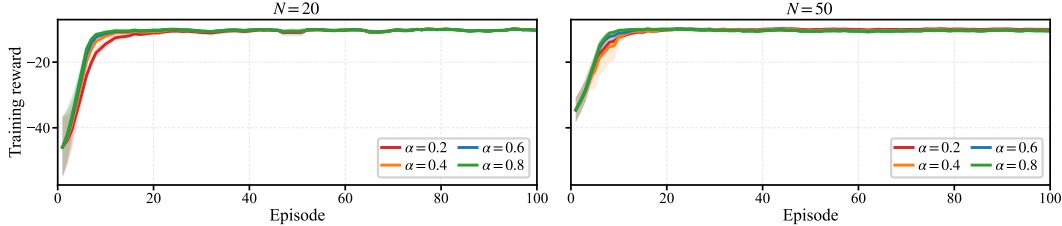


Figure 2: Training reward at $N=20$ and $N=50$ for varying guidance weight α .

Results. Figure 2 shows training curves at $N=20$ and $N=50$ for all tested α values. The results reveal a trade-off in which *larger α accelerates early convergence, while smaller α achieves marginally superior best performance*. We evaluate the best checkpoint for each α on 10 held-out test seeds. At $N=50$, $\alpha=0.2$ achieves -8.75 ± 0.21 , slightly outperforming $\alpha=0.8$ at -9.04 ± 0.12 . The same pattern appears at $N=20$. Importantly, the small spread across tested α values also shows that DG-PG is *robust to the choice of α* .

Selected Hyperparameters. Based on the observed trade-off, we use a dynamic α schedule for scales within $N \leq 50$, starting from $\alpha=0.9$ for fast early convergence and decaying to $\alpha=0.2$ for later fine-tuning. For scales with $N \geq 100$, where cross-agent noise is more severe, we fix $\alpha=0.9$ to maintain strong guidance throughout training. Specific settings are provided in Appendix I.

6.3 Controlled Comparison

All three methods are trained under matched core hyperparameters (Appendix I). To give MAPPO and IPPO the best chance to learn, they receive $2.5\times$ more training than DG-PG (500 vs. 200 episodes) and tuned entropy coefficients. We evaluate $N \in \{2, 5, 10\}$, where gradient variance is lowest.

Figure 3 shows the training curves for the controlled comparison. For quantitative comparison, we evaluate each method on the test set at its best checkpoint. At $N=2$, all methods are comparable. At $N=5$, DG-PG reaches -10.97 ± 0.38 , while MAPPO and IPPO reach -21.63 ± 0.77 and -23.04 ± 0.29 . At $N=10$, the gap widens, with DG-PG reaching -9.69 ± 0.23 compared with -29.40 ± 0.07 for MAPPO and -28.71 ± 2.81 for IPPO. These results show that the performance gap widens as N increases, consistent with the cross-agent noise bottleneck.

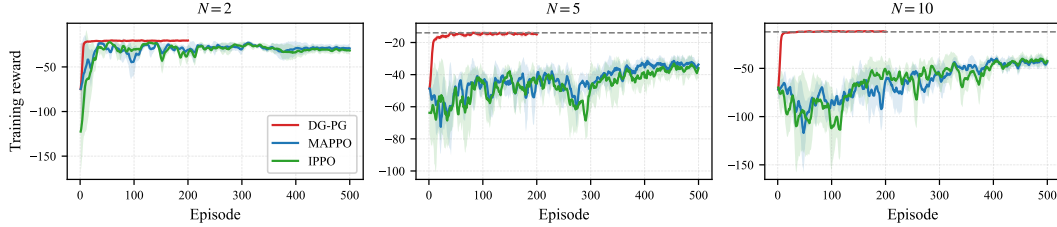


Figure 3: Training reward under the controlled comparison at $N \in \{2, 5, 10\}$.

6.4 Scalability

We next evaluate whether DG-PG scales up to $N=1500$ agents. MAPPO and IPPO are excluded as they fail to converge at these scales despite extensive tuning (Appendix J).

Scale-Invariant Convergence. Figure 4(a) shows DG-PG’s large-scale training curves from $N=50$ to $N=1500$, with smaller scales omitted for visual clarity. Across these scales, DG-PG reaches within 10% of its best observed reward in 6.8–17.7 episodes on average across training seeds, and the required number of episodes remains bounded as N increases. *This scale-invariant convergence is the empirical counterpart of Theorem 5.3, confirming that DG-PG maintains $\mathcal{O}(1)$ sample complexity with respect to N .*

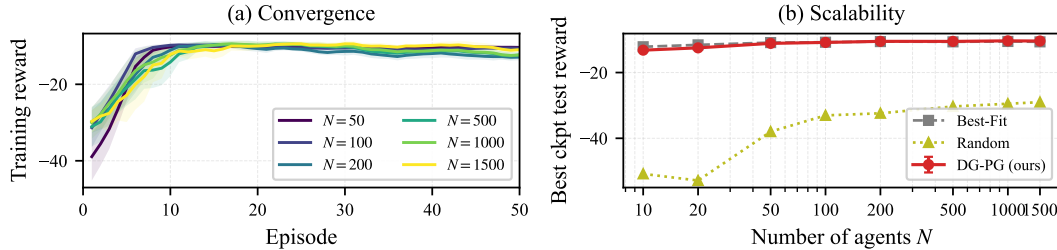


Figure 4: Scalability and convergence.

Best Checkpoint Performance. Figure 4(b) and Table 1 report performance on the same 10 held-out test seeds, with DG-PG evaluated at its best checkpoint. As introduced above, Best-Fit relies on a *centralized scheduler* to assign jobs, while DG-PG makes *decentralized decisions* through learned agent policies. Nevertheless, DG-PG matches the centralized Best-Fit scheduler at smaller scales and surpasses it from $N=200$ onward, showing that decentralized decision-making can learn coordination comparable to centralized scheduling. *This makes DG-PG suitable for distributed systems, where coordination must emerge from agent-level decisions rather than an online centralized scheduler.*

Table 1: Performance on held-out test seeds across scales. DG-PG reports mean $\pm$ std over training seeds, and Best-Fit and Random are deterministic point estimates.

	$N=10$	$N=20$	$N=50$	$N=100$	$N=200$	$N=500$	$N=1000$	$N=1500$
DG-PG (ours)	-13.19 ± 0.14	-12.49 ± 0.10	-11.13 ± 0.03	-10.84 ± 0.02	-10.51 ± 0.11	-10.52 ± 0.06	-10.36 ± 0.08	-10.41 ± 0.02
Best-Fit	-12.15	-11.58	-10.88	-10.77	-10.58	-10.66	-10.55	-10.66
Random	-50.82	-52.85	-37.98	-33.04	-32.40	-30.32	-29.49	-29.01

Computational Cost. The episode-level convergence gain also translates into practical wall-clock cost. Under *CPU-only execution (Apple M1)*, the per-episode training time ranges from 10.5 seconds at $N=10$ to 335.0 seconds at $N=1500$. However, because DG-PG reaches the convergence threshold within fewer than 18 episodes on average across all reported scales, the total wall-clock convergence time remains moderate. Specifically, it takes less than 10 minutes for $10 \leq N \leq 50$ (2.9–5.0 minutes) and less than 50 minutes for $100 \leq N \leq 1000$ (16.2–42.1 minutes). Even at the largest scale, $N=1500$, where each training episode is more than $30\times$ slower than at $N=10$, DG-PG reaches the convergence threshold in about 99 minutes. This shows that DG-PG can scale to large cooperative MARL systems on commodity CPU-only hardware, without requiring GPUs or specialized training infrastructure. Detailed per-scale timing and breakdown are provided in Appendix K.

These experiments further show that DG-PG learns a standalone decentralized policy, rather than a policy that depends on the analytical reference at execution time. The analytical model is used only to construct gradient guidance during training. Once training is complete, each agent acts through its learned policy without guidance or an online centralized scheduler. This differs from residual policy methods, where the learned component remains tied to a base controller at deployment. The held-out results above show that this standalone policy internalizes effective load-balancing behavior and can match or exceed the centralized Best-Fit scheduler.

7 Conclusion

In this work, we identified cross-agent noise as the central bottleneck in scaling cooperative MARL. DG-PG addresses this bottleneck by deriving a per-agent descent signal from how each agent’s action changes the deviation from an analytical reference state, then using this signal to guide policy-gradient updates. This guidance reduces per-agent gradient variance from $\mathcal{O}(N)$ to $\mathcal{O}(1)$ and yields agent-independent sample complexity, while preserving the Nash equilibria of the original cooperative game. The learned policy remains decentralized at execution time, with the analytical reference used only for training-time gradient guidance. On a heterogeneous cloud resource scheduling task with up to $N=1500$ agents, DG-PG shows scale-invariant convergence and maintains strong performance across all tested scales.

DG-PG shows that analytical models can be used not merely to provide heuristics or controllers, but to address cross-agent noise in cooperative MARL. It only requires a directionally aligned reference, rather than an exact optimum, an attainable target, or an executable policy. DG-PG offers a principled route to reliable distributed decision-making and efficient learning in large-scale engineered systems, such as cloud platforms, traffic networks, and power grids.

Limitations and future work. DG-PG requires an analytical reference that is directionally aligned with the cooperative objective. Domains without such structure will need learned or alternative guidance, and applying DG-PG to a new system requires specifying the system state, deriving the reference, and verifying its alignment. The current guidance weight α is selected empirically, so future work should adapt α from online estimates of gradient variance, guidance variance, and their correlation.

References

- [1] Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *Journal of Machine Learning Research*, 22(98):1–76, 2021.
- [2] Dimitri Bertsekas and Robert Gallager. *Data networks*. Athena Scientific, 2021.
- [3] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 153–167, 2017.
- [4] Christina Delimitrou and Christos Kozyrakis. Quasar: Resource-efficient and QoS-aware cluster management. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 127–144, 2014.
- [5] Samah El-Tantawy, Baher Abdulhai, and Hossam Abdelgawad. Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atssc): Methodology and large-scale application on downtown toronto. *IEEE transactions on Intelligent transportation systems*, 14(3):1140–1150, 2013.
- [6] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [7] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *8th USENIX symposium on networked systems design and implementation (NSDI 11)*, 2011.
- [8] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pages 6023–6029. IEEE, 2019.
- [9] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 795–811. Springer, 2016.
- [10] Frank P Kelly, Aman K Maulloo, and David Kim Hong Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society*, 49(3):237–252, 1998.
- [11] Jakub Grudzien Kuba, Muning Wen, Linghui Meng, Haifeng Zhang, David Mguni, Jun Wang, and Yaodong Yang. Settling the variance of multi-agent policy gradients. *Advances in Neural Information Processing Systems*, 34:13458–13470, 2021.
- [12] Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2022.
- [13] Thomas G Kurtz. *Approximation of population processes*. SIAM, 1981.
- [14] Sergey Levine and Vladlen Koltun. Guided policy search. In *International conference on machine learning*, pages 1–9. PMLR, 2013.
- [15] Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. Learning scheduling algorithms for data processing clusters. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 270–288, 2019.
- [16] Jincheng Mei, Chenjun Xiao, Csaba Szepesvari, and Dale Schuurmans. On the global convergence rates of softmax policy gradient methods. In *International Conference on Machine Learning*, pages 6820–6829. PMLR, 2020.

- [17] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, pages 278–287, 1999.
- [18] Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.
- [19] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.
- [20] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.
- [21] Charles Reiss, John Wilkes, and Joseph L Hellerstein. Google cluster-usage traces: format+schema. *Google Inc., White Paper*, 1:1–14, 2011.
- [22] Charles Reiss, Alexey Tumanov, Gregory R Ganger, Randy H Katz, and Michael A Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the third ACM symposium on cloud computing*, pages 1–13, 2012.
- [23] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.
- [24] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.
- [25] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. In *International Conference on Learning Representations*, 2021.
- [26] John Glen Wardrop. Road paper. some theoretical aspects of road traffic research. *Proceedings of the institution of civil engineers*, 1(3):325–362, 1952.
- [27] David H Wolpert and Kagan Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(02n03):265–279, 2001.
- [28] Allen J Wood, Bruce F Wollenberg, and Gerald B Sheblé. *Power generation, operation, and control*. John wiley & sons, 2013.
- [29] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In *International conference on machine learning*, pages 5571–5580. PMLR, 2018.
- [30] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624, 2022.

A Comparison with Existing Methods

Table 2 provides a systematic comparison of DG-PG with representative methods from major paradigms in cooperative MARL. We evaluate methods across key dimensions, including gradient variance scaling, sample complexity, support for heterogeneous policies, parallel updates, architectural overhead, and domain knowledge requirements. The comparison highlights DG-PG’s unique position as the only method that simultaneously achieves $\mathcal{O}(1)$ gradient variance, supports heterogeneous agent policies with parallel updates, and requires no additional learned components beyond a standard centralized critic.

Key Insights from the Comparison

1. Every existing method sacrifices at least one critical property. For large-scale cooperative MARL, we identify five desirable properties: $\mathcal{O}(1)$ variance, $\mathcal{O}(1/\epsilon)$ sample complexity, heterogeneous policies, parallel updates, and zero extra learned components. No prior method achieves all five:

- **MAPPO / IPPO:** $\mathcal{O}(N)$ variance $\Rightarrow \mathcal{O}(N/\epsilon)$ sample complexity.
- **HAPPO:** sequential updates $\Rightarrow \mathcal{O}(N)$ wall-clock cost per iteration.
- **Mean-Field:** requires agent exchangeability $\Rightarrow$ no heterogeneous policies.
- **QMIX / QPLEX:** extra mixing networks or $\mathcal{O}(N^2)$ attention layers.
- **Diff. Rewards:** requires an exact simulator, often intractable at scale.

DG-PG satisfies all five simultaneously (Theorems 5.2–5.3), with only a minimal modification to the advantage computation in standard policy gradient algorithm such as MAPPO.

2. Leveraging analytical models without sacrificing optimality. All other methods in Table 2 are purely model-free, either imposing no assumptions at all (MAPPO, IPPO, HAPPO, COMA) or encoding structural constraints into the architecture (QMIX, VDN, Mean-Field). DG-PG is unique in leveraging an *external analytical model* to reduce variance while remaining model-free in its convergence guarantees.

DG-PG’s gradient-level use of analytical models also distinguishes it from other knowledge-informed learning paradigms: Physics-Informed Neural Networks [19] embed governing equations directly into the loss, requiring exact PDEs; Residual RL [8] uses a hand-designed controller as a permanent base policy, inheriting its biases; Guided Policy Search [14] relies on trajectory optimization with known dynamics. These methods all incorporate domain knowledge into the objective or policy. In contrast, DG-PG derives the gradient of a guidance term from the analytical model to steer policy updates, leaving the original objective J unchanged. As a result, model inaccuracies affect only variance, not the Nash equilibria of the original game (Theorem 5.1). The adjustable α further allows practitioners to tune the degree of reliance based on model fidelity.

B Theoretical Assumptions and Dependencies

For reviewer convenience, Table 3 summarizes the dependency structure between our theoretical results and the assumptions they rely upon.

Discussion of Assumption Strength.

- **Assumption 4.1 (Exogeneity)** is a design choice rather than a restrictive condition. It is satisfied by construction when the reference is computed from exogenous system parameters (e.g., server capacities, total workload) and treated as a fixed target during gradient computation via the stop-gradient operator $\perp(\cdot)$.
- **Assumption 4.2 (Descent-Aligned Reference)** is the core structural requirement. It holds when the analytical model captures the dominant performance drivers of the cooperative objective. We verify this rigorously for cloud scheduling in Appendix C.3 and outline verification templates for traffic, communication, and power grid domains.

Table 2: Systematic comparison of cooperative MARL methods across key architectural and algorithmic dimensions. DG-PG is the only method that simultaneously achieves $\mathcal{O}(1)$ variance, $\mathcal{O}(1/\epsilon)$ sample complexity, full heterogeneity support, parallel updates, and zero extra learned components.

Method	Approach	Variance Scaling	Sample Complexity	Heterog. Policies	Parallel Updates	Extra Learned Components	Assumption / Requirement
<i>Policy Gradient Methods</i>							
MAPPO [30]	Centralized PG	$\mathcal{O}(N)$	$\mathcal{O}(N/\epsilon)$	✓	✓	None	None
IPPO [30]	Independent PG	$\mathcal{O}(N)$	$\mathcal{O}(N/\epsilon)$	✓	✓	None	None
HAPPO [12]	Sequential PG	$\mathcal{O}(1)^\dagger$	$\mathcal{O}(N/\epsilon)^\dagger$	✓	×	None	None
COMA [6]	Counterfactual	$\mathcal{O}(N)^\ddagger$	$\mathcal{O}(N/\epsilon)$	✓	✓	N critic heads	None
<i>Value Decomposition Methods</i>							
QMIX [20]	Mixing Network	N/A (Value)	—	✓	✓	Mixing net	Monotonicity
VDN [23]	Additive	N/A (Value)	—	✓	✓	None	Additivity
QPLEX [25]	Attention	N/A (Value)	—	✓	✓	Attn mixer, $\mathcal{O}(N^2)$	Dueling struct.
<i>Mean-Field Approximation</i>							
Mean-Field [29]	Neighbor Avg.	$\mathcal{O}(1)^\S$	$\mathcal{O}(1/\epsilon)^\S$	×	✓	None	Exchangeability
<i>Reward Modification Methods</i>							
PBRS [17]	Potential Shaping	$\mathcal{O}(N)$	$\mathcal{O}(N/\epsilon)$	✓	✓	None	Potential func.
Diff. Rewards [27]	Counterfactual	Problem-dep.*	Problem-dep.*	✓	✓	None	Exact simulator
<i>Gradient Guidance (Proposed)</i>							
DG-PG (Ours)	Analytic Guide	$\mathcal{O}(1)$	$\mathcal{O}(1/\epsilon)$	✓	✓	None	Analytical model

Notes: “Variance Scaling” = per-agent gradient variance w.r.t. N ; “Sample Complexity” = number of iterations to reach ϵ -optimality (proven bounds for PG methods); “Extra Learned Components” = trainable modules beyond standard actor-critic; “—” = not directly comparable (value-based methods lack explicit PG complexity bounds).

[†]HAPPO reduces per-agent variance via sequential updates with trust-region constraints, but sacrifices parallelism. Consequently, wall-clock time per iteration scales as $\mathcal{O}(N)$, yielding effective sample complexity $\mathcal{O}(N/\epsilon)$ in wall-clock time.

[‡]COMA requires N separate critic heads to compute counterfactual baselines. Reduces own-action variance σ_{self}^2 , but does not suppress cross-agent noise $(N - 1)\sigma_{\text{others}}^2$.

*Difference Rewards reduce variance by subtracting a counterfactual baseline $r(s, \mathbf{a}) - r(s, \mathbf{a}^{-i})$, but the degree of reduction depends on the reward structure (separability). For non-separable rewards, cross-agent interaction terms persist. Computing the counterfactual also requires an exact simulator, which is often intractable at scale.

[§]Mean-Field methods assume homogeneous agents (identical and exchangeable). Results hold only under this approximation, and convergence is to the mean-field equilibrium, which may differ from the Nash equilibrium of the original N -agent game.

Table 3: Dependency of theoretical results on assumptions. “✓” indicates that the result requires the corresponding assumption. All assumptions are stated in the main text (Section 4.1) and Appendix G.

Result	Exogeneity (Asm. 4.1)	Alignment (Asm. 4.2)	Smoothness & PL (Asm. G.1)	Bounded Variance	Role of Assumptions
Thm. 5.1 (Nash Invariance)	✓	✓			Exogeneity ensures reference does not shift; Alignment ensures guidance gradient vanishes at Nash equilibria.
Thm. 5.2 (Variance Bound)	✓	✓			Exogeneity makes reference deterministic; Alignment ensures negative correlation $\rho < 0$.
Thm. 5.3 (Sample Complexity)	✓	✓	✓	✓	Smoothness enables the per-iteration bound; PL condition yields linear convergence rate; $\mathcal{O}(1)$ variance from Thm. 5.2.
Prop. 1 in App. C.3 (Cloud Verification)					Standalone verification that Alignment holds for cloud scheduling via Cauchy-Schwarz on the load imbalance functional.

- **Assumption G.1 (Smoothness & PL)** is standard in the stochastic optimization literature and is required only for the convergence rate result (Theorem 5.3). The variance reduction (Theorem 5.2) and Nash invariance (Theorem 5.1) hold without these regularity conditions.

C Case Study: Verification of Structural Assumptions

In this appendix, we provide a detailed verification that the structural Assumptions 4.1 and 4.2 hold for the cloud resource scheduling task used in our experiments. This serves as a representative case study demonstrating how domain-specific priors can be systematically validated within the DG-PG framework.

Applicability to Other Domains. While we focus on cloud scheduling, similar verification procedures apply to other control problems where analytical priors are available:

- *Communication networks:* Network utility maximization theory provides proportional fairness as a natural reference allocation [10].
- *Power grids:* DC optimal power flow and economic dispatch yield reference generation schedules [28].
- *Traffic systems:* Wardrop equilibrium and system-optimal routing define reference flow distributions [26].

In each case, the key requirement is that the prior defines a descent direction aligned with the cooperative objective, a property we formally verify below for the cloud scheduling domain using fluid-limit analysis from queueing theory [13].

C.1 System Model and Objective Approximation

System Setup. We study a cooperative multi-agent system for task scheduling in a heterogeneous cluster of N servers. Tasks are characterized by multi-dimensional resource requirements (e.g., CPU, memory), and servers possess heterogeneous capacities across these dimensions. Let K denote the number of resource types. We model the system state using the full resource utilization vector.

Specifically, let $\mathbf{x}_t^i = (x_t^{i,1}, \dots, x_t^{i,K})^\top \in \mathbb{R}_{\geq 0}^K$ denote the *resource utilization vector* on server i at time t , where $x_t^{i,k}$ is the load on the k -th resource dimension. The server capacities are given by $\boldsymbol{\mu}^i = (\mu^{i,1}, \dots, \mu^{i,K})^\top$, where $\mu^{i,k}$ is the maximum processing rate of server i on resource dimension k . This generalization captures the coupling between resources, avoiding the information loss associated with scalar bottleneck abstractions.

The total workload pressure in the system on dimension k , denoted as C^k , is conserved under task assignment decisions:

$$\sum_{i=1}^N x_t^{i,k} = C^k, \quad x_t^{i,k} \geq 0, \quad \forall t, \forall k.$$

Surrogate Objective via Queueing Theory. To construct the reference state $\tilde{\mathbf{X}}$, we need an analytically tractable objective whose minimizer serves as the guidance target. The cooperative MARL objective J itself is intractable because it aggregates queueing delay, energy efficiency, and fairness through complex, non-separable interactions. However, queueing theory reveals that improving each of these criteria reduces to a common structural condition:

- **Queueing delay** is a convex function of server utilization, scaling as $1/(\mu - x)$ near capacity [2]. By Jensen's inequality, total delay is minimized when utilization is equalized across servers.
- **Energy cost** in heterogeneous clusters with varying $\mu^{i,k}$ is minimized by capacity-proportional allocation, which avoids overloading low-efficiency machines.
- **Fairness** improves as utilization variance decreases, reducing service-level violations on congested servers.

All three criteria improve when server utilization is balanced proportionally to capacity. We formalize this shared structure as the *load imbalance functional*:

$$\mathcal{J}_{\text{load}}(\mathbf{X}) \triangleq \sum_{k=1}^K \sum_{i=1}^N \frac{1}{\mu^{i,k}} (x^{i,k})^2, \quad (18)$$

where $\mathbf{X} = \{\mathbf{x}^1, \dots, \mathbf{x}^N\}$ is the global system state. The gradient $\frac{\partial \mathcal{J}_{\text{load}}}{\partial x^{i,k}} = \frac{2x^{i,k}}{\mu^{i,k}}$ is proportional to the utilization rate of resource k on server i , so minimizing $\mathcal{J}_{\text{load}}$ drives the system toward capacity-proportional allocation. Crucially, $\mathcal{J}_{\text{load}}$ is convex and separable across resource dimensions, admitting a closed-form minimizer. This tractability also enables rigorous verification of Assumption 4.2, which would be intractable for the composite objective J directly.

Optimal Reference via Load Balancing. To construct the reference state $\tilde{\mathbf{X}}$, we solve the constrained optimization problem of minimizing $\mathcal{J}_{\text{load}}$ under workload conservation. For each resource dimension k , this reduces to:

$$\min_{\mathbf{X}} \sum_{i=1}^N \frac{1}{\mu^{i,k}} (x^{i,k})^2 \quad \text{subject to} \quad \sum_{i=1}^N x^{i,k} = C^k.$$

This is a classical convex optimization problem whose solution is well-established in queueing theory and resource allocation [2, 7]. Constructing the Lagrangian for dimension k :

$$\mathcal{L}_{\text{Lag}}^k(\mathbf{x}^{\cdot,k}, \lambda_k) = \sum_{i=1}^N \frac{1}{\mu^{i,k}} (x^{i,k})^2 - \lambda_k \left(\sum_{i=1}^N x^{i,k} - C^k \right),$$

and taking the first-order optimality condition:

$$\frac{\partial \mathcal{L}_{\text{Lag}}^k}{\partial x^{i,k}} = \frac{2x^{i,k}}{\mu^{i,k}} - \lambda_k = 0 \quad \implies \quad x^{i,k} = \frac{\lambda_k \mu^{i,k}}{2}.$$

This shows that at the optimum, load is allocated proportionally to capacity: $x^{i,k} \propto \mu^{i,k}$. Applying the workload constraint $\sum_i x^{i,k} = C^k$ yields:

$$x^{*,i,k} = \frac{\mu^{i,k}}{\sum_j \mu^{j,k}} \cdot C^k.$$

Equivalently, the optimal state satisfies the *utilization balancing condition*, i.e., the utilization rate $x^{i,k}/\mu^{i,k}$ is equalized across all servers for each resource type k :

$$\frac{x^{1,k}}{\mu^{1,k}} = \frac{x^{2,k}}{\mu^{2,k}} = \dots = \frac{x^{N,k}}{\mu^{N,k}}, \quad \forall k \in \{1, \dots, K\}.$$

This principle is fundamental to heterogeneous resource allocation and aligns with dominant resource fairness (DRF) in multi-dimensional scheduling systems.

Theory-Guided Reference State. Based on the optimal solution derived above, we define the *capacity-weighted load-balancing equilibrium* as our theory-guided reference:

$$\tilde{x}_t^{i,k} = \frac{\mu^{i,k}}{\sum_{j=1}^N \mu^{j,k}} \cdot C^k, \quad \forall i, k. \quad (19)$$

By construction, $\tilde{\mathbf{X}}_t = \{\tilde{x}_t^1, \dots, \tilde{x}_t^N\}$ is the unique global minimizer of $\mathcal{J}_{\text{load}}(\mathbf{X})$ under the workload constraints. Note that $\tilde{\mathbf{X}}_t$ is derived from a continuous (fluid) optimization and may not be exactly reachable under discrete task assignments; however, this does not affect the verification below. We now verify that this reference satisfies Assumptions 4.1 and 4.2. Specifically, in Appendix C.3, we prove that $(\tilde{\mathbf{X}} - \mathbf{X})$ is aligned with the descent direction of $\mathcal{J}_{\text{load}}$, guaranteeing that moving toward the reference reduces load imbalance.

C.2 Verification of Assumption 4.1 (Exogeneity)

The reference state $\tilde{\mathbf{X}}_t$ is computed deterministically from the current aggregate system workload C^k but is treated as a fixed constant during the policy gradient computation step. Specifically, for each server i and resource k , we define:

$$\tilde{x}_t^{i,k} := \perp \left(\frac{\mu^{i,k}}{\sum_{j=1}^N \mu^{j,k}} \cdot C^k \right).$$

Here, $\perp$ denotes the stop-gradient operator. Since $\tilde{\mathbf{X}}_t$ depends only on the current state information (total load) and not on the immediate action a_t selected by the policy π_θ , we have $\nabla_\theta \tilde{\mathbf{X}}_t = \mathbf{0}$, ensuring that no gradient flows back through the reference target. This satisfies Assumption 4.1.

C.3 Verification of Assumption 4.2 (Directional Alignment)

Assumption 4.2 requires that the direction $(\tilde{\mathbf{X}} - \mathbf{X})$ is a descent direction for the cooperative objective J . Since $\mathcal{J}_{\text{load}}$ is a surrogate for J justified by queueing theory (Section C), it suffices to show that $(\tilde{\mathbf{X}} - \mathbf{X})$ is a descent direction for $\mathcal{J}_{\text{load}}$.

Proposition C.1 (Alignment with Load Imbalance Reduction). *For any feasible state $\mathbf{X} \neq \tilde{\mathbf{X}}$ satisfying the workload constraints, the direction $(\tilde{\mathbf{X}} - \mathbf{X})$ is aligned with the descent direction of the load imbalance functional:*

$$\langle \nabla_{\mathbf{X}} \mathcal{J}_{\text{load}}(\mathbf{X}), \tilde{\mathbf{X}} - \mathbf{X} \rangle < 0.$$

That is, moving towards $\tilde{\mathbf{X}}$ reduces load imbalance.

Proof. Using the definition of the load imbalance functional in Equation 18, we compute the inner product between its gradient and the guidance vector. Since $\mathcal{J}_{\text{load}}$ is separable across resource

dimensions, the total inner product decomposes as a sum over resource types k :

$$\begin{aligned}\langle \nabla_{\mathbf{X}} \mathcal{J}_{\text{load}}, \tilde{\mathbf{X}} - \mathbf{X} \rangle &= \sum_{k=1}^K \sum_{i=1}^N \frac{\partial \mathcal{J}_{\text{load}}}{\partial x^{i,k}} (\tilde{x}^{i,k} - x^{i,k}) \\ &= \sum_{k=1}^K \underbrace{\sum_{i=1}^N \frac{2x^{i,k}}{\mu^{i,k}} (\tilde{x}^{i,k} - x^{i,k})}_{I_k}.\end{aligned}$$

To establish that the total inner product is negative, it suffices to show that each component $I_k \leq 0$. We analyze a fixed resource dimension k . First, observe that by the equilibrium definition in Equation 19, the reference utilization rate is constant across all servers:

$$\frac{\tilde{x}^{i,k}}{\mu^{i,k}} = \frac{C^k}{\sum_{j=1}^N \mu^{j,k}}.$$

Substituting this constant ratio into the expression for I_k :

$$\begin{aligned}I_k &= 2 \sum_{i=1}^N x^{i,k} \left(\frac{\tilde{x}^{i,k}}{\mu_i^k} - \frac{x^{i,k}}{\mu_i^k} \right) \\ &= 2 \left(\frac{C^k}{\sum_{j=1}^N \mu_j^k} \right) \sum_{i=1}^N x^{i,k} - 2 \sum_{i=1}^N \frac{(x^{i,k})^2}{\mu^{i,k}} \\ &= \frac{2(C^k)^2}{\sum_{j=1}^N \mu^{j,k}} - 2 \sum_{i=1}^N \frac{(x^{i,k})^2}{\mu^{i,k}}.\end{aligned}$$

To determine the sign of I_k , we apply the Cauchy-Schwarz inequality to vectors $\mathbf{u} = (\frac{x^{i,k}}{\sqrt{\mu^{i,k}}})_i$ and $\mathbf{v} = (\sqrt{\mu^{i,k}})_i$:

$$\left(\sum_{i=1}^N u_i v_i \right)^2 \leq \left(\sum_{i=1}^N u_i^2 \right) \left(\sum_{i=1}^N v_i^2 \right).$$

Substituting the definitions of u_i and v_i :

$$\left(\sum_{i=1}^N x^{i,k} \right)^2 \leq \left(\sum_{i=1}^N \frac{(x^{i,k})^2}{\mu^{i,k}} \right) \left(\sum_{i=1}^N \mu^{i,k} \right).$$

Rearranging terms and using $\sum x^{i,k} = C^k$:

$$\sum_{i=1}^N \frac{(x^{i,k})^2}{\mu^{i,k}} \geq \frac{(C^k)^2}{\sum_{j=1}^N \mu^{j,k}}.$$

Comparing this with the expression for I_k , we see that the second term (actual variance) is always greater than or equal to the first term (minimal variance). Thus, $I_k \leq 0$ for all k , with strict inequality if the system is unbalanced. Summing over all dimensions implies:

$$\langle \nabla_{\mathbf{X}} \mathcal{J}_{\text{load}}(\mathbf{X}), \tilde{\mathbf{X}} - \mathbf{X} \rangle = \sum_{k=1}^K I_k < 0.$$

This confirms that the direction toward the reference $\tilde{\mathbf{X}}$ is aligned with the descent direction of the load imbalance functional, thereby verifying Assumption 4.2. $\square$

Generalization to Other Domains. The verification procedure demonstrated above serves as a *template* for applying DG-PG to other systems. As outlined in the main text, domains such as communication networks (proportional fairness), power grids (economic dispatch), and traffic systems (Wardrop equilibrium) all admit analytical priors with similar structural properties. In each case, the

key steps are: (1) identify a domain-specific performance metric with known convexity properties; (2) derive the reference state as its constrained optimum; (3) verify that the reference defines a descent direction using tools from convex analysis or variational inequalities. This case study establishes the feasibility and rigor of such verification, providing a blueprint for extending DG-PG beyond cloud scheduling.

Robustness to Approximate Satisfaction. In practice, the structural assumptions may not hold exactly. For instance, the fluid-limit approximation underlying the reference state assumes Poisson arrivals and homogeneous service times, which are often violated in real systems due to heavy-tailed workloads and temporal fluctuations. However, DG-PG remains effective as long as the assumptions hold *approximately*, that is, the reference captures the dominant performance drivers even if the analytical model is idealized. The adjustable guidance weight α provides a natural mechanism for handling model inaccuracies. When the prior is less accurate, reducing α diminishes its influence while preserving the variance reduction benefits. Our experiments (Section 6.2) demonstrate that DG-PG is robust across a range of α values, confirming that directional correctness, rather than exact model fidelity, is sufficient for effective guidance.

D Derivation of Local Influence Vector

In this appendix, we derive the local influence vector $\mathbf{z}_t^i = \frac{\partial \mathbf{x}_t}{\partial a_t^i}$ for the cloud resource scheduling example. Here $\mathbf{z}_t^i$ captures the marginal effect of agent i 's action on the system state, i.e., how choosing a particular server changes the global utilization vector. This derivation generalizes to other domains with similar additive structure.

We use the notation established in Appendix C: N servers, K resource dimensions, global utilization vector $\mathbf{x}_t \in \mathbb{R}^{NK}$, and per-server utilization $\mathbf{x}_t^j \in \mathbb{R}^K$. Each agent i holds a task with resource requirement vector $\mathbf{w}^i = (w^{i,1}, \dots, w^{i,K})^\top \in \mathbb{R}^K$, where $w^{i,k}$ is the demand on resource dimension k (e.g., CPU cores, memory). Each agent i selects a target server $a_t^i \in \{1, \dots, N\}$, and the next-step utilization on server j is determined by the joint actions of all agents:

$$\mathbf{x}_{t+1}^j = \mathbf{x}_t^j + \sum_{l=1}^N \mathbf{w}^l \cdot \mathbb{I}(a_t^l = j).$$

To compute the marginal effect of agent i 's action, we differentiate with respect to a_t^i . Since the contributions of all other agents $l \neq i$ are held fixed, their terms vanish, leaving only agent i 's contribution:

$$\mathbf{z}_t^i \triangleq \frac{\partial \mathbf{x}_{t+1}}{\partial a_t^i} = (\mathbf{0}, \dots, \underbrace{\mathbf{w}^i}_{\text{server } j}, \dots, \mathbf{0})^\top.$$

That is, assigning task i to server j increases server j 's load by exactly $\mathbf{w}^i$ across all resource types, while leaving all other servers unaffected.

Projected coefficient. In the DG-PG update (Equations 8–10), each agent's guidance signal is scaled by the inner product $\langle \mathbf{x}_t - \tilde{\mathbf{x}}_t, \mathbf{z}_t^i \rangle$, which measures how much agent i 's action contributes to the change in deviation from the reference. Substituting the influence vector $\mathbf{z}_t^i$ yields:

$$\langle \mathbf{x}_t - \tilde{\mathbf{x}}_t, \mathbf{z}_t^i \rangle = \sum_{k=1}^K w^{i,k} \cdot (x_t^{j,k} - \tilde{x}_t^{j,k}).$$

Each term measures how overloaded server j is on resource k , weighted by the task's own demand $w^{i,k}$, so resource types that the task consumes most heavily dominate the signal. Importantly, this coefficient provides a *direction* for policy improvement, not a hard constraint that forces convergence to $\tilde{\mathbf{X}}$. The actual system trajectory is determined by the learned policy interacting with the environment.

D.1 General Applicability

The key requirement is that the influence vector $\mathbf{z}_t^i$ captures the marginal effect of an action on the system state. This structure arises naturally in many multi-agent systems:

- **Traffic routing:** x_t represents link flows and actions are path selections, so z_t^i indicates which links are used.
- **Inventory management:** x_t represents stock levels and actions are replenishment orders, where z_t^i adds to specific stock dimensions.
- **Communication networks:** x_t represents channel interference levels and actions are power allocations, and z_t^i captures each agent's marginal contribution to interference.

In each case, z_t^i enables the projection-based variance reduction by isolating the relevant components of the global state deviation.

E Detailed Proof of Nash Invariance (Theorem 5.1)

In this appendix, we provide the detailed proof for Theorem 5.1, establishing that our DG-PG framework preserves the Nash equilibria of the original cooperative game.

Theorem 5.1 (Nash Invariance). *Let θ^* be a Nash equilibrium of the original cooperative game, i.e., $\nabla_{\theta^i} J(\theta^*) = \mathbf{0}$ for all $i \in \mathcal{N}$. Under Assumptions 4.1–4.2, the DG-PG gradient also vanishes at θ^* , i.e., $\nabla_{\theta^i} J_\alpha(\theta^*) = \mathbf{0}$ for any $\alpha \in (0, 1)$ and all $i \in \mathcal{N}$.*

To prove this theorem, we first establish a key lemma stating that the gradient of the guidance term vanishes at any Nash equilibrium of the original game.

E.1 Lemma: Guidance Consistency

Lemma E.1. *Under Assumptions 4.1 and 4.2, if $\nabla_{\theta^i} J(\theta^*) = \mathbf{0}$ for all $i \in \mathcal{N}$, then $\nabla_{\theta^i} \mathcal{G}(\theta^*) = \mathbf{0}$ for all $i \in \mathcal{N}$.*

Proof. We proceed by contradiction. Suppose $\nabla_{\theta^i} J(\theta^*) = \mathbf{0}$ for all $i \in \mathcal{N}$ (the Nash condition), but $\nabla_{\theta^i} \mathcal{G}(\theta^*) \neq \mathbf{0}$ for some agent i . Recall that $\mathcal{G}(\theta) = \mathbb{E}_{s \sim \nu^\pi} [d(x^\pi|_s, \tilde{x}|_s)]$ is the expected deviation under the state visitation distribution ν^π , where $x^\pi|_s$ is the system state jointly determined by agents' actions under policy π at MDP state s , and $\tilde{x}|_s$ is the corresponding reference. We write $x^* \triangleq x^{\pi^*}|_s$ and $\tilde{x} \triangleq \tilde{x}|_s$ for a given MDP state s , where $\pi^* \triangleq \pi_{\theta^*}$.

Step 1. If $x^* = \tilde{x}$ for all states s visited under ν^{π^*} , then $d(x^*, \tilde{x}) = 0$ everywhere and $\mathcal{G}(\theta^*) = 0$. This is the global minimum of $\mathcal{G} \geq 0$, so the first-order condition gives $\nabla_{\theta^i} \mathcal{G}(\theta^*) = \mathbf{0}$ for all i , contradicting our assumption. Therefore there exists a visited state s at which $x^* \neq \tilde{x}$.

Step 2. By Assumption 4.2, at any state s where $x^* \neq \tilde{x}$:

$$\langle \nabla_x J(x^*), \tilde{x} - x^* \rangle > 0.$$

Since $\tilde{x} - x^* \neq \mathbf{0}$, the state-space gradient $\nabla_x J(x^*)$ must be non-zero.

Step 3. Under sufficient policy expressiveness (see Remark E.2), a non-zero state-space gradient $\nabla_x J(x^*) \neq \mathbf{0}$ implies $\nabla_{\theta^i} J(\theta^*) \neq \mathbf{0}$ for some agent i . This contradicts the Nash condition.

Therefore $\nabla_{\theta^i} \mathcal{G}(\theta^*) = \mathbf{0}$ for all $i \in \mathcal{N}$. $\square$

Remark E.2 (Sufficient Expressiveness). The proof requires the Jacobian $\nabla_{\theta^i} x$ to have full row rank for at least one agent i , ensuring that a non-zero state-space gradient $\nabla_x J$ lifts to a non-zero parameter-space gradient $\nabla_{\theta^i} J$. This is a standard condition for overparameterized neural network policies ($p \gg m$) and holds generically in the absence of degenerate symmetries.

E.2 Proof of Theorem 5.1

Proof. At a Nash equilibrium θ^* , we have $\nabla_{\theta^i} J(\theta^*) = \mathbf{0}$ for all $i \in \mathcal{N}$. By Lemma E.1, $\nabla_{\theta^i} \mathcal{G}(\theta^*) = \mathbf{0}$ for all $i \in \mathcal{N}$. Therefore:

$$\nabla_{\theta^i} J_\alpha(\theta^*) = (1 - \alpha) \nabla_{\theta^i} J(\theta^*) - \alpha \nabla_{\theta^i} \mathcal{G}(\theta^*) = \mathbf{0},$$

confirming that the augmentation does not alter the Nash equilibria of the original game. $\square$

F Variance Decomposition and Reduction Analysis (Theorem 5.2)

In this appendix, we prove that the DG-PG estimator achieves agent-independent variance scaling. As established in Section 5.2, the standard policy gradient estimator has variance $\sigma_J^2 = \mathcal{O}(N)$ due to cross-agent exploration noise [11], while the gradient estimator for the guidance term has variance $\sigma_G^2 = \mathcal{O}(1)$ since its coefficient depends only on deterministic quantities given a realized trajectory.

Theorem 5.2 (Agent-Independent Variance). *Under Assumptions 4.1–4.2, let $\sigma_J^2 \triangleq \text{Var}(\hat{g}_J^i)$, $\sigma_G^2 \triangleq \text{Var}(\hat{g}_G^i)$, and $\rho \triangleq \text{Corr}(\hat{g}_J^i, \hat{g}_G^i) \in (-1, 0)$. The minimum variance over the guidance weight α is*

$$\min_{\alpha \in (0,1)} \text{Var}(\hat{g}_{\text{DG}}^i) = \frac{\sigma_J^2 \sigma_G^2 (1 - \rho^2)}{\sigma_J^2 + \sigma_G^2 + 2\rho \sigma_J \sigma_G}.$$

Since $\sigma_J^2 = \mathcal{O}(N)$ and $\sigma_G^2 = \mathcal{O}(1)$, this minimum variance is $\mathcal{O}(1)$, independent of N .

F.1 Proof of Theorem 5.2

Proof. For the DG-PG estimator $\hat{g}_{\text{DG}}^i = (1 - \alpha)\hat{g}_J^i - \alpha\hat{g}_G^i$, the variance is:

$$\sigma_{\text{DG}}^2(\alpha) = (1 - \alpha)^2 \sigma_J^2 + \alpha^2 \sigma_G^2 - 2\alpha(1 - \alpha)\rho \sigma_J \sigma_G, \quad (20)$$

where $\rho = \text{Corr}(\hat{g}_J^i, \hat{g}_G^i) \in (-1, 0)$ by Assumption 4.2.

The estimator variance $\sigma_{\text{DG}}^2(\alpha)$ is a convex quadratic in α . To minimize it, we set $\frac{d}{d\alpha} \sigma_{\text{DG}}^2 = 0$:

$$-2(1 - \alpha)\sigma_J^2 + 2\alpha\sigma_G^2 - 2(1 - 2\alpha)\rho \sigma_J \sigma_G = 0 \implies \alpha^* = \frac{\sigma_J^2 + \rho \sigma_J \sigma_G}{\sigma_J^2 + \sigma_G^2 + 2\rho \sigma_J \sigma_G}. \quad (21)$$

Substituting α^* back into $\sigma_{\text{DG}}^2(\alpha)$, the minimum variance is:

$$\sigma_{\text{DG},\min}^2 = \frac{\sigma_J^2 \sigma_G^2 (1 - \rho^2)}{\sigma_J^2 + \sigma_G^2 + 2\rho \sigma_J \sigma_G}. \quad (22)$$

As established in Section 5.2, $\sigma_J^2 = \mathcal{O}(N)$ because the return aggregates all N agents' stochastic actions, while $\sigma_G^2 = \mathcal{O}(1)$ because the guidance coefficient is computed from deterministic quantities. Dividing numerator and denominator by σ_J^2 :

$$\sigma_{\text{DG},\min}^2 = \frac{\sigma_G^2 (1 - \rho^2)}{1 + \frac{\sigma_G^2}{\sigma_J^2} + 2\rho \frac{\sigma_G}{\sigma_J}} = \frac{\sigma_G^2 (1 - \rho^2)}{1 + \mathcal{O}(N^{-1})}.$$

The numerator $\sigma_G^2 (1 - \rho^2) = \mathcal{O}(1)$ since $\sigma_G^2 = \mathcal{O}(1)$ and $\rho \in (-1, 0)$ is a constant. The denominator $1 + \mathcal{O}(N^{-1})$ is bounded away from zero for all $N \geq 1$. Therefore:

$$\sigma_{\text{DG},\min}^2 = \mathcal{O}(1), \quad \text{for all } N \geq 1. \quad (23)$$

□

Remark F.1. The optimal mixing coefficient $\alpha^* \rightarrow 1$ as $N \rightarrow \infty$, indicating that large-scale systems should rely primarily on the guidance signal to suppress cross-agent variance. In practice, we adopt a dynamic α schedule, starting with a large α for fast early convergence and decaying to a smaller value to allow the RL signal to fine-tune beyond the reference (see Section 6).

G Convergence Rate and Sample Complexity (Theorem 5.3)

We establish the sample complexity of DG-PG under standard regularity conditions.

Theorem 5.3 (Sample Complexity). *Under the smoothness and PL conditions (Assumption G.1 below) and with $\sigma_{\text{DG}}^2 = \mathcal{O}(1)$ from Theorem 5.2, DG-PG achieves an ϵ -optimal policy in $T = \mathcal{O}\left(\frac{L}{\mu^2 \epsilon} \log \frac{1}{\epsilon}\right)$ iterations, independent of N .*

Assumption G.1 (Smoothness and PL Condition). The cooperative objective $J(\boldsymbol{\theta})$ is L -smooth and satisfies the Polyak-Łojasiewicz (PL) condition with PL constant $\mu \in (0, L]$:

$$\frac{1}{2} \|\nabla J(\boldsymbol{\theta})\|^2 \geq \mu(J^* - J(\boldsymbol{\theta})),$$

where $J^* \triangleq \sup_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ denotes the optimal value of the cooperative objective.

Together, these conditions guarantee that stochastic gradient ascent achieves linear convergence of the optimality gap. These are standard conditions in stochastic optimization [9], needed only to convert the $\mathcal{O}(1)$ variance bound (Theorem 5.2) into an explicit iteration complexity. They are commonly satisfied for smooth policy parameterizations such as softmax policies [16, 1].

G.1 Proof of Theorem 5.3

Proof. Our proof first characterizes, through L -smoothness and the update rule, how the variance and bias of $\hat{g}_{\text{DG}}$ affect the per-iteration change in J . We then apply the PL condition to connect the gradient norm $\|\nabla J\|^2$ to the optimality gap $J^* - J(\boldsymbol{\theta}_k)$, thereby obtaining the N -independent iteration complexity. The following analysis holds for a *fixed* mixing parameter $\alpha \in (0, 1)$. When α is varied dynamically in practice, the convergence guarantee applies to each phase as a snapshot analysis.

Step 1: Smoothness bound. Let $\bar{g}_{\text{DG}}(\boldsymbol{\theta}_k) \triangleq \mathbb{E}_k[\hat{g}_{\text{DG}}(\boldsymbol{\theta}_k)]$ denote the expected augmented gradient. Consider the stochastic gradient ascent update

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta \hat{g}_{\text{DG}}(\boldsymbol{\theta}_k) \quad (24)$$

By L -smoothness of J :

$$J(\boldsymbol{\theta}_{k+1}) \geq J(\boldsymbol{\theta}_k) + \langle \nabla J(\boldsymbol{\theta}_k), \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k \rangle - \frac{L}{2} \|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2.$$

Substituting $\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k = \eta \hat{g}_{\text{DG}}(\boldsymbol{\theta}_k)$ and taking the conditional expectation $\mathbb{E}_k[\cdot] \triangleq \mathbb{E}[\cdot | \boldsymbol{\theta}_k]$:

$$\mathbb{E}_k[J(\boldsymbol{\theta}_{k+1})] \geq J(\boldsymbol{\theta}_k) + \eta \langle \nabla J(\boldsymbol{\theta}_k), \bar{g}_{\text{DG}}(\boldsymbol{\theta}_k) \rangle - \frac{L\eta^2}{2} \mathbb{E}_k[\|\hat{g}_{\text{DG}}(\boldsymbol{\theta}_k)\|^2]. \quad (25)$$

Step 2: Per-iteration descent bound. The descent inequality Equation 25 shows that the per-iteration ascent on J is governed by two quantities: the inner product term $\langle \nabla J, \bar{g}_{\text{DG}} \rangle$, which captures the alignment between the true gradient and the augmented gradient, and the second moment term $\mathbb{E}_k[\|\hat{g}_{\text{DG}}\|^2]$, which captures the estimator's variance and bias. We bound both independently of N .

For the inner product term $\langle \nabla J(\boldsymbol{\theta}_k), \bar{g}_{\text{DG}}(\boldsymbol{\theta}_k) \rangle$, by definition of the expected augmented gradient:

$$\bar{g}_{\text{DG}}(\boldsymbol{\theta}_k) = (1 - \alpha) \nabla J(\boldsymbol{\theta}_k) - \alpha \nabla \mathcal{G}(\boldsymbol{\theta}_k).$$

Therefore:

$$\langle \nabla J(\boldsymbol{\theta}_k), \bar{g}_{\text{DG}}(\boldsymbol{\theta}_k) \rangle = (1 - \alpha) \|\nabla J(\boldsymbol{\theta}_k)\|^2 - \alpha \langle \nabla J(\boldsymbol{\theta}_k), \nabla \mathcal{G}(\boldsymbol{\theta}_k) \rangle.$$

Since improving J reduces the deviation $\mathcal{G}$ (Assumption 4.2), the true gradients satisfy $\langle \nabla J(\boldsymbol{\theta}_k), \nabla \mathcal{G}(\boldsymbol{\theta}_k) \rangle \leq 0$, so $-\alpha \langle \nabla J, \nabla \mathcal{G} \rangle \geq 0$. Dropping this non-negative term yields the lower bound for the inner product term in Equation 25:

$$\langle \nabla J(\boldsymbol{\theta}_k), \bar{g}_{\text{DG}}(\boldsymbol{\theta}_k) \rangle \geq (1 - \alpha) \|\nabla J(\boldsymbol{\theta}_k)\|^2. \quad (26)$$

For the second moment $\mathbb{E}_k[\|\hat{g}_{\text{DG}}(\boldsymbol{\theta}_k)\|^2]$, by the bias-variance decomposition:

$$\mathbb{E}_k[\|\hat{g}_{\text{DG}}(\boldsymbol{\theta}_k)\|^2] = \|\bar{g}_{\text{DG}}(\boldsymbol{\theta}_k)\|^2 + \sigma_{\text{DG}}^2. \quad (27)$$

The variance $\sigma_{\text{DG}}^2 = \mathcal{O}(1)$ by Theorem 5.2. Since $\bar{g}_{\text{DG}} = (1 - \alpha) \nabla J - \alpha \nabla \mathcal{G}$, applying the triangle inequality:

$$\|\bar{g}_{\text{DG}}(\boldsymbol{\theta}_k)\|^2 \leq 2(1 - \alpha)^2 \|\nabla J(\boldsymbol{\theta}_k)\|^2 + 2\alpha^2 \|\nabla \mathcal{G}(\boldsymbol{\theta}_k)\|^2. \quad (28)$$

For the $\|\nabla \mathcal{G}(\boldsymbol{\theta}_k)\|$ term, by Equation 9 the gradient of the guidance term has the form $\nabla_{\theta^i} \mathcal{G} = \mathbb{E}[\langle \mathbf{x}_t - \tilde{\mathbf{x}}_t, \mathbf{z}_t^i \rangle \nabla_{\theta^i} \log \pi^i]$. Each factor is bounded:

- $\|\mathbf{x}_t - \tilde{\mathbf{x}}_t\|$ is bounded by compactness of the system state space.
- $\|\mathbf{z}_t^i\|$ is bounded because each agent has bounded marginal effect on the system state.
- $\|\nabla_{\theta^i} \log \pi^i\|$ is bounded by standard score function regularity.

Applying Cauchy–Schwarz to the inner product $\langle \mathbf{x}_t - \tilde{\mathbf{x}}_t, \mathbf{z}_t^i \rangle$ and combining with the score function bound, there exists a constant $B > 0$, independent of N , such that:

$$\|\nabla \mathcal{G}(\boldsymbol{\theta}_k)\| \leq B. \quad (29)$$

Combining Equations 29 and 28 with Equation 27, we obtain:

$$\mathbb{E}_k[\|\hat{g}_{\text{DG}}(\boldsymbol{\theta}_k)\|^2] \leq 2(1 - \alpha)^2 \|\nabla J(\boldsymbol{\theta}_k)\|^2 + 2\alpha^2 B^2 + \sigma_{\text{DG}}^2. \quad (30)$$

which is the upper bound for the second moment term in Equation 25.

Substituting the inner product bound (Equation 26) and the second moment bound (Equation 30) into the descent inequality (Equation 25), we obtain the per-iteration descent bound:

$$\begin{aligned} \mathbb{E}_k[J(\boldsymbol{\theta}_{k+1})] &\geq J(\boldsymbol{\theta}_k) + \eta(1 - \alpha) \|\nabla J(\boldsymbol{\theta}_k)\|^2 - \frac{L\eta^2}{2} (2(1 - \alpha)^2 \|\nabla J(\boldsymbol{\theta}_k)\|^2 + 2\alpha^2 B^2 + \sigma_{\text{DG}}^2) \\ &= J(\boldsymbol{\theta}_k) + [\eta(1 - \alpha) - L\eta^2(1 - \alpha)^2] \|\nabla J(\boldsymbol{\theta}_k)\|^2 - \frac{L\eta^2}{2} (2\alpha^2 B^2 + \sigma_{\text{DG}}^2). \end{aligned}$$

Under the update rule (Equation 24), for the gradient signal to contribute positively to the expected change in J , the coefficient of $\|\nabla J(\boldsymbol{\theta}_k)\|^2$ must be positive. Choosing $\eta \leq \frac{1}{2L(1-\alpha)}$ ensures $\eta(1 - \alpha) - L\eta^2(1 - \alpha)^2 \geq \frac{\eta(1-\alpha)}{2}$:

$$\mathbb{E}_k[J(\boldsymbol{\theta}_{k+1})] \geq J(\boldsymbol{\theta}_k) + \frac{\eta(1 - \alpha)}{2} \|\nabla J(\boldsymbol{\theta}_k)\|^2 - \frac{L\eta^2}{2} (2\alpha^2 B^2 + \sigma_{\text{DG}}^2). \quad (31)$$

Step 3: PL condition and recursion. We now use the PL condition to convert the gradient norm bound into optimality gap contraction.

Subtracting both sides of Equation 31 from J^* (Assumption G.1):

$$J^* - \mathbb{E}_k[J(\boldsymbol{\theta}_{k+1})] \leq J^* - J(\boldsymbol{\theta}_k) - \frac{\eta(1 - \alpha)}{2} \|\nabla J(\boldsymbol{\theta}_k)\|^2 + \frac{L\eta^2}{2} (2\alpha^2 B^2 + \sigma_{\text{DG}}^2).$$

Taking full expectation and defining $\Delta_k \triangleq \mathbb{E}[J^* - J(\boldsymbol{\theta}_k)]$:

$$\Delta_{k+1} \leq \Delta_k - \frac{\eta(1 - \alpha)}{2} \mathbb{E}[\|\nabla J(\boldsymbol{\theta}_k)\|^2] + \frac{L\eta^2}{2} (2\alpha^2 B^2 + \sigma_{\text{DG}}^2).$$

By the PL condition (Assumption G.1), $\mathbb{E}[\|\nabla J(\boldsymbol{\theta}_k)\|^2] \geq 2\mu\Delta_k$. Substituting:

$$\Delta_{k+1} \leq (1 - (1 - \alpha)\mu\eta)\Delta_k + \frac{L\eta^2}{2} (2\alpha^2 B^2 + \sigma_{\text{DG}}^2).$$

Unrolling this recurrence for T iterations by the geometric series formula:

$$\Delta_T \leq (1 - (1 - \alpha)\mu\eta)^T \Delta_0 + \frac{L\eta^2}{2} (2\alpha^2 B^2 + \sigma_{\text{DG}}^2) \cdot \frac{1 - (1 - (1 - \alpha)\mu\eta)^T}{(1 - \alpha)\mu\eta}.$$

Since $\mu, \eta > 0$ and $\alpha \in (0, 1)$, we have $(1 - \alpha)\mu\eta > 0$. Combined with $\eta \leq \frac{1}{2L(1-\alpha)}$ and $\mu \leq L$ (Assumption G.1), this gives $(1 - \alpha)\mu\eta \in (0, \frac{1}{2}]$. Therefore $1 - (1 - (1 - \alpha)\mu\eta)^T \in (0, 1]$, which gives:

$$\Delta_T \leq (1 - (1 - \alpha)\mu\eta)^T \Delta_0 + \frac{L\eta}{2(1 - \alpha)\mu} (2\alpha^2 B^2 + \sigma_{\text{DG}}^2). \quad (32)$$

To achieve $\Delta_T \leq \epsilon$, it suffices to bound both terms on the right-hand side of Equation 32 by $\epsilon/2$. For the second term, setting it equal to $\epsilon/2$:

$$\frac{L\eta}{2(1 - \alpha)\mu} (2\alpha^2 B^2 + \sigma_{\text{DG}}^2) = \frac{\epsilon}{2} \implies \eta = \frac{(1 - \alpha)\mu\epsilon}{L(2\alpha^2 B^2 + \sigma_{\text{DG}}^2)}.$$

For the first term, requiring $(1 - (1 - \alpha)\mu\eta)^T \Delta_0 \leq \frac{\epsilon}{2}$ gives:

$$T \geq \frac{1}{(1 - \alpha)\mu\eta} \log \frac{2\Delta_0}{\epsilon}.$$

Substituting the chosen η :

$$T \geq \frac{L(2\alpha^2 B^2 + \sigma_{\text{DG}}^2)}{(1 - \alpha)^2 \mu^2 \epsilon} \log \frac{2\Delta_0}{\epsilon}.$$

The quantities L , μ , B , α , and $\sigma_{\text{DG}}^2 = \mathcal{O}(1)$ (by Theorem 5.2) are all independent of N . This simplifies to:

$$T = \mathcal{O}\left(\frac{L}{\mu^2 \epsilon} \log \frac{1}{\epsilon}\right),$$

completing the proof. $\square$

H Cloud Scheduling Environment

We provide complete specifications of the cloud scheduling environment used in all experiments.

H.1 Physical Environment

Cluster Configuration. The cluster consists of N servers drawn from 12 heterogeneous instance types spanning three hardware generations, with configurations based on AWS instance types. Table 4 details the full configuration.

Table 4: Server instance types used in the simulation, based on AWS instance configurations.

Gen.	Instance Type	vCPUs	Memory (GB)	η_{CPU}	η_{Mem}	Weight
Gen-1	m4.8xlarge	32	128	0.70	0.68	10%
	t3.2xlarge	16	64	0.72	0.70	3%
	t3a.2xlarge	16	64	0.75	0.72	2%
Gen-2	m5.8xlarge	32	128	0.95	0.93	20%
	c5.18xlarge	64	128	0.98	0.93	10%
	c5.12xlarge	48	96	0.97	0.93	8%
	r5.8xlarge	32	256	0.87	0.92	9%
	r5.6xlarge	24	192	0.84	0.93	7%
	m5n.12xlarge	48	192	0.96	0.92	6%
Gen-3	m6i.8xlarge	32	128	1.06	0.95	15%
	c5n.18xlarge	64	256	1.08	0.95	8%
	c6i.32xlarge	96	384	1.08	0.96	2%

The efficiency parameters η_{CPU} and η_{Mem} model performance-per-watt differences across hardware generations. Gen-1 instances (circa 2015) have $\eta_{\text{CPU}} \in [0.70, 0.75]$, Gen-2 (circa 2017) have $\eta_{\text{CPU}} \in [0.84, 0.98]$, and Gen-3 (circa 2020+) have $\eta_{\text{CPU}} \in [1.06, 1.08]$. Job completion time scales as duration/ η_{CPU} , making high-efficiency servers more productive. Server types are sampled according to the specified weights and randomly shuffled across indices at scenario generation to prevent agents from exploiting positional patterns.

Workload Generation. We employ a *bimodal task distribution* reflecting real-world workload diversity:

- **CPU-intensive tasks** ($\sim 60\%$ of arrivals): CPU requirement drawn from Pareto($\alpha = 1.7$, $x_m = 0.6$), clipped to $[0.5, 20]$ cores; memory requirement $m = c \cdot U(1.5, 3.0)$, clipped to $[0.5, 64]$ GB (low memory-to-CPU ratio).
- **Memory-intensive tasks** ($\sim 40\%$ of arrivals): CPU requirement drawn from Pareto($\alpha = 2.2$, $x_m = 0.4$), clipped to $[0.2, 8]$ cores; memory requirement $m = c \cdot U(6, 12)$, clipped to $[1, 128]$ GB (high memory-to-CPU ratio).

All tasks share the same duration distribution: Log-normal($\mu = 3.0$, $\sigma = 0.5$), clipped to $[5, 150]$ timesteps, yielding mean duration ~ 20 timesteps with high variance.

The Pareto shape parameter $\alpha \in [1.7, 2.2]$ is calibrated to match the heavy-tailed resource demand distributions observed in the Google cluster trace [21], where a small fraction of jobs consume disproportionate resources. The log-normal duration distribution is consistent with empirical observations from production clusters [22]. The bimodal CPU-to-memory ratio reflects the coexistence of compute-intensive (e.g., batch analytics) and memory-intensive (e.g., in-memory databases) workloads in modern data centers [4].

Arrival Process. Jobs arrive according to a non-homogeneous Poisson process with time-varying intensity:

$$\lambda(t) = \bar{\lambda} \cdot \max \left(0.1, 1 + 0.3 \sin \left(\frac{2\pi t}{1000} \right) + \epsilon_t \right), \quad \epsilon_t \sim \mathcal{N}(0, 0.1)$$

where the base rate $\bar{\lambda}$ is calibrated via a capacity-aware formula to achieve target utilization $\rho \in [0.80, 0.85]$ across all scales. The sinusoidal term introduces a 1000-step tidal period, capturing the diurnal traffic patterns observed in production systems [3], while ϵ_t adds stochastic perturbation.

H.2 Agent Interface

At each timestep, newly arrived jobs are assigned to dedicated agents that select placement actions concurrently. We set the number of job-dispatch agents equal to the number of servers so that newly arrived jobs can be dispatched without accumulating at the assignment stage. Under this convention, N denotes both the server scale and number of job-dispatch agents. We evaluate configurations from $N=2$ to $N=1500$.

Scale Configurations. Table 5 summarizes the experimental configurations. Each scale is defined by the number of servers N , the number of action-space clusters K , the number of servers per cluster $S = N/K$, and the target cluster utilization ρ used to calibrate the arrival rate $\bar{\lambda}$ (see Arrival Process above). Each episode consists of 3000 timesteps, with job arrivals during the first 2000 and the remaining 1000 reserved as a drain period.

Table 5: Scale configurations used in the experimental evaluation.

Servers (N)	Clusters (K)	Servers/Cluster (S)	Episode Length	Target Util. (ρ)
2	2	1	3000	$[0.80, 0.85]$
5	5	1	3000	$[0.80, 0.85]$
10	10	1	3000	$[0.80, 0.85]$
20	20	1	3000	$[0.80, 0.85]$
50	25	2	3000	$[0.80, 0.85]$
100	25	4	3000	$[0.80, 0.85]$
200	40	5	3000	$[0.80, 0.85]$
500	50	10	3000	$[0.80, 0.85]$
1000	50	20	3000	$[0.80, 0.85]$
1500	75	20	3000	$[0.80, 0.85]$

Action Space. For scales with $N \leq 20$, each agent directly selects one of the N servers ($K = N$, $S = 1$). For scales with $N \geq 50$, the action space becomes prohibitively large, so we first sort servers by capacity ($c_k^{\text{CPU}}, c_k^{\text{Mem}}$) and then partition them into $K < N$ clusters of $S = N/K$ servers with similar profiles. Each agent selects a *cluster* (action space of size K). Within the selected cluster, the simulator uses a deterministic Best-Fit rule to place the job on the server with the highest current utilization that still has sufficient free resources. If no server in the selected cluster can accommodate the job, it is queued on the least-loaded server in that cluster.

Observation Space. Each agent receives an observation consisting of:

- **Global server state** ($7N$ dims): For each server $k \in \{1, \dots, N\}$, we include CPU utilization, memory utilization, local queue length (clipped at 50), CPU efficiency, memory efficiency, CPU capacity, and memory capacity.
- **All job information** ($2N_{\text{agents}}$ dims): CPU and memory requirements for jobs arrived at current timestep.
- **Job features** (2 dims): CPU and memory requirements of this agent’s assigned job.
- **Temporal feature** (1 dim): Timestep t/T_{max} .
- **Agent identifier** (1 dim): Agent index, used with a learnable embedding (see below).

Observation Compression. The raw observation provides the full server and job information needed for placement decisions. However, because it explicitly enumerates all servers and newly arrived jobs, its dimension grows quickly with N . We therefore replace the two high-dimensional components separately. Per-server states are represented by cluster summaries, and per-job information is represented by job summaries.

- **Cluster summary.** Each cluster is represented by a 28-dimensional vector consisting of five groups of features.
 - **CPU utilization.** Min, Q25, median, Q75, max, mean, and std.
 - **Memory utilization.** Min, Q25, median, Q75, max, mean, and std.
 - **CPU-memory joint distribution.** Correlation, fraction CPU-heavy, fraction memory-heavy, and fraction balanced.
 - **Queue congestion.** Total, max, mean, std, and fraction non-empty.
 - **Capacity and efficiency.** Free CPU fraction, free memory fraction, mean efficiency, cluster size, and overloaded fraction.
- **Job summary.** Jobs arriving at the current timestep are represented by 6 aggregate job-demand features, including the mean, std, and max of CPU demand and the mean, std, and max of memory demand.
- **Retained features.** The agent’s own job features, temporal feature, and agent identifier are kept unchanged.

The total observation dimension is therefore $28K + 10$, consisting of $28K$ cluster-summary features, 6 job-summary features, and 4 retained agent/time features. Since the compression is most beneficial once each cluster contains multiple servers, we apply it for $N \geq 200$ ($S = N/K \geq 5$), achieving up to $6.4\times$ reduction compared with the raw representation.

Agent ID Embedding. To enable parameter sharing across heterogeneous agents, we embed each agent’s normalized index via a learnable `nn.Embedding` layer (dimension 16), which is concatenated with the remaining observation features before the first hidden layer. This allows a single shared network to learn agent-specific behaviors.

H.3 Reward Function

The shared reward signal is a weighted combination of two objectives:

$$r_t = - \left(w_1 \cdot \underbrace{\frac{Q_{\text{global}}(t) + \sum_{k=1}^N Q_k(t)}{N}}_{\text{Queue Penalty (Latency)}} + w_2 \cdot \underbrace{\frac{\sum_{k=1}^N \frac{L_k(t)}{\eta_k}}{C_{\text{total}}}}_{\text{Energy Penalty}} \right)$$

where $Q_{\text{global}}(t)$ is the global buffer size, $Q_k(t)$ is the local queue at server k , $L_k(t)$ is the workload (CPU + memory) at server k , η_k is the server’s efficiency, and C_{total} is the total cluster capacity. We set $w_1 = 1$ and $w_2 = 20$ so that both penalties contribute comparably to the reward. Without the multiplier, the queue penalty typically ranges in $[0, 20]$, whereas the normalized energy ratio remains below 1. Load-balancing (utilization variance) is handled entirely by the DG-PG guidance signal and is therefore excluded from the reward.

I Training Configuration

I.1 Training and Evaluation Protocol

For training, each run uses an independent random seed for model initialization and stochastic simulation. We run 3 seeds for each α and scale in the α -selection experiment, 5 seeds for each method and scale in the controlled comparison, and 5 seeds for $N \leq 100$ and 3 seeds for $N \geq 200$ in the scalability experiment. For evaluation, every saved checkpoint is tested on the same 10 held-out test seeds, and we report the checkpoint with the highest mean test reward. DG-PG, MAPPO, and IPPO report mean $\pm$ standard deviation across training seeds, while Best-Fit and Random are deterministic baselines evaluated on the same test seeds.

I.2 Training Settings

DG-PG, MAPPO, and IPPO are all trained with PPO using parameter sharing across agents. Across all learned methods, we set the discount factor to $\gamma = 0.99$ and the Generalized Advantage Estimation parameter to $\lambda = 0.95$, and use Adam with separated actor-critic updates, independent gradient clipping, return normalization before critic updates, and advantage normalization before policy updates.

However, MAPPO and IPPO with neural-network actor and critic models failed to learn competitive policies despite extensive tuning (Appendix J). The controlled comparison therefore uses linear actor and critic models, together with stronger stabilization settings, for all three methods. This keeps the actor and critic function classes matched within the comparison, while the guidance-weight and scalability experiments use the neural-network DG-PG configuration.

I.2.1 Linear Models (Controlled Comparison)

For the controlled comparison at $N \in \{2, 5, 10\}$, all three methods use the same linear actor $\pi(a|s) = \text{softmax}(Ws + b)$ and linear critic $V(s) = w^\top s + b$. They also share the same core training parameters. Table 6 lists the full configuration.

Table 6: Hyperparameters for controlled comparison (Section 6.3).

Parameter	Value
Model class	Linear (single layer, no hidden units)
Learning rate	1×10^{-3} (linear decay to 1%)
PPO clipping ϵ	0.2
Critic epochs / Actor epochs	20 / 3 (split-epoch)
Mini-batch size	10,000
Gradient clipping	max norm 10.0
Huber loss δ	10.0
Parallel environments	4
Rollouts per episode	24
DG-PG entropy coefficient	0.01 (exponential decay to 10^{-4})
MAPPO / IPPO entropy coefficient	0.005 (fixed, no decay)
DG-PG training episodes	200
MAPPO / IPPO training episodes	500
DG-PG α schedule	dynamic schedule ($0.9 \rightarrow 0.2$)

Several shared parameters were set to stabilize MAPPO and IPPO, based on the tuning documented in Section J:

- **Split-epoch training** (critic 20 epochs, actor 3 epochs): MAPPO and IPPO require a well-trained critic before the actor receives meaningful advantage signals. A single linear layer needs many gradient steps to fit the value function. Without this asymmetric schedule, advantage estimates are too noisy for stable policy updates.
- **Large mini-batch** (10,000) and **high rollout count** (24): With smaller batches (e.g., 256–1,024), MAPPO and IPPO exhibit large reward oscillations and do not converge. Multi-

agent policy gradients in this environment have high variance due to heterogeneous servers, bimodal workloads, and non-stationary arrivals, requiring large batches to average over noise.

These settings are based on the MAPPO implementation guidelines in [30], with additional stabilization needed for our scheduling environment. DG-PG uses the same linear architecture and the shared training settings for fair comparison.

The remaining differences also favor MAPPO/IPPO:

- **Entropy:** MAPPO/IPPO use a fixed coefficient (0.005) rather than decay, because entropy decay caused premature policy collapse in our tuning. The fixed value was selected as the best-performing setting across grid search.
- **Training length:** MAPPO/IPPO are trained for 500 episodes ($2.5\times$ longer than DG-PG’s 200), providing additional convergence time.

I.2.2 Neural Network Models

For the DG-PG experiments in Sections 6.2 and 6.4, we use neural-network actor and critic models rather than the linear models used in the controlled comparison. Both models are two-layer feedforward networks with Tanh activations.

Hyperparameters.

- **Learning rate:** 1×10^{-4} (exponential decay)
- **Entropy coefficient:** 0.02 (exponential decay to 10^{-4})
- **PPO update epochs:** $K = 4$ (unified for actor and critic)
- **Gradient clipping:** max norm 0.5
- **Guidance signal:** See Appendix C
- **Guidance weight α in Section 6.2:** fixed $\alpha \in \{0.2, 0.4, 0.6, 0.8\}$
- **Guidance weight α in Section 6.4:** dynamic schedule for $N \in \{5, 10, 20, 50\}$ and fixed $\alpha = 0.9$ for $N \geq 100$
- **Guidance normalization:** running guidance statistics are updated with momentum 0.99, and the normalized guidance signal is clipped to $[-3, 3]$

Scale-Specific Settings. For the scalability experiments, we use scale-specific settings according to the input size and training cost, as summarized in Table 7.

Table 7: Scale-specific hyperparameters.

Servers (N)	Hidden Dimension	Mini-batch	Rollouts	Parallel Environments	PPO Clip (ϵ)	Active Samples
10	128	512	12	4	0.2	$\sim 120,000$
20	128	512	12	4	0.2	$\sim 240,000$
50	256	1024	8	4	0.4	$\sim 400,000$
100	512	1024	6	2	0.4	440,000 (cap)
200	1024	1024	2	1	0.4	440,000 (cap)
500	1024	1024	1	1	0.4	440,000 (cap)
1000	1024	1024	1	1	0.4	440,000 (cap)
1500	1024	1024	1	1	0.4	440,000 (cap)

As the observation and action spaces grow, we increase the hidden dimension of the actor and critic. At smaller scales, each rollout contains fewer active-agent transitions, so we use smaller mini-batches, more rollouts per episode, and more parallel environment workers to improve trajectory diversity. For $N \geq 50$, we use a wider PPO clipping ratio $\epsilon = 0.4$, because DG-PG tolerated more aggressive policy updates and smaller clipping ratios slowed early learning. At larger scales, the number of active samples collected per episode grows rapidly, so we cap the retained active samples used for PPO updates at approximately 440,000.

Training Stability. Since the guidance signal stabilizes DG-PG by reducing gradient variance (Theorem 5.2), DG-PG does not require the stabilization mechanisms used for MAPPO and IPPO,

such as split-epoch training, larger mini-batches, or more rollouts. Moreover, our experiments show that DG-PG can go beyond conservative stabilization settings and tolerate more aggressive PPO updates, such as a wider clipping ratio.

J Baseline Failure Analysis

MAPPO and IPPO fail to learn competitive policies with neural-network actor and critic models due to a fundamental *actor-critic update frequency dilemma*: the actor and critic have conflicting requirements for gradient update frequency, and no configuration resolves both simultaneously. We verified this across ~ 20 training runs at $N=5$ and $N=2$, varying batch size, update epochs, entropy coefficient, value normalization, gradient clipping, and model architecture. Table 8 lists representative runs at $N=5$ with a two-layer feedforward network (128 hidden units). Each representative run uses approximately 200,000 training samples per episode.

Table 8: Representative MAPPO runs with neural-network actor and critic models at $N=5$.

Run	Key Change	Mini-batch	PPO Epochs	c_{ent}	Observed Failure
0	DG-PG defaults	256	4	0.02 (decay)	Entropy collapse ($1.59 \rightarrow 0.26$)
1	+ValueNorm	256	10	0.01	Entropy collapse ($1.60 \rightarrow 0.88$)
2	K : $10 \rightarrow 4$	256	4	0.01	Slow entropy collapse ($1.54 \rightarrow 1.36$)
3	c_{ent} : $0.01 \rightarrow 0.03$	256	4	0.03	Entropy bonus drowns advantage
4	Align with MAPPO defaults	50,000	10	0.01	Critic too slow (explained variance ≤ 0.30)
5	Batch: $50K \rightarrow 20K$	20,000	15	0.01	Critic still insufficient
6	Batch: $20K \rightarrow 10K$	10,000	15	0.01	No convergence
7	+Potential-based shaping	10,000	15	0.01	Shaping signal negligible

J.1 The Gradient Update Frequency Dilemma

In MAPPO and IPPO training, the most sensitive quantity is the number of mini-batch gradient updates performed after each episode. When this number is large, the critic receives enough optimization steps, but the actor repeatedly updates on noisy advantages and quickly loses entropy. When this number is small, the actor remains stable, but the critic does not learn accurate value estimates. Table 8 summarizes these outcomes in three regimes:

- **High update frequency** (Runs 0–3): The critic learns well (explained variance reaches 0.7), but the actor undergoes entropy collapse, converging to a near-deterministic, suboptimal action within 50–70 episodes, after which performance degrades irreversibly.
- **Low update frequency** (Runs 4–5): Entropy remains stable, but the critic receives too few updates to learn accurate value estimates (explained variance ≤ 0.30), leaving the actor with uninformative advantages.
- **Intermediate update frequency** (Run 6): The update frequency avoids immediate entropy collapse and improves critic explained variance to 0.35–0.49, but the training curve oscillates for 1,000 episodes with no convergence trend.

J.2 Attempts to Break the Gradient Update Frequency Dilemma

We tested four structural modifications, none of which resolved the fundamental conflict:

ValueNorm [30] (Run 1). Value normalization improved critic fitting by increasing the critic explained variance from 0.0 to 0.7 under the high-update-frequency setting. However, the actor still suffered entropy collapse, showing that better value normalization alone does not resolve the update-frequency dilemma.

MAPPO implementation alignment (Run 4). We aligned the configuration with the MAPPO reference implementation from Yu et al. [30], including the mini-batch size ($256 \rightarrow 50,000$), value loss coefficient ($0.5 \rightarrow 1.0$), Huber loss parameter ($1.0 \rightarrow 10.0$), gradient clipping threshold ($0.5 \rightarrow 10.0$), entropy setting, and rollout count ($12 \rightarrow 24$). This eliminated entropy collapse by reducing gradient updates from 3,064 to 40 per episode, but made the critic too slow.

Potential-based reward shaping (Run 7). Potential-based reward shaping with $\Phi(s) = -\text{Var}(\text{utilization})$ produced negligible shaping signals. This indicates that the dominant bottleneck is gradient variance from cross-agent credit assignment, rather than temporal credit assignment.

Asynchronous actor-critic ($N=2$). Separate critic-then-actor optimization at the smallest scale produced near-zero KL divergence and 0% PPO clipping. This suggests that the policy-gradient signal remains too noisy to drive meaningful policy updates even with only 2 agents.

J.3 Transition to Linear Models

These failures make neural-network MAPPO and IPPO unsuitable for a controlled comparison in this environment. We therefore use matched linear actor and critic models, $\pi(a|s) = \text{softmax}(Ws + b)$ and $V(s) = w^\top s + b$, for all three methods. The linear models reduce function-approximation capacity and make critic fitting more stable, which allows the split-epoch training scheme in Section I to be effective. Even under this simplified model class, MAPPO and IPPO still require additional stabilization, including split-epoch training, large mini-batches, and elevated rollout counts. This gives the baselines a more favorable training configuration and makes the comparison a conservative test of DG-PG’s guidance signal.

These failures reflect not only the cross-agent variance scaling analyzed in Section 3, but also the statistical difficulty of the cloud scheduling environment.

- *Server heterogeneity* ($\eta_{\text{CPU}} \in [0.70, 1.08]$): The best placement decisions depend strongly on the server mix, making value estimation more scenario-dependent.
- *Non-stationary arrivals*: Poisson processes with time-varying rates make episodes structurally different, reducing the transferability of value estimates across episodes.
- *Bimodal workloads*: Jobs arrive with two distinct resource profiles, increasing variability in returns and advantage estimates.

Together, these factors make both value estimation and policy learning difficult, because the policy must infer a highly state-dependent placement rule from noisy multi-agent advantage estimates. DG-PG addresses this difficulty by providing an analytical descent direction during training, rather than relying solely on end-to-end discovery from returns.

K Computational Resources

All experiments were conducted on a single Apple M1 chip (8-core CPU, 16 GB unified memory, circa 2021) using CPU-only computation, without any GPU acceleration.

Convergence speed. A key practical advantage of DG-PG is its rapid convergence. We define convergence as the first episode at which reward reaches within 10% of the best observed reward. As shown in Table 9, the average convergence episode across the reported scales from $N=10$ to $N=1500$ is 11.4. Both convergence episode and per-episode time are averaged across training seeds.

Table 9: DG-PG convergence and wall-clock timing on Apple M1 (CPU-only).

Agents (N)	Converge Episode	Converge Time	Per-Episode
10	16.6	~2.9 min	~10.5 s
20	11.6	~3.7 min	~18.9 s
50	7.4	~5.0 min	~40.9 s
100	6.8	~16.2 min	~2.4 min
200	9.0	~37.8 min	~4.2 min
500	12.0	~42.1 min	~3.5 min
1000	9.7	~41.1 min	~4.2 min
1500	17.7	~99.0 min	~5.6 min

The non-monotonic per-episode time is due to scale-specific rollout counts and the active-sample cap. For example, the $N=500$ runs use one rollout per episode, whereas $N=200$ uses two rollouts, so the larger configuration can have lower wall-clock time per episode despite the larger system size.

Overall, these results indicate that DG-PG’s scalability is not only statistical but also computationally practical. Even at $N=1500$, the policy reaches the convergence threshold in about 99 minutes on CPU-only consumer hardware. This suggests that the analytical guidance reduces the training burden enough to make large-scale cooperative MARL feasible without specialized accelerators.