The 18th International Conference of Iranian Operations Research Society
30-31 October 2025
هجدهمین کنفرانس بین المللی انجمن ایرانی تحقیق در عملیات
۸ و ۹ آبان ۱۴۰۴

MARL-Enhanced Partial Reinforcement Optimizer: A Novel Evolutionary Algorithm

Seyed Ali Alavizadeh[1], Hooman Kaseban[2], Omid Solaymani Fard[3]
[1]Student, Department of Applied Mathematics, Faculty of Mathematical Sciences, Ferdowsi University of Mashhad, Iran; seyyedali.alavizadeh@studio.unibo.it
[2]Student, Department of Applied Mathematics, Faculty of Mathematical Sciences, Ferdowsi University of Mashhad, Iran; hoomankaseban@mail.um.ac.ir
[3]Associate Professor, Department of Applied Mathematics, Faculty of Mathematical Sciences, Ferdowsi University of Mashhad, Iran; soleimani@um.ac.ir

Abstract

We propose Multi-Agent Reinforcement Learning with Partial Reinforcement Optimizatizer (MARL-PRO), a novel population-based optimizer that extends the Partial Reinforcement Optimizer (PRO) by incorporating cooperative multi-agent reinforcement learning. In MARL-PRO, each agent employs tabular Q-learning over a factorized discrete action space—subspace size, step scale, and direction (attraction or repulsion)—while a sensitivity vector guides updates toward the most responsive coordinates. A simple two-phase schedule further balances exploration and exploitation. Agents coordinate indirectly through shared incumbents and peer-based repulsion, fostering both convergence and diversity. Periodic local refinement and diversity resets enhance stability and prevent collapse. Evaluations on four benchmark functions (Sphere, Rosenbrock, Schwefel 1.2, and Schwefel 2.22) demonstrate that MARL-PRO consistently achieves faster and more stable convergence than PRO, particularly on ill-conditioned and nonseparable landscapes. These results highlight MARL-PRO's potential as a scalable and interpretable framework for black-box optimization in distributed and cooperative environments.

Keywords: Multi-Agent Reinforcement Learning; Partial Reinforcement Optimizer; Non-Linear Optimization.

Introduction

Highly complex optimization problems require algorithms that can balance adaptive exploration with focused exploitation. The Partial Reinforcement Optimizer (PRO) [1] is a recently proposed evolutionary algorithm inspired by the psychological theory of the Partial Reinforcement Effect. In PRO, candidate solutions are modeled as learners, with each behavior represented as a decision variable. Stimulation corresponds to targeted perturbations of learners' behavior vectors, while responses—captured through objective-function evaluations—guide a dynamic scheduling mechanism. This mechanism simulates partial reinforcement by adjusting the priority of specific behaviors, thereby mimicking positive and negative reinforcement to elicit more effective responses. Through this process, PRO achieves a balance between exploration and exploitation. The algorithm has been rigorously validated on diverse benchmark suites and shown to outperform several recent memetic and classical evolutionary methods, including GA [2], PSO [3], and Grey Wolf Optimization (GWO) [4].

In this study we introduce Multi-Agent Reinforcement Learning PRO (MARL-PRO), a principled extension that integrates multi-agent reinforcement concepts into PRO's priority-based framework. In MARL-PRO, a population of cooperative agents jointly explores the search space while the scheduling mechanism mediates inter-agent credit assignment and behavior prioritization, thereby enhancing coordination in distributed and federated environments. We evaluate MARL-PRO on some benchmark functions employing two identical statistical analyses and comparative baselines to ensure a fair assessment. Results indicate that MARL-PRO retains the strengths of PRO while providing improved scalability and stability in multi-agent and distributed optimization scenarios.

Multi-Agent Reinforcement Learning

Formulation

A formalization for multi-agent reinforcement learning is as:

$$G=(N,S,\{A_i\}_{i=1}^{N},\{O_i\}_{i=1}^{N},P,\{R_i\}_{i=1}^{N},\gamma),$$

where N is the number of agents, S the state space, A_i and O_i the action and observation spaces of agent i, $P(s' \vee s,a)$ the transition kernel for joint action $a=(a_1,\dots,a_N)$, R_i the per-agent reward, and $\gamma \in (0,1)$ the discount. Each agent chooses actions with a (possibly stochastic) policy $\pi_i(a_i \vee o_i,h_i)$ that can depend on current observation o_i and its local history h_i. The learning goal is to optimize a joint policy $\pi=(\pi_1,\dots,\pi_N)$ to maximize expected return(s) [5].

A widely used paradigm is **Centralized Training, Decentralized Execution (CTDE)**. During training, agents may access global information (e.g., state s or other agents' actions) through a centralized critic, yet the learned policies execute with only local observations [5]. CTDE mitigates non-stationarity (each agent sees others learning) while preserving decentralization at test time.

We consider black-box, derivative-free minimization $\min_{x\in[l,u]} f(x)$. A MARL view treats *search* as sequential decision making [5]:

- **Parallel exploration:** many agents concurrently probe different regions, increasing the chance of finding good basins in rugged landscapes.
- **Diversity by design:** heterogeneity in policies/actions acts like an implicit ensemble, reducing premature convergence.
- **Credit assignment over *moves*, not points:** MARL can learn which *kinds* of moves (step sizes, directions, subspace choices) produce consistent improvement.
- **Anytime performance:** population methods yield useful intermediate solutions; RL can adapt exploration → exploitation over time.

Canonical MARL Strategies

Value-based (discrete actions).

Independent Q-Learning (IQL) learns $Q_i(o_i,a_i)$ per agent; value decomposition (VDN) and QMIX learn factored joint Q with mixing networks for cooperation. Pros: sample-efficient for small discrete action spaces. Cons: scaling to large/continuous action spaces is hard [5].

Policy-based.

REINFORCE and PPO-style methods directly optimize π_θ with likelihood-ratio gradients and entropy bonuses (often under CTDE with centralized critics). Pros: handle stochastic policies and constraints; good with large action spaces. Cons: higher variance; need careful tuning.

Actor–Critic.

MADDPG (continuous) and multi-agent PPO variants use centralized critics $Q_i(s,a)$ with decentralized actors $\pi_i(o_i)$. Pros: stabilizes training with critics; supports continuous actions. Cons: critics must generalize across joint actions.

Exploration strategies.

ε-greedy, Boltzmann/softmax over action-values, entropy regularization, Gaussian parameter noise, optimism/UCB-like bonuses.

Coordination and credit assignment.

Team rewards vs. per-agent rewards; difference rewards and counterfactual baselines reduce spurious credit assignment; value factorization enables cooperative behaviors while training per-agent policies [5].

Stabilizers.

Target networks, replay buffers (for off-policy), recurrent policies for partial observability, and curriculum/phase schedules to anneal exploration.

In this article, MARL-PRO adopts a *value-based* multi-agent strategy with *independent tabular Q-learning* per agent. Each agent maintains three small Q-tables for factorized discrete decisions—subspace size $\lambda \in \Lambda$, step scale $\beta \in B$, and direction dir $\in \{0,1\}$—and selects actions via ε-greedy. The update is standard TD(0):

$$Q(\mathrm{s,a}) \leftarrow Q(\mathrm{s,a}) + \alpha(r + \gamma \max_{\mathrm{a'}} Q(\mathrm{s',a'}) - Q(\mathrm{s,a})) \qquad \mathrm{,r:} = -\big(f(x^{\mathrm{new}}) - f(x)\big),$$

with a *two-state* phase variable $s \in \{0,1\}$ (early vs. late iterations).

In this work, our action space is intentionally discrete and compact (interpretable knobs on the move operator), which makes tabular Q stable and sample-efficient. The objective is black-box and potentially non-smooth; tabular value-learning avoids score-function or critic approximation. Factorization over (λ,β,dir) keeps the tables tiny (no curse of dimensionality from joint actions) while preserving expressive combinations [5].

Although there is no centralized critic (hence not CTDE in the strict sense), agents *share global cues*: the current incumbent best $x^\star$ and a partner $x^{(p)}$ sampled from strictly-better peers. These signals stabilize learning and reduce non-stationarity by aligning improvements toward common objectives.

For derivative-free optimization, encode *actions* as *move operators* over x (e.g., step size, subspace size, direction). Use a scalar team reward r$= -\big(f(x^{\mathrm{new}}) - f(x)\big)$ so improvements are positive. Keep the action space compact and interpretable (discrete bins) to allow tabular/value methods or lightweight function approximation. Use CTDE-inspired cues (e.g., a shared best incumbent) to reduce non-stationarity and align agents [5].

MARL-PRO: Multi-Agent RL with Partial Reinforcement Optimization

Problem Setup

We seek

$$\min_{x \in \mathbb{R}^d} f(x) \qquad s.t. \qquad \ell \le x \le u. \qquad (1)$$

MARL-PRO maintains a population of agents $\{L_k\}_{k=1}^{P}$; each agent *learns which partial move to take* at each step through tabular Q-learning over a small, factorized action space.

Agent Anatomy and Factorized Actions

Each agent stores: current position x, fitness $f(x)$, and a **sensitivity vector** $s \in R_0^d$ measuring coordinate-wise responsiveness to past updates [5]. The discrete (factorized) action heads are:

1. **Subspace size** $\lambda \in \Lambda$ (how many coordinates to update; we pick the top-λ indices of s).
1. **Step scale** $\beta \in B$ (a multiplicative step magnitude).
2. **Direction flag** dir $\in \{0,1\}$:
   - dir=0: *attraction* toward the current global best $x^\star$ (exploitation).
   - dir=1: *repulsion* away from a better partner $x^{(p)}$ (diversification).

Actions are selected with ε-greedy from three tabular Q-functions $\left(Q_{\lambda},Q_{\beta},Q_{\text{dir}}\right)$. A simple two-state schedule ($s \in \{0,1\}$ for early/late phases) conditions the Q-tables: early phase favors larger λ / bolder steps; late phase shrinks moves.

In this method, We factorize the action space because a full joint action over (λ,β,dir) scales multiplicatively; factorization preserves expressivity while keeping learning stable and sample-efficient.

MARL Strategy Suite Used in MARL-PRO

MARL-PRO adopts a cooperative, value-based scheme with three discrete, factorized action heads per agent: *subspace size* $\lambda \in \Lambda$, *step scale* $\beta \in B$, and *direction* dir $\in \{0,1\}$ (attract/repel). Each design choice is purposeful [5]:

1. **Factorized discrete actions (value-based).** Tabular Q-learning over small action sets is stable and sample-efficient. Factorization avoids the combinatorial blow-up of a joint table while permitting expressive combinations (λ,β,dir).

1. **Sensitivity-guided subspace selection.** A dimension-wise *sensitivity* vector s encodes credit for coordinates that historically reduced f. At each step, the agent updates only the top-λ indices B by s_i. This combats the curse of dimensionality and channels computation to promising subspaces.

2. **Attract & repel directions.** dir=0 (attraction) moves toward $x^{\star}$ for exploitation; dir=1 (repulsion) moves away from $x^{(p)}$ for diversity and local-minima escape. A single bit gives a powerful exploration–exploitation lever.

3. **Phase-aware scheduling (two macro-states).** A coarse state $s \in \{0,1\}$ (early vs. late iterations) conditions the three Q-heads: early favors larger λ and bolder β; late favors smaller λ and conservative β. This CTDE-inspired cue reduces non-stationarity without a centralized critic.

4. **Diversity maintenance.** If std$(s) \approx 0$ and the agent is not the incumbent best, re-seed (x,s) from the bounds. This prevents population collapse and keeps search radiating.

5. **Global–local hybridization.** Periodically (every K iterations), a bounded local solver refines $x^{\star}$. Exploration discovers basins; local search polishes—improving anytime performance with negligible algorithmic overhead.

Reward System and Credit Assignment

The objective function is the environment. After proposing x^{new} and evaluating $f(x^{\text{new}})$, agents receive a shaped, bounded, *immediate* reward:

$$r_t=\text{clip}\left(\frac{f(x_t)-f(x_{t+1})}{|f(x_t)|+\mathcal{E}}-\lambda_{cost}.\frac{\lambda}{d}-step_{pen}.\frac{\|x_{t+1}-x_t\|_2}{\|u-\ell\|_2},r_{min},r_{max}\right) \qquad (2)$$

where the first term is *normalized improvement* (scale-free), the second discourages wasteful large subspaces, and the third dampens overshooting. ϵ avoids division by zero and $\left(r_{\text{min}},r_{\text{max}}\right)$ bound the signal for stable learning (e.g., $[-1,1]$). In practice, a minimal, very stable variant suffices:

$$r_t=-\left(f(x^{\text{new}})-f(x)\right) \text{ (used in our implementation).}$$

We apply credit at two levels:

- **Action-level (Q-learning).** For each head $Q \in \left\{Q_{\lambda},Q_{\beta},Q_{\text{dir}}\right\}$, $Q(\text{s,a}) \leftarrow Q(\text{s,a})+\alpha\left(r_t+\gamma\max_{\text{a'}}Q(\text{s,a'})-Q(\text{s,a})\right)$, with ε-greedy selection and a small floor ε_{min}.

- **Dimension-level (sensitivities).** If f improves, $s_i \leftarrow s_i(1+\text{rr}/2)$ for $i \in B$; else $s_i \leftarrow s_i(1-\text{rr})$.

Stages of PRO (Partial Reinforcement Optimization Operator)

Given actions (λ,β,dir) and state $(\mathbf{x},\mathbf{s})$:

P1. **Select subspace.** $B \leftarrow$ Top-λ indices of $\boldsymbol{s}$.

P2. **Form a stimulus.**

$$\Delta_B = \begin{cases} x_B^{\star} - x_B, & \text{dir=0(attractive)} \\ x_B - x_B^{(p)}, & \text{dir=1(repulsive)} \end{cases}, \Delta_{\bar{B}} = 0,$$

where $\bar{B}$ denotes the complement of B.

P3. **Compute step factor.** With phase τ=t/T,

$$\eta = \tau + \beta \cdot \overline{\left(\frac{s_B}{\max_j s_j}\right)},$$

i.e., anneal by time and amplify by the (normalized) responsiveness of the active subspace.

P4. **Projected proposal.**

$$x^{\text{new}} = \Pi_{[\ell,u]}(x+\eta\Delta).$$

P5. **Evaluate and accept.** If $f(x^{\text{new}})<f(x)$, accept and boost s_B; otherwise penalize s_B.

Stages of MARL (Control Loop Around PRO)

For t=1, ... ,T:

1. **Phase and exploration.** Set $s \in \{0,1\}$ (early/late), decay $\varepsilon \leftarrow \max(\varepsilon_{\min}, 0.995\varepsilon)$.
2. **Partnering.** Sample $x^{(p)}$ from agents with strictly better fitness (fallback to self).
3. **Action selection.** Each head chooses a via ε-greedy over its Q-row for state s; decode (λ,β,dir).
4. **Apply PRO.** Run (P1)–(P5) to obtain x^{new}.
5. **Reward and updates.** Compute r_t (Eq. 2 or the minimal variant) and update Q_{λ},Q_{β},Q_{dir}; update s on B.
6. **Incumbent & diversity.** Update $x^{\star}$ if improved; if std$(s) \approx 0$ and not best, re-seed (x,s).
7. **Local polish (periodic).** If $K|t$, run a bounded local solver from $x^{\star}$ and adopt improvements.

How MARL Settles the Actions of PRO

MARL-PRO learns *which PRO moves work where* by repeatedly applying TD updates to three small Q-heads:

- **Convergence of preferences.** Because rewards are immediate and shaped by actual improvement, Q-values for beneficial choices (e.g., small λ late in training, moderate β, dir=0 near optima) steadily dominate.
- **Phase conditioning.** Separate rows for s=0 and s=1 encode different regimes: early rows learn exploratory settings; late rows learn exploitative settings without interfering with early behavior.
- **Population effect.** All agents share $x^{\star}$ and draw $x^{(p)}$ from better peers. This aligns incentives toward the same objective and implicitly coordinates the learned preferences, yielding a soft consensus on effective (λ,β,dir) patterns.

- **Stability.** Bounded rewards, ε floors, and occasional re-seeding prevent value blow-ups and mode collapse, so the settled policies remain useful throughout the run.

PRO Operator: Partial, Sensitivity-Weighted Moves

At iteration t (budget T), define $\tau = t/T$. Select B = indices of the λ largest entries of s. Form a *stimulus* vector

$$\Delta_B = \begin{cases} x_B^\star - x_B, & \text{dir}=0\,(\text{attractive}) \\ x_B - x_B^{(p)}, & \text{dir}=1\,(\text{repulsive}) \end{cases}, \Delta_{\bar{B}} = 0,$$

then compute a step factor

$$\eta := \tau + \beta \cdot \overline{\left(\frac{s_B}{\max_j s_j} \right)},$$

where the bar denotes the mean over B. The proposal is projected to the box:

$$x^{\text{new}} = \Pi_{[\ell,u]}(x + \eta\Delta).$$

Why these design choices?

- **Partial moves** ($\lambda \ll d$) concentrate effort on the most promising coordinates, combating the curse of dimensionality.
- **Sensitivities** s provide *dimension-wise credit*: coordinates that historically yielded improvement become more likely to be chosen again.
- **Attract/repel** gives a minimal yet powerful coordination mechanism: attraction accelerates convergence, repulsion preserves diversity and helps escape local minima.
- **Phase factor** τ anneals step sizes automatically (explore → exploit).

Credit Assignment and Learning Updates

After evaluating $f(x^{\text{new}})$:

(i) Sensitivity update (local credit).

For updated indices $i \in B$,

$$s_i \leftarrow \begin{cases} s_i(1 + \text{rr}/2), & \text{if } f(x^{\text{new}}) < f(x) \\ s_i(1 - \text{rr}), & \text{otherwise} \end{cases}$$

(and s_i unchanged for $i \notin B$).

(ii) Q-learning (action credit).

Define reward $r := -\big(f(x^{\text{new}}) - f(x)\big)$ (positive when we improve). For each head $Q \in \{Q_\lambda, Q_\beta, Q_{\text{dir}}\}$ with learning rate α and discount γ,

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

with phase state $s'=s$ (two-phase schedule), and ε-greedy selection with a small floor $\varepsilon_{\min}$.

Diversity reset.

If $\text{std}(s) \approx 0$ and the agent is not the incumbent best, re-seed (x,s) uniformly to avoid collapse.

Partner Selection and Local Refinement

A partner $x^{(p)}$ is sampled from agents with strictly better fitness (fallback to self if none); this provides a *directional signal* even far from the incumbent best. Every K iterations, MARL-PRO runs a bounded local optimizer from $x^{\star}$ to polish the best solution and tighten convergence.

Complexity and Defaults

Selecting B via sorting costs $O(d \log d)$ (or $O(d)$ via partial selection); per-iteration cost is $O(\mathrm{P}\ \mathrm{d} \log d)$. Memory is $O(\mathrm{Pd})$ plus three small tabular heads for the two phases. Stable defaults: P=30–50, T=300–1000, α=0.05–0.2, γ=0.9, $\varepsilon_0 = 0.1$–0.3, rr=0.5–0.8, K=20–50.

Algorithm 1 MARL-PRO (Multi-Agent RL with Partial Reinforcement Optimization)

Require: f, bounds $[\boldsymbol{\ell}, \boldsymbol{u}]$, dimension d, population P, iterations T, $\alpha, \gamma, \varepsilon_0$, rr, local period K

Initialize agents $\{L_k\}_{k=1}^{P}$ with $\boldsymbol{x}_k \sim \mathcal{U}([\boldsymbol{\ell}, \boldsymbol{u}])$, $\boldsymbol{s}_k \sim \mathcal{U}(0.9, 1.0)$, and $Q_\lambda, Q_\beta, Q_{\mathrm{dir}} = 0$
$\boldsymbol{x}^{\star} \leftarrow \arg\min_{\boldsymbol{x}_k} f(\boldsymbol{x}_k)$
for $t = 1$ to T **do**
  $\tau \leftarrow t/T$; $s \leftarrow \mathbb{I}[t \geq T/2]$ ▷ phase: early vs. late
  for each agent L **do**
    $\varepsilon \leftarrow \max(\varepsilon_{\min},\ \varepsilon \cdot 0.995)$
    Choose $a_\lambda \sim \varepsilon$-greedy$(Q_\lambda[s, \cdot])$, $a_\beta \sim \varepsilon$-greedy$(Q_\beta[s, \cdot])$, $a_{\mathrm{dir}} \sim \varepsilon$-greedy$(Q_{\mathrm{dir}}[s, \cdot])$
    Decode $(\lambda, \beta, \mathrm{dir}) \leftarrow (a_\lambda, a_\beta, a_{\mathrm{dir}})$
    $B \leftarrow$ Top-λ indices of $\boldsymbol{s}$
    Select partner $\boldsymbol{x}^{(p)}$ from strictly-better agents (fallback to self)
    Form Δ (attract to $\boldsymbol{x}^{\star}$ if dir $= 0$; repel from $\boldsymbol{x}^{(p)}$ if dir $= 1$)
    $\eta \leftarrow \tau + \beta \cdot (\boldsymbol{s}_B / \max_j s_j)$; $\boldsymbol{x}^{\mathrm{new}} \leftarrow \Pi_{[\boldsymbol{\ell}, \boldsymbol{u}]}(\boldsymbol{x} + \eta\, \Delta)$
    $f^{\mathrm{new}} \leftarrow f(\boldsymbol{x}^{\mathrm{new}})$
    if $f^{\mathrm{new}} < f(\boldsymbol{x})$ **then**
      $\boldsymbol{x} \leftarrow \boldsymbol{x}^{\mathrm{new}}$; $f(\boldsymbol{x}) \leftarrow f^{\mathrm{new}}$; $\boldsymbol{s}_B \leftarrow \boldsymbol{s}_B \cdot (1 + \mathrm{rr}/2)$
      if $f(\boldsymbol{x}) < f(\boldsymbol{x}^{\star})$ **then** $\boldsymbol{x}^{\star} \leftarrow \boldsymbol{x}$
      end if
    else
      $\boldsymbol{s}_B \leftarrow \boldsymbol{s}_B \cdot (1 - \mathrm{rr})$
    end if
    $r \leftarrow -(f^{\mathrm{new}} - f(\boldsymbol{x}))$; update $Q_\lambda, Q_\beta, Q_{\mathrm{dir}}$ with learning rate α and discount γ
    if std$(\boldsymbol{s}) \approx 0$ and $L \neq \boldsymbol{x}^{\star}$ **then** re-seed $(\boldsymbol{x}, \boldsymbol{s})$
  end for
  if $K \mid t$ **then**
    $\boldsymbol{x}^{\mathrm{loc}} \leftarrow \mathrm{LocalRefine}(f, \boldsymbol{x}^{\star})$; **if** $f(\boldsymbol{x}^{\mathrm{loc}}) < f(\boldsymbol{x}^{\star})$ **then** $\boldsymbol{x}^{\star} \leftarrow \boldsymbol{x}^{\mathrm{loc}}$
  end if
end for
return $\boldsymbol{x}^{\star}$

Results and Analysis

We compare **MARL-PRO** against the baseline **PRO** on four standard real-parameter test functions under identical box constraints and comparable evaluation budgets. Our aim is to assess:

- *Convergence speed* (best-so-far objective vs. evaluations),
- *Robustness across landscape types* (separable/nonseparable, conditioning),
- *Stability* (smoothness of descent, absence of collapse).

We use four well-known benchmarks:

Sphere: $f(x) = \sum_{i=1}^{d} x_i^2$ (convex, well-conditioned, separable)

Rosenbrock: $f(x) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$ (ill-conditioned, nonseparable)

Schwefel 1.2: $f(x) = \sum_{i=1}^{d} (\sum_{j=1}^{i} x_j)^2$ (strongly nonseparable, cumulative coupling)

Schwefel 2.22: $f(x) = \sum_{i=1}^{d} |x_i| + \prod_{i=1}^{d} |x_i|$ (nonseparable due to product term, valley to origin)

Table 1. Landscape characteristics relevant to search operators.

Function	Separable	Conditioning	Notes
Sphere	Yes	Easy	Smooth convex bowl
Rosenbrock	No	Hard	Narrow, curved valley
Schwefel 1.2	No	Hard	Strong coordinate coupling
Schwefel 2.22	No	Moderate	L_1 + product (ridge/valley)

Table 2. Final best objective at the evaluation budget, aggregated over R independent runs. Each function has two sub-rows: Mean and Std.

Function	Metric	PRO	MARL-PRO
Sphere	Mean	1.824 × 10e−11	2.495 × 10e−26
	Std	1.631 × 10e−11	4.589 × 10e−26
Rosenbrock	Mean	2.389 × 10e01	7.973 × 10e−11
	Std	2.448 × 10e01	1.681 × 10e−11
Schwefel 1.2	Mean	2.985 × 10e00	9.950 × 10e−01
	Std	1.216 × 10e00	5.138 × 10e00
Schwefel 2.22	Mean	1.827 × 10e01	7.973 × 10e−10
	Std	2.437 × 10e01	1.681 × 10e−10

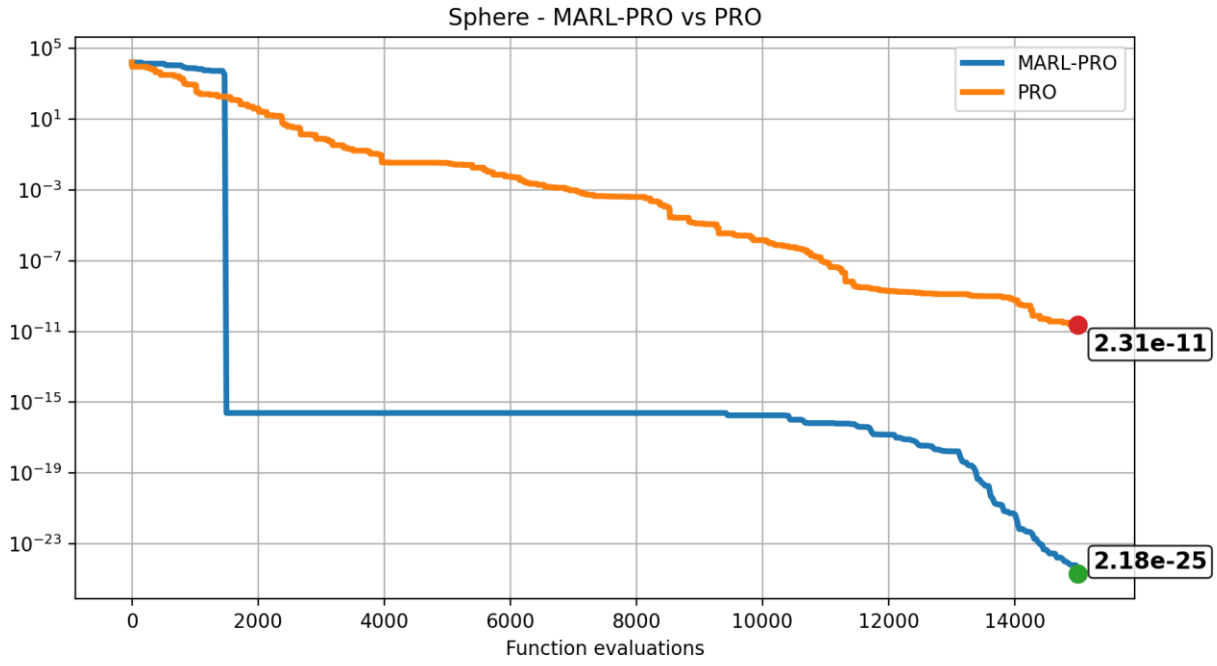

Figure 1: MARL-PRO vs. PRO on **Sphere**. Best-so-far objective vs. evaluations

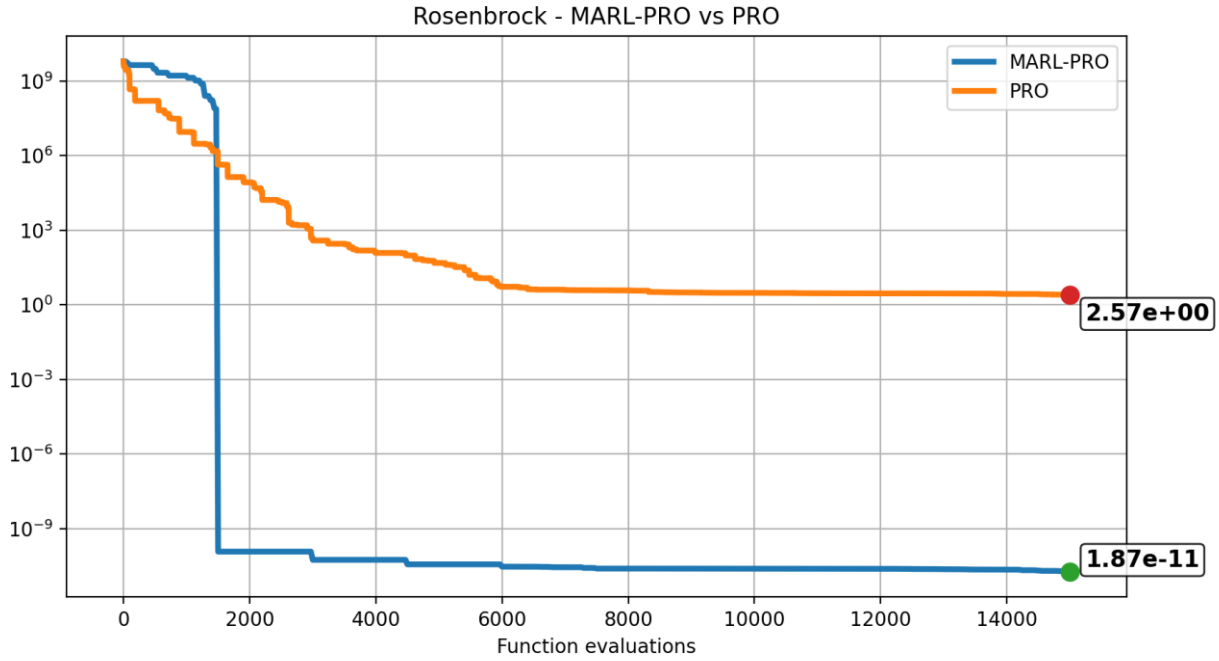

Figure 2: MARL-PRO vs. PRO on **Rosenbrock**. Best-so-far objective vs. evaluations

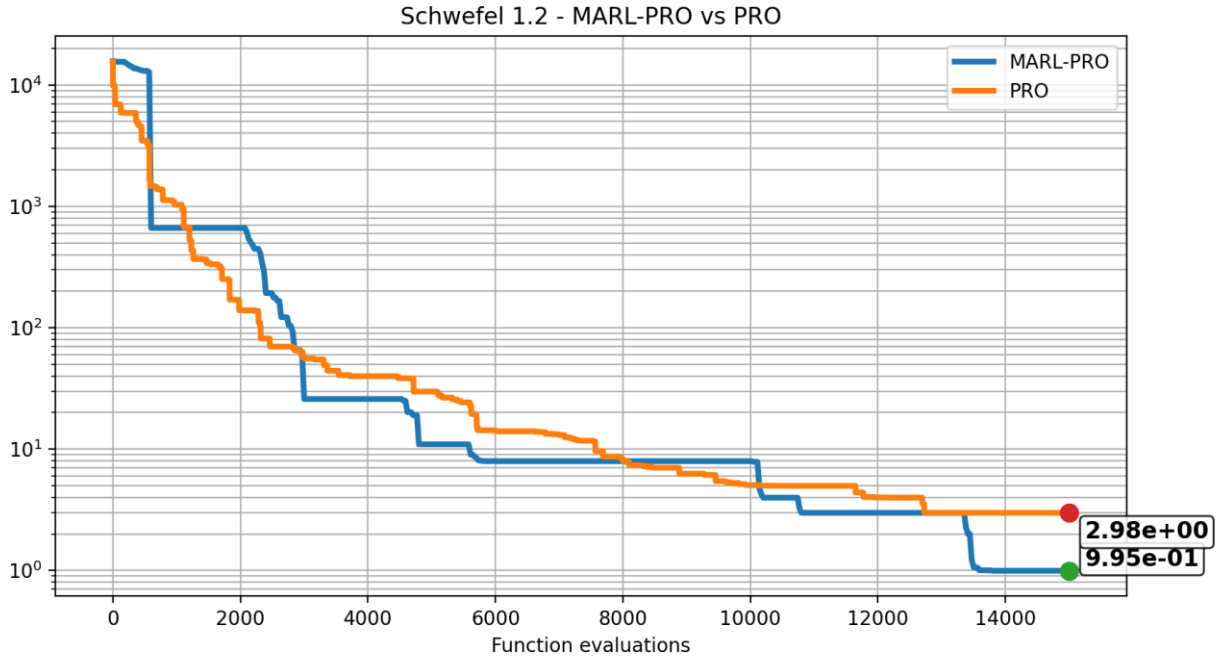

Figure 3: MARL-PRO vs. PRO on **Schwefel 1.2**. Best-so-far objective vs. evaluations

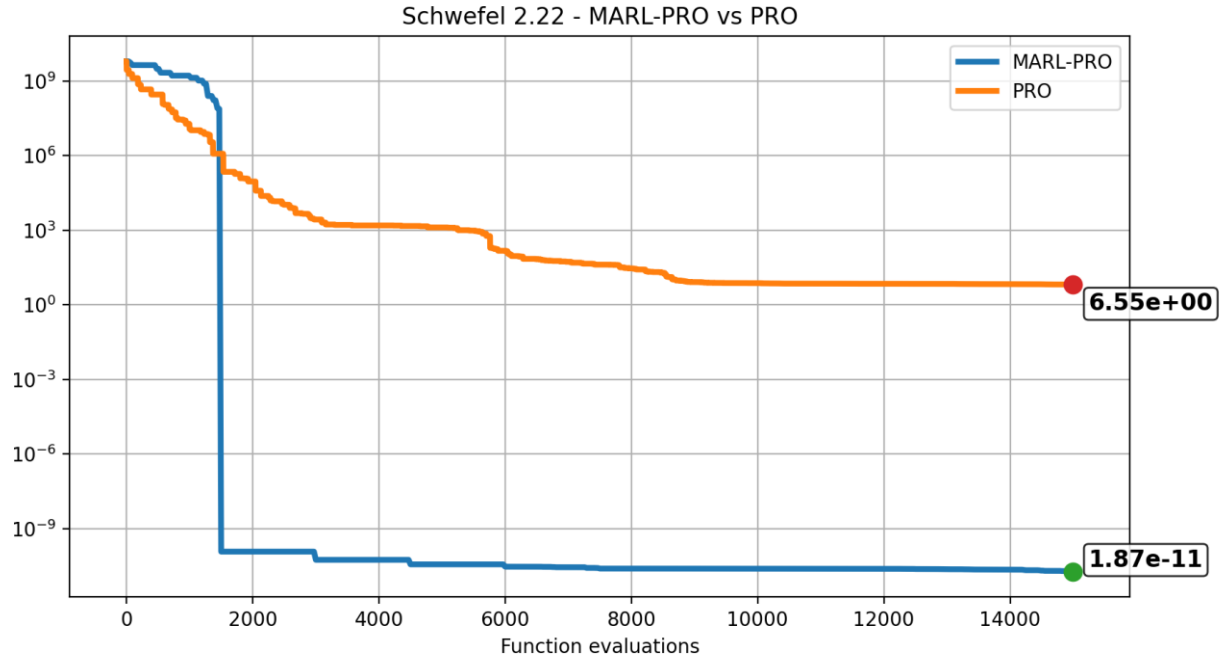

Figure 4: MARL-PRO vs. PRO on **Schwefel 2.22**. Best-so-far objective vs. evaluations

Overall Analysis and Advantages of MARL-PRO

Across all four landscapes, three patterns recur:

1. **Learned move selection (factorized Q).** By learning preferences over *subspace size* λ, *step scale* β, and *direction* (attract/repel), MARL-PRO tailors its operator to the local landscape. Small λ emerges late on ill-conditioned valleys (Rosenbrock) while larger λ is favored early on smooth bowls (Sphere).
2. **Sensitivity-guided partial updates.** The per-coordinate sensitivity vector concentrates effort on the most responsive coordinates, reducing waste in high dimensions and improving nonseparable cases (Schwefel 1.2).
3. **Population coordination with diversity.** Sharing the incumbent best x^* (attraction) aligns exploitation, while repulsion from a better peer $x^{(p)}$ injects directionally meaningful diversity—useful on ridged or deceptive regions (Schwefel 2.22) and for escaping poor basins.

Conclusion

We introduced **MARL-PRO**, a population-based optimizer that couples independent, value-based multi-agent reinforcement learning with a partial reinforcement operator. Each agent learns factorized, discrete preferences over subspace size, step scale, and attract/repel direction, while a sensitivity vector focuses updates on responsive coordinates and a simple two-phase schedule anneals exploration to exploitation. Periodic local polishing and a lightweight diversity reset complete the loop. This design keeps the control policy compact and interpretable while allowing the move operator to adapt to the local geometry of the objective.

Across four standard benchmarks under matched evaluation budgets, MARL-PRO demonstrates faster and more stable convergence than PRO, particularly on ill-conditioned and nonseparable landscapes (e.g., Rosenbrock and

Schwefel 2.22). The gains arise from (i) learned move selection that shrinks the active subspace late and moderates step scales near optima, (ii) sensitivity-guided partial updates that reduce wasted effort in high dimensions, and (iii) coordinated population signals (attraction to the incumbent best and repulsion from better peers) that balance exploitation with meaningful diversity.

References:

[1] Taheri, A.; RahimiZadeh, K.; Beheshti, A.; Baumbach, J.; *Partial Reinforcement Optimizer: An Evolutionary Optimization Algorithm*, Expert Systems with Applications, Elsevier, 2024.
[2] Holland, J.; *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge (MA), 2nd Edition, 1992.
[3] Kennedy, J.; Eberhart, R.; *Particle Swarm Optimization*, Proceedings of IEEE International Conference on Neural Networks, Volume IV, pp. 1942–1948, IEEE, Perth, 1995.

[4] Mirjalili, S.; Mirjalili, S. M.; Lewis, A.; *Grey Wolf Optimizer*, Advances in Engineering Software, Volume 69, pp. 46–61, Elsevier, 2014.

[5] Albrecht, S. V.; Christianos, F.; Schäfer, L.; *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*, MIT Press, Cambridge (MA), 2024.