PRODUCT-SET DEFICIENCY: A TRAINING-FREE MEASURE OF JOINT-ACTION COUPLING IN MULTI-ROBOT TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Cooperative multi-agent reinforcement learning often relies on value factorisation to enable decentralised execution, but this can restrict the combinations of actions that multiple robots can select jointly. We apply product-set deficiency, δ, the complement of a known bounding-box fill ratio, as a training-free measure of this restriction. Given the feasible joint actions for an instruction, obtained by enumerating joint actions in a contact simulator, δ measures how far this set deviates from the Cartesian product of the individually feasible actions; $\delta = 0$ when every combination of individually feasible actions is jointly feasible. We evaluate δ on a two-robot rod-pushing task. Instructions requiring coordination through the object have substantially higher δ than an instruction that can be solved independently. Although the absolute value of δ depends on action discretisation, the relative ordering of the instructions remains stable across $K \in [6, 24]$, and the same pattern holds for three and four robots pushing a box. We then test whether higher δ translates into greater learning difficulty. Across our experiments, it does not: δ does not predict the cost of factorised learning, and coordination graphs show no detectable learning advantage over independent learners despite substantially higher computational cost. An edge-gating variant guided by δ gains where $\delta \approx 0$ and beats both on one set of fresh seeds, but not on a second, and controls show the same gain without δ. These results suggest that δ is useful for characterising non-factorisable joint-action structure, but should not be interpreted as a direct measure of learning difficulty.

1 INTRODUCTION

Manipulating a shared object with multiple robots — for example, two robots pushing, carrying, or turning a load — is an important capability for warehouse, construction, and field robots. In these tasks, each robot must choose actions from its own observations, while the team must learn coordinated behaviour without explicitly enumerating the full joint action space, whose size grows rapidly with the number of robots. Cooperative multi-agent reinforcement learning (MARL) addresses this problem through several forms of value factorisation, in which the joint value is represented using per-robot utilities while retaining decentralised action selection. A key condition in many of these methods is the Individual-Global-Max (IGM) property: the optimal joint action can be obtained by combining the individually optimal actions. When this assumption is restrictive, actions that appear reasonable for the individual robots may fail when executed together. For example, two robots pushing a rod may each choose an action that moves the object in the desired direction, while their combined sideways motion causes the rod to rotate. This raises a more basic question: can we measure, directly from the task and before training, how strongly successful joint actions depend on combinations of individual actions?

Over the last decade, several families of factorised value methods have been developed. VDN (Sunehag et al., 2018) represents the joint value as a sum of per-robot utilities, while QMIX (Rashid et al., 2018) combines them through a monotonic mixing network. QTRAN (Son et al., 2019) explicitly formulates the IGM condition and relaxes some of the structural constraints of value factorisation. QPLEX (Wang et al., 2021) and Weighted QMIX (Rashid et al., 2020) further extend the representable class while retaining decentralised greedy action selection. Independent learners can also

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

suffer from relative overgeneralisation, in which actions that are safe against a randomly behaving partner may be preferred over actions that are optimal when the agents coordinate (Wei & Luke, 2016). These results establish that factorisation can impose representational constraints, but they do not by themselves provide a task-level measure of how strongly a particular physical task exhibits such constraints.

Coordination graphs (Guestrin et al., 2001) and their deep formulation, DCG (Böhmer et al., 2020), represent interactions between agents explicitly, through pairwise payoff terms that let coupled joint actions be represented. Sparse cooperative Q-learning (Kok & Vlassis, 2006) restricts these interactions to the edges of a sparse coordination graph, while CASEC (Wang et al., 2022) uses learned payoff information to determine which agent pairs should be connected. These approaches address which interactions should be represented by the learned model. In contrast, our focus is on measuring the structure of the feasible action set induced by the task itself, before a value function is learned. Castellini et al. (2019) study how different factorisation classes represent task classes, but do not provide the task-level geometric measure considered here.

Language has also been used to condition multi-robot coordination. ICCO (Yano et al., 2025) maps a team instruction to an instruction representation for individual robots, while Gupta et al. (2026) and Kuang et al. (2026) use language models to influence coordination-graph structure. SMART-LLM (Kannan et al., 2023) uses language to plan robot coalitions. These methods show that task instructions can influence coordination, but they do not directly measure the feasible combinations of robot actions induced by an instruction.

In this paper, we measure this structure without training. We enumerate joint actions on a discretised action grid in a contact simulator and evaluate each action against the success predicate associated with an instruction. From the resulting feasible joint-action set, we compute the product-set deficiency δ, the complement of its bounding-box fill ratio — a shape descriptor long established in image analysis (Rosin, 1999), applied here to joint action sets. Intuitively, δ measures how far the feasible joint-action set departs from a Cartesian product of individually feasible actions. A value of $\delta = 0$ indicates that every combination of individually feasible actions is jointly feasible. The measure is related formally to the set of joint actions that can be selected by a factorised greedy policy (Section 2.1, Proposition 1). Importantly, we do not assume that a larger δ necessarily implies greater learning difficulty. Instead, we experimentally test what information about learning, if any, can be obtained from the task-level measure.

The contributions of this paper are:

- A training-free, enumeration-based measure δ of the departure of a feasible joint-action set from a product set, together with a formal connection to the joint actions that can be selected by a factorised greedy policy (Proposition 1).

- An evaluation of δ on a two-robot contact task, showing that instructions requiring coordination through the object have substantially different product-set structure from an instruction that can be solved independently, with a ranking that is stable under action discretisation; the same pattern holds for three and four robots pushing a box, although pairwise δ does not reliably identify which pair must coordinate. We also compare δ with other training-free measures: it ranks tasks almost exactly as total correlation does, so its advantage is interpretability.

- A measurement of δ directly in the multi-step learning environment, together with an algorithm that uses the measure before training to determine whether a coordination-graph edge is retained (Algorithm 2); it gains on the instruction where $\delta \approx 0$ and beats the full coordination graph and the independent learner on one set of fresh seeds, but not on a second, and controls show the same gain without δ.

- Pre-registered experiments examining what δ implies for learning. These experiments test factorised learning, coordination graphs, and instruction-conditioned coordination graphs, including a fresh-seed replication and predefined evaluation criteria. δ does not predict the cost of factorised learning, and the coordination graph shows no detectable advantage over the independent learner.

2 METHODOLOGY

2.1 PROBLEM FORMULATION

Notation. Agents $N = \{1, \dots, n\}$ with discrete action sets $\mathcal{A}_i$; joint action $a \in \mathcal{A} = \prod_i \mathcal{A}_i$; world state s; instruction $L \in \mathcal{L}$. The joint action space is exponential in n: eight agents with six actions each already exceed a million joint actions.

The condition under test. Factorised methods require the Individual-Global-Max condition

$$\arg\max_a Q_{\text{tot}}(\tau, a) = \big(\arg\max_{a_i} q_i(a_i \mid \tau_i) \big)_{i=1}^{n}, \tag{1}$$

as formalised for QTRAN (Son et al., 2019) (τ is the joint action-observation history and τ_i robot i's), which permits each agent to act on its own observation at execution time. The alternative that keeps decentralised execution is a coordination graph (Guestrin et al., 2001; Böhmer et al., 2020), which factors the joint value into per-agent utilities and pairwise payoffs over an edge set E, averaged as in (Böhmer et al., 2020),

$$Q_{\text{tot}}(\tau, a) = \frac{1}{n} \sum_i q_i(a_i \mid \tau_i) + \frac{1}{|E|} \sum_{(i,j) \in E} q_{ij}(a_i, a_j \mid \tau), \tag{2}$$

maximised by message passing.

Definition 1 (Feasible set)

$$F(s, L) = \{\, a \in \mathcal{A} \,:\, \text{Pred}_L\big(\text{Rollout}(s, a)\big) \,\}, \tag{3}$$

where Pred_L is instruction L's success predicate evaluated on the simulated outcome of holding joint action a from state s (Section 3.1).

Definition 2 (Product-set deficiency)

$$\boxed{\delta(s, L) \;=\; 1 \;-\; \frac{\big|F(s, L)\big|}{\prod_{i=1}^{n} \big|\, \text{proj}_i F(s, L) \,\big|}} \tag{4}$$

with $\text{proj}_i F = \{a_i : \exists\, a_{-i},\ (a_i, a_{-i}) \in F\}$; δ is undefined when $F = \emptyset$, and such cases are reported rather than scored.

$\delta = 0$ if and only if F is a product set: every agent may choose freely within its own projection and the combination is always valid. The denominator is the smallest product set containing F, so δ is the fraction of that hull which F fails to fill.

Proposition 1 (What a factorised greedy policy can select) *For any per-agent utilities q_i, the greedy set $G = \arg\max_a \sum_i q_i(a_i)$ equals $\prod_i \arg\max_{a_i} q_i(a_i)$ and is therefore a product set. Consequently a factorised learner can have $G = F$ only if $\delta(F) = 0$. If $\delta(F) > 0$, there are actions $a_i \in \text{proj}_i F$, each individually part of some success, whose combination fails; agents that choose independently within their projections of F then miscoordinate with positive probability.*

Proof. The maximiser of a sum of functions of disjoint variables is the product of the per-variable maximisers. $F \subseteq \prod_i \text{proj}_i F$ always, with equality if and only if $\delta = 0$; if the inclusion is strict, some element of the product lies outside F. □

Proposition 1 does *not* say that IGM fails when $\delta > 0$: any single $a^* \in F$ is itself a product set and can be the unique greedy action of a factorised learner at any δ, and Section 3.3 shows independent learners finding one at $\delta = 0.84$. δ counts how many combinations of individually admissible actions fail; it is a property of the task, not a statement about learning difficulty.

For $n = 2$, $\delta \le 1 - 1/\min(|\text{proj}_0 F|, |\text{proj}_1 F|)$, and a set made of one full row and one full column has $\delta = 1 - (P_0 + P_1 - 1)/(P_0 P_1)$ (Proposition 2, Appendix B). With a single pairwise term, a coordination graph (2) can make its greedy set equal to any F for $n = 2$ (take q_{12} to be the indicator of F). This is why the pairwise payoff is the one term the variants below modify.

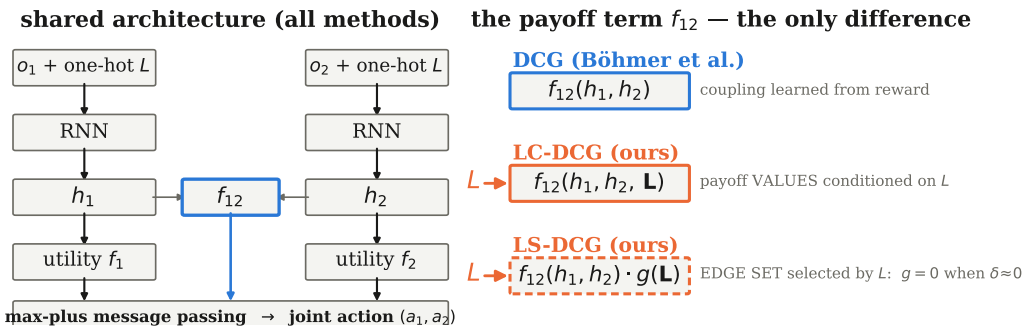

Figure 1: Architecture. Left: the pipeline shared by all learners. Right: the pairwise payoff f_{12}, the only term that differs between DCG, LC-DCG and LS-DCG.

2.2 Architecture

All learners in this paper share the pipeline on the left of Fig. 1, inherited unchanged from the reference DCG implementation (Böhmer et al., 2020). At each step, robot i receives its observation o_i^t with the instruction appended as a one-hot $e(L)$, together with its previous action and its identity, and encodes its history with a gated recurrent unit,

$$h_i^t = \mathrm{GRU}\big(\mathrm{ReLU}(W[o_i^t, e(L), a_i^{t-1}, e_i]),\ h_i^{t-1}\big), \tag{5}$$

where a_i^{t-1} is the previous action (one-hot) and e_i the robot's identity (one-hot). The hidden state feeds a per-robot utility and, for every edge of the coordination graph, a pairwise payoff,

$$q_i(a_i \mid \tau_i) = f_i(h_i^t), \qquad q_{ij}(a_i, a_j \mid \tau) = f_{ij}\big([h_i^t, h_j^t]\big), \tag{6}$$

where f_i is linear and f_{ij} is a two-layer network returning a $K \times K$ matrix, symmetrised by averaging the two orderings of its inputs. The joint value is (2); the greedy joint action is found by max-plus message passing over the graph, and every network is trained by temporal-difference learning on Q_{tot} against a target network. For $n = 2$ the graph has one edge, so DCG has exactly one term that a factorised learner lacks: the payoff. The two variants change that term only, in the two ways an instruction can enter it.

LC-DCG (language-conditioned) routes the instruction into the payoff's *values*,

$$q_{ij}(a_i, a_j \mid \tau, L) = f_{ij}\big([h_i^t, h_j^t, e(L)]\big), \tag{7}$$

so the coupling can be read off the instruction rather than inferred from reward through the recurrent state. It adds $|\mathcal{L}|$ input weights per hidden unit of f_{ij} and nothing else.

LS-DCG (language-selected) leaves the payoff's values alone and lets the instruction select the *edge set*,

$$\begin{aligned} Q_{\mathrm{tot}}(\tau, a) &= \frac{1}{n}\sum_i q_i(a_i \mid \tau_i) + \frac{g(L)}{|E|}\sum_{(i,j)\in E} q_{ij}(a_i, a_j \mid \tau), \\ g(L) &= \begin{cases} 0 & L \in \mathcal{U} \\ 1 & \text{otherwise,} \end{cases} \end{aligned} \tag{8}$$

where $\mathcal{U}$ is the set of instructions whose coupling is negligible; it is fixed before training, not learned. Algorithm 2 selects it by thresholding δ. In the experiments reported here, $\mathcal{U} = \{\texttt{independent}\}$ was specified by us when LS-DCG was pre-registered, on the basis of the one-step enumeration, before δ had been measured on the learning environment; that later measurement confirms it (Section 3.2). The learning environment's `independent` reward is a sum of per-robot terms by construction, so a near-zero δ there is expected. With $g = 0$ the model reduces to a sum of utilities, so LS-DCG switches between an additive learner and a full coordination graph per instruction, and adds no parameters. The gate reads the instruction as the arg-max of robot 1's one-hot (both robots receive the same one-hot, up to observation noise), so noise cannot partially open an edge.

Algorithm 1 MEASUREDEFICIENCY(s, $\mathcal{L}$, K)

Input: state s, instruction set $\mathcal{L}$, grid size K
Output: $\delta(s, L)$ for every $L \in \mathcal{L}$
$\Theta \leftarrow \{\, 2\pi k/K \;:\; k = 0, \ldots, K-1 \,\}$
for $(\theta_0, \theta_1) \in \Theta \times \Theta$ **do**
    $r[\theta_0, \theta_1] \leftarrow \textsc{Rollout}(s, \theta_0, \theta_1)$ ▷ one physics rollout
end for
for $L \in \mathcal{L}$ **do**
    $F \leftarrow \{\, (\theta_0, \theta_1) : \textsc{Pred}_L(r[\theta_0, \theta_1]) \,\}$ ▷ same rollouts, every L
    if $F = \emptyset$ **then**
        continue ▷ predicate admits nothing; report, do not score
    end if
    $P_0 \leftarrow |\{\theta_0 : \exists \theta_1,\ (\theta_0, \theta_1) \in F\}|$
    $P_1 \leftarrow |\{\theta_1 : \exists \theta_0,\ (\theta_0, \theta_1) \in F\}|$
    $\delta(s, L) \leftarrow 1 - |F| \,/\, (P_0 \cdot P_1)$
end for
return $\delta(s, \cdot)$

Algorithm 2 DELTAGATEDCG($\mathcal{L}$, $\mathcal{S}$, K, ε_δ)

Input: instructions $\mathcal{L}$, sampled states $\mathcal{S}$, grid size K, threshold ε_δ
Output: decentralised greedy policy
for $L \in \mathcal{L}$ **do** ▷ before training, no learning
    $\bar{\delta}(L) \leftarrow$ mean over $s \in \mathcal{S}$ of Alg. $1(s, \mathcal{L}, K)[L]$
end for
$\mathcal{U} \leftarrow \{\, L : \bar{\delta}(L) \le \varepsilon_\delta \,\}$ ▷ uncoupled instructions
initialise GRU, f_i, f_{ij} and their targets
for each episode, with instruction L **do**
    for $t = 0, \ldots, T-1$ **do**
        $h_i^t \leftarrow$ (5); $q_i, q_{ij} \leftarrow$ (6) for every robot and edge
        $g \leftarrow 0$ **if** $\arg\max e(L) \in \mathcal{U}$ **else** 1 ▷ gate of (8)
        $a^t \leftarrow \textsc{MaxPlus}(q_i,\ g\, q_{ij})$, ϵ-greedy; step; store transition
    end for
    TD update of Q_{tot} in (8) on a replay batch; periodic target update
end for

The DCG baseline uses the same two-layer payoff network as the variants, so the three differ only in how the instruction enters, not in capacity.

2.3 MEASURING δ

The procedure (Alg. 1; pipeline in Fig. A2, Appendix B) is: sample a state s; enumerate the K^n joint actions on a discretised grid; roll each out in the simulator; score each rollout against every instruction's success predicate to obtain $F(s, L)$; evaluate (4).

Two design choices matter. First, *one rollout set is shared across instructions*. Because the same K^n rollouts are scored under every predicate, any difference in δ between instructions is attributable to the instruction alone — the physics is literally identical. Second, no learning is involved, so δ characterises the task rather than any algorithm applied to it. The cost is K^n rollouts per state: at $K = 12$, $n = 2$, one state's 144 rollouts take about 1 s on one core of an AMD Ryzen 7 7840HS, but the count grows exponentially in the number of robots, so exhaustive enumeration is limited to small teams.

The only modelling choice is the success predicate; its thresholds are swept (Section 3.1.3).

Using the measurement. Algorithm 2 composes the two parts: Algorithm 1 is run once, before training, and LS-DCG drops the coordination-graph edge for the instructions whose δ falls below a threshold ε_δ. In our experiments this selection was made by hand and later confirmed by the measurement (Section 2.2).

3 RESULTS & DISCUSSION

Table 1: Summary of results. Each answer is tested against criteria fixed before the experiments.

question	evidence	answer
Q1. Is a physical task's joint-action structure non-factorisable, and is δ a sound measure of it?	discretisation sweep, learning environment, 3–4 robots, total correlation	**Yes**: large where robots are coupled, ranking stable; absolute value depends on K
Q2. Does δ predict learning difficulty?	two bandit tests; five published learners	**No**
Q3. Can δ guide a learner's coordination structure?	two variants, replication, two controls, per-instruction test	**No**: δ marks where removing the edge helps, but the edge is needed nowhere, so gating by δ gains nothing
Robustness	observation noise	no difference detected

Table 1 gives the answers; the setup follows, then one subsection per question.

3.1 EXPERIMENTAL SETUP

3.1.1 ENUMERATION ENVIRONMENT

Two pushers on planar slide joints sit at the two ends of a free rod on a plane in MuJoCo (Todorov et al., 2012). Each chooses one of $K = 12$ directions and pushes with $60\,\mathrm{N}$ for $1.2\,\mathrm{s}$; rod length, mass, friction and pusher placement are randomised per state (Appendix A). Four instructions are scored on the same rollouts. With Δx, $\|\Delta p\|$, $\Delta\psi$ the rod's forward displacement, planar displacement and yaw, and d_0, d_1 the pushers' displacements, their success predicates are: `push forward`, $\Delta x > 0.10 \wedge |\Delta\psi| < 0.25$; `move (any way)`, $\|\Delta p\| > 0.08$; `leave alone`, $\|\Delta p\| < 0.02 \wedge \min(d_0, d_1) > 0.10$; `each just move`, $\min(d_0, d_1) > 0.10$.

3.1.2 LEARNING ENVIRONMENT

Learning uses a multi-step variant, `RodPushMulti`, in which each robot chooses one of $K = 8$ directions at every step for 25 steps. The instruction is sampled uniformly per episode from {`push forward`, `turn`, `independent`} and appended to both robots' observations as a one-hot, so every method receives the same information. Both robots share a shaped team reward: `push forward` and `turn` reward progress on the rod and terminate on success, and `independent` rewards each pusher's distance from the start, a sum of per-robot terms (full definitions in Appendix A).

3.1.3 MEASUREMENT PROTOCOLS

δ is measured over 20 randomised states at $K = 12$, every instruction scored on the same 144 rollouts per state. Three properties are measured rather than assumed, none involving training: **grid dependence** ($K \in \{6, 8, 12, 16, 24\}$, 10 states), **threshold sensitivity** (15 predicate settings for `push forward` and 9 for `leave alone`, 8 states; Appendix G), and **environment dependence** (δ measured directly on the learning environment, holding a joint action for the whole episode, under the terminal condition and three return thresholds).

3.1.4 LEARNING COMPARISON PROTOCOL

We train published implementations on the learning environment: IQL, VDN, QMIX, QTRAN and DCG, all run from the reference DCG repository (Böhmer et al., 2020) with the shared settings each configuration ships with (Appendix A); **no algorithm was reimplemented**. Every metric and criterion in Table 2 was fixed before the runs it governs, and every learning number is computed from the raw per-evaluation records.

Table 2: Learning comparison protocol. All criteria were fixed before the runs they govern.

methods	IQL, VDN, QMIX, QTRAN, DCG; LC-DCG, LS-DCG
budget	300k environment steps; 20 evaluations of 30 greedy episodes
metric	test return at the final evaluation (≈285k steps)
seeds	0–7 for every method; 8–19 for the replication
LC-DCG	fails unless it beats DCG by >1 sd on final return *or* on steps to reach DCG's final return; framing fails if DCG does not beat IQL and QMIX
LS-DCG	H-1: fails unless it beats DCG by >1 sd on final return; H-3: fails if it shows no advantage on `independent`; H-2 (sample efficiency ≥ DCG) reported, not a failure condition
replication	LS-DCG vs DCG and vs IQL, seeds 8–19; supported iff both Welch $p < 0.05$ after Holm correction
controls	DCG with the edge removed for every instruction; LS-DCG gated on the wrong instruction
noise	Gaussian noise $\sigma \in \{0.05, 0.20\}$ on every observation entry incl. the one-hot; IQL, QMIX, DCG, LS-DCG; 3 seeds
compute	one run at a time on an idle machine, CPU and GPU, s per 1k steps

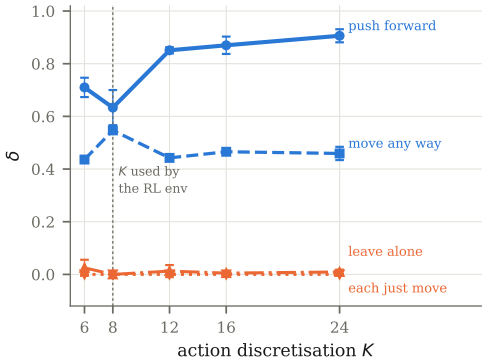

Figure 2: δ against action discretisation K; 10 states, error bars ± 1 population sd across states.

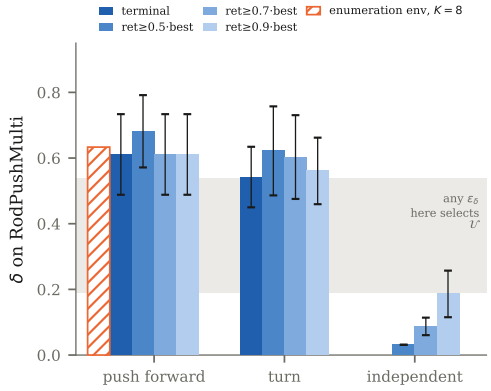

Figure 3: δ on the learning environment ($K = 8$; 12 states, 8 for `turn` terminal-based) under four success definitions. Hatched: enumeration environment, `push forward` only.

3.2 Q1: Is the structure non-factorisable, and is δ sound?

Yes, and the ranking, not the value, is what is robust. At $K = 12$, δ is 0.854 for `push forward` and 0.446 for `move any way`, and near zero for the two uncoupled instructions (Table 3); across instructions it spans 0.854 against a mean across-state standard deviation of 0.009, so it is set by the instruction, not the physical state. Its absolute value depends on the discretisation: for `push forward` it runs from 0.633 ($K = 8$) to 0.906 ($K = 24$), the spread across K is 3.4× the spread across states, and it tracks the ideal cross-shaped set $1-(2K-1)/K^2$ of Proposition 2 (Fig. 2). The ranking of instructions never changes over $K \in [6, 24]$. Three further checks agree. On the learning environment itself, δ is 0.54–0.68 for the coupled instructions and 0.03–0.19 for the uncoupled one under four success definitions (Fig. 3), so any threshold in $(0.19, 0.54)$ selects the same uncoupled set. On a box pushed by three or four robots it stays large (0.86–0.92 for `push forward`) with an identical ranking, although pairwise δ does not reliably identify the pair that must coordinate (Appendix D). And across all 230 sets it ranks tasks almost exactly as total correlation does ($\rho = 0.978$), so its advantage is interpretability (Proposition 1), not a different ordering (Appendix E).

Table 3: δ over 20 randomised states at $K = 12$ (population sd across states).

instruction	mean	std	min	max
push the rod forward	0.854	0.006	0.843	0.868
move the rod (any way)	0.446	0.022	0.438	0.542
leave the rod alone	0.003	0.006	0.000	0.016
each robot just move	0.000	0.000	0.000	0.000

Table 4: All learning results: test return at the final evaluation, sample sd, 95% t-interval, and Welch p against DCG on the same device and seeds.

	mean	sd	95% CI	p vs DCG
Published learners, seeds 0–7, GPU				
VDN (Sunehag et al., 2018)	10.73	4.90	[6.63, 14.83]	0.908
DCG (Böhmer et al., 2020)	10.47	3.85	[7.25, 13.69]	—
IQL	9.34	3.97	[6.03, 12.66]	0.574
QTRAN (Son et al., 2019)	7.51	1.78	[6.02, 9.00]	0.077
QMIX (Rashid et al., 2018)	4.97	2.37	[2.99, 6.95]	0.005
Variants and controls, seeds 0–7, CPU				
DCG	10.41	5.51	[5.81, 15.02]	—
LC-DCG (ours)	11.87	4.37	[8.22, 15.52]	0.568
LS-DCG (ours)	14.26	1.88	[12.69, 15.83]	0.096
DCG, edge removed everywhere	15.52	4.57	[11.70, 19.34]	0.064
LS-DCG, gate on wrong instruction	13.74	3.22	[11.05, 16.43]	0.167
Replication, seeds 8–19, CPU				
DCG	11.49	3.51	[9.26, 13.72]	—
LS-DCG (ours)	14.93	4.11	[12.31, 17.54]	0.039
IQL	8.40	2.72	[6.67, 10.14]	0.026

3.3 Q2: Does δ predict learning difficulty?

No. On the one-step bandit defined by each instruction's feasible set, independent Q-learners match a joint learner at every δ, with or without a pre-registered penalty for miscoordination (Appendix I). On the full task, the coordination graph, which can represent any feasible set, shows no detectable advantage over the independent learner, which represents no coupling at all: DCG − IQL is $+1.13$ (95% CI $[-3.07, 5.32]$; Table 4). Eight seeds detect only effects of about 1.5 standard deviations (≈ 5.9 return units), so we claim no equivalence. Of the published learners only QMIX is significantly worse, and QTRAN does no better. On an otherwise idle machine, one run at a time, the coordination graph costs $3.4\times$ the independent learner per environment step on CPU and $5.2\times$ on GPU; with these small networks the CPU is faster than the GPU for every method (Appendix M). Every method declines after an early peak (Fig. 4; Appendix J).

3.4 Q3: Can δ guide a learner's coordination structure?

No: the gain is real, but it does not depend on δ. Against their pre-registered bar (DCG plus one standard deviation, 14.33), LC-DCG fails and LS-DCG misses by 0.07 (Appendix K). On twelve fresh seeds LS-DCG then beats both DCG ($p = 0.039$) and IQL ($p = 0.0002$), both surviving Holm correction, so the pre-registered replication succeeds. But the controls match it: DCG with the edge removed for *every* instruction (15.52) and LS-DCG gated on the *wrong* instruction (13.74) do as well as LS-DCG ($p = 0.488, 0.705$). Removing the pairwise term tends to help wherever it is removed. A third pre-registered set of twelve seeds (20–31), logging returns per instruction, locates the gain: LS-DCG exceeds DCG on `independent` by 3.62 (95% CI $[1.52, 5.73]$, $p = 0.002$), where $\delta \approx 0$ and the edge is removed, and not on the coupled instructions ($p = 0.78, 0.92$), so the mechanism's pre-registered criterion H-3 is met. But on these seeds LS-DCG's overall return is not ahead of DCG's (14.29 against 13.37, $p = 0.692$), so the overall advantage of the replication did not recur. Together: δ correctly marks where removing the edge helps, but on this task the edge is not needed anywhere, so gating by δ gains nothing over removing it everywhere. The edgeless DCG has

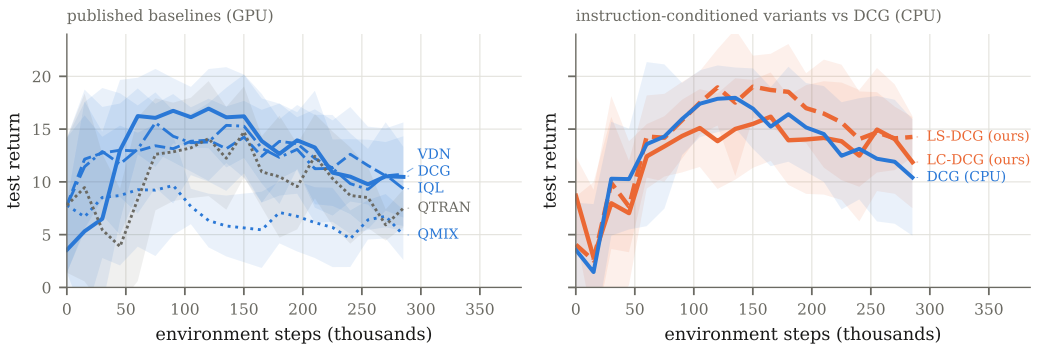

Figure 4: Test return during training, 8 seeds, ± 1 sample sd. Left: published baselines (GPU). Right: the variants and DCG (CPU).

VDN's function class yet exceeds VDN by 4.78 ($p = 0.063$), so training settings, not coordination structure, appear to drive much of the gain.

3.5 Robustness to observation noise

Under Gaussian noise on every observation entry, coupling-aware learners do not retain more return than the independent learner at three seeds, so the pre-registered inference is not supported (Appendix L).

4 Limitations

δ is defined on a discretised grid, its value depends on the resolution and on the shape of F, and exhaustive enumeration scales as K^n; only the ranking of instructions is claimed, from hand-written predicates on a rod (two robots) and a box (three and four), and all learning experiments use two robots. The learning comparison uses shipped hyperparameters and eight seeds, and the edge-gating variant's uncoupled set was specified by us and later confirmed by the measurement. The full list is in Appendix N.

5 Conclusion

We measured, by enumeration and without training, how far the feasible joint action sets of a physical multi-robot task depart from product sets. The departure is large where the robots are coupled through the object and small where they are not; its size depends on the discretisation, but its ranking of instructions is stable, holds with three and four robots, and matches that of total correlation. Its consequences for learning did not hold up: it does not predict the cost of factorised learning, a coordination graph shows no detectable advantage over an independent learner, and an edge-gating variant gains exactly where $\delta \approx 0$ but, because the pairwise term is needed nowhere on this task, no more than removing it everywhere, and not consistently across seed sets. δ is therefore a description of non-factorisable joint-action structure, not a guide to learning. Whether that changes when miscoordination carries a real cost, or with more robots and learned pairwise structure, is the question this work leaves open.

AI use statement

Generative AI (Anthropic Claude, via Claude Code) was used for manuscript formatting and to write the scripts that produce the figures and diagrams. All content was reviewed by the authors; the research direction, design decisions and conclusions are the authors' own.

Reproducibility statement

The measurement procedure is Algorithm 1; both environments, the training settings and the code changes are in Appendix A; the evaluation protocol, including every pre-registered metric and criterion, is in Section 3.1.4 and Table 2. All learners are the reference implementations, with two dependency changes and one changed setting (the payoff network's hidden layer), all listed in Appendix A. Every reported number is computed from the raw per-evaluation records.

References

Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *International Conference on Machine Learning (ICML)*, 2020.

Jacopo Castellini, Frans A. Oliehoek, Rahul Savani, and Shimon Whiteson. The representational capacity of action-value networks for multi-agent reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2019.

Bo Fu, William Smith, Denise Rizzo, Matthew Castanier, Maani Ghaffari, and Kira Barton. Learning task requirements and agent capabilities for multi-agent task allocation. *arXiv preprint arXiv:2211.03286*, 2022.

Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems 14 (NIPS)*, 2001.

Nikunj Gupta, Rajgopal Kannan, and Viktor Prasanna. Do LLM-derived graph priors improve multi-agent coordination? *arXiv preprint arXiv:2604.17191*, 2026.

Shyam Sundar Kannan, Vishnunandan L. N. Venkatesh, and Byung-Cheol Min. SMART-LLM: Smart multi-agent robot task planning using large language models. *arXiv preprint arXiv:2309.10062*, 2023.

Jelle R. Kok and Nikos Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7:1789–1828, 2006.

Zihao Kuang, Zengmin Xu, Linkai Wan, and Chunjiong Zhang. LLM-guided graph neural coordination framework for cooperative multi-agent reinforcement learning. *Complex & Intelligent Systems*, 2026. doi: 10.1007/s40747-026-02356-7.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *NeurIPS Datasets and Benchmarks Track*, 2021.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Paul L. Rosin. Measuring rectangularity. *Machine Vision and Applications*, 11:191–196, 1999.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 5887–5896, 2019.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033, 2012.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. QPLEX: Duplex dueling multi-agent Q-learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Tonghan Wang, Liang Zeng, Weijun Dong, Qianlan Yang, Yang Yu, and Chongjie Zhang. Context-aware sparse deep coordination graphs. In *International Conference on Learning Representations (ICLR)*, 2022.

Ermo Wei and Sean Luke. Lenient learning in independent-learner stochastic cooperative games. *Journal of Machine Learning Research*, 17(84):1–42, 2016.

Yoshiki Yano, Kazuki Shibata, Maarten Kokshoorn, and Takamitsu Matsubara. ICCO: Learning an instruction-conditioned coordinator for language-guided task-aligned multi-robot control. *arXiv preprint arXiv:2503.12122*, 2025.

A IMPLEMENTATION DETAILS

Learning-environment rewards. Both robots receive the same team reward per step, with Δx the rod's forward progress in the step, $|\psi|$ its absolute yaw and p_i pusher i's position: `push forward`, $r = 10\,\Delta x - 0.5\max(0, |\psi| - 0.35)$, with -2 while $|\psi| > 1.05$ and $+10$ and termination when $x \geq 1.5$ m with $|\psi| < 0.35$; `turn`, $r = 8\,\Delta|\psi| - 2|\Delta x|$, with $+10$ and termination when $|\psi| \geq 1.2$; `independent`, $r = 0.1(\|p_0\| + \|p_1\|)$, a sum of per-robot terms, with no termination. That the three instructions demand different joint actions is verified in Section C. `push forward` is the only instruction present in both environments, and even it is scored by a different predicate in each.

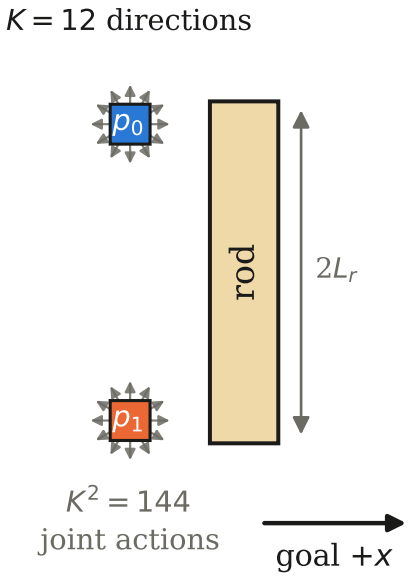

Figure A1: Enumeration environment: two pushers at the ends of a free rod, $K = 12$ directions each.

Table A1: Enumeration environment (on which δ is measured) and compute.

parameter	value
simulator	MuJoCo 3.13 (Todorov et al., 2012)
timestep / integrator	2 ms, `implicitfast`
rod half-length L_r	$\mathcal{U}(0.22,\ 0.40)$ m
rod mass	$\mathcal{U}(1.2,\ 3.5)$ kg
floor friction μ	$\mathcal{U}(0.4,\ 0.8)$
pusher stand-off	$\mathcal{U}(0.11,\ 0.16)$ m
pusher position along rod	$\mathcal{U}(0.85,\ 0.98) \cdot L_r$
action set	$K = 12$ directions, 60 N, 1.2 s
settle before push	0.2 s
states	20 randomised
GPU / CPU	RTX 4060 laptop (8 GB) / Ryzen 7 7840HS
software	PyTorch 2.13, NumPy 1.26, SciPy 1.15

Two modifications to the released code were required, besides the setting marked in Table A3. EPyMARL imports `smaclite` unconditionally and it is not on PyPI; that import is wrapped in

Table A2: Learning environment `RodPushMulti` (all learning results).

parameter	value
timestep / integrator	4 ms, `implicitfast`
action set	$K = 8$ directions, 30 N held 0.08 s per step
horizon	25 steps (settle 0.15 s first)
rod half-length, mass, μ	$\mathcal{U}(0.24, 0.36)$ m, $\mathcal{U}(1.4, 3.0)$ kg, $\mathcal{U}(0.45, 0.75)$
pushers	fixed at $x = -0.13$ m, $y = \pm 0.92\, L_r$
observation (per robot)	rod $(x, y, \cos\psi, \sin\psi)$, own and partner (x, y), $x_g - x$ with goal $x_g = 1.5$ m, and the instruction one-hot

Table A3: Learning comparison settings, as shipped with the reference code except where marked *.

setting	value
environment steps / horizon	300k / 25 steps
evaluation	30 greedy episodes every 15k steps
exploration	ϵ-greedy, $1.0 \to 0.05$ over 50k steps
discount γ	0.99
optimiser	RMSProp, lr 5×10^{-4}, $\alpha = 0.99$, clip 10
replay buffer	5000 episodes (IQL, VDN, QMIX); 500 (DCG family)
batch	32 episodes
target network	updated every 200 episodes
recurrent encoder	GRU, 64 units
utility / payoff nets	linear / one hidden layer of 64* (DCG family)
QMIX mixer	embed 32, 2-layer hypernetwork of 64
message passing	max-plus, 8 iterations, normalised, anytime
seeds	0–7; replication 8–19; noise sweep 0–2
device	GPU (baselines), CPU (variants, controls, replication)

`try/except`. The DCG code imports `torch_scatter`, for which no wheel exists for our PyTorch build and which fails to compile; it uses one function, `scatter_add`, at three call sites with a one-dimensional index, supplied here from native PyTorch and verified numerically identical to an explicit loop. Neither change touches any algorithm.

After the first three-seed runs we found and fixed four errors in *our own* environment wrapper and configuration; every learning result in this paper was obtained after the fixes: (a) truncation at the 25-step horizon had been signalled to the learner as a terminal state, cutting the bootstrap; (b) the reward had been scalarised differently in the two frameworks first used; (c) the pairwise payoff network had no hidden layer, so an instruction-conditioned payoff could add only a constant matrix per instruction — it now has one, in DCG and both variants alike; (d) observation noise had been applied to the physical observation but not to the appended one-hot. All three-seed learning numbers are void and none is reported.

B Range of δ and pipeline

Proposition 2 (Range) *For $n = 2$, with $P_i = |\mathrm{proj}_i F|$, $0 \le \delta \le 1 - 1/\min(P_0, P_1)$, the upper bound being attained by a set with one element per row or column. A set made of one full row and one full column of the $P_0 \times P_1$ hull has $\delta = 1 - (P_0 + P_1 - 1)/(P_0 P_1)$.*

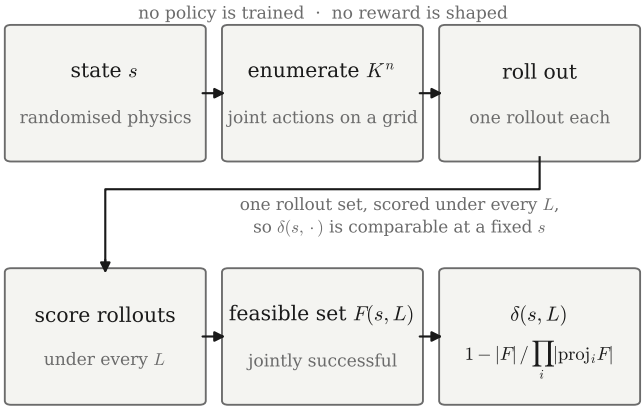

Figure A2: Measurement pipeline (Algorithm 1).

Table A4: Cross-evaluation of each instruction's best constant joint action on `RodPushMulti` (mean episode return over three fixed states).

	push fwd	turn	independent
push fwd	**25.51**	−11.55	12.61
turn	−25.33	**10.93**	16.50
independent	−0.00	−0.00	**21.83**

Proof. Every element of the larger projection needs a distinct element of F, so $|F| \geq \max(P_0, P_1)$; substitute into (4). The cross has $P_0 + P_1 - 1$ elements. □

C Instruction cross-evaluation

Before the learning comparison, we verified that the three instructions of Section 3.1.2 demand genuinely different couplings, by cross-evaluating each instruction's best fixed joint action under every instruction, averaged over three fixed states (environment seeds 3, 4, 5; rows: optimised for; columns: evaluated under; Table A4). The diagonal dominates every column, so the instruction is genuinely required. The penalty for using the wrong instruction's action is not uniform: it is severe for the two object-coupling instructions, where the off-diagonal entries go strongly negative (-25.33, -11.55), and mild for `independent`, where they fall only 24–42% below the diagonal. Two earlier candidate instruction sets failed this check and were discarded: one pair shared an identical optimum, and one instruction was geometrically unreachable because both pushers sit on the same side of the rod.

D Three and four robots

A second scene tests whether the measurement survives more robots: a free box with three or four pushers, two behind it and the others at its sides, with the same kind of predicates plus a partial-coupling instruction in which only robots 0 and 1 must push while the others keep clear. The criteria were fixed before any rollout, and δ was also computed for every pair of robots from the pair's projection of F (Table A5). **The measurement survives**: δ stays large for the coupled instructions (0.86–0.92 for `push forward`, 0.37–0.52 for `rotate`), and the ranking of instructions is identical with three and four robots. **Two expectations fail.** `each just move`, a product set by construction with two robots, is not exactly one with more (up to 0.014 in a state), because pushers can block each other through the box. And pairwise δ does not reliably single out the pair that must coordinate: robots 0 and 1 have the largest pairwise value in every state at only one of the four settings. That test was weaker than intended, because the partial-coupling instruction admitted the same joint actions as `push forward` in 39 of 40 states, so the others' keeping clear never bound. A further observation bears on coordination graphs directly: for `rotate` pairwise δ is essentially

Table A5: δ on a box pushed by n robots; 10 states per row. Last column: pairwise δ of pair (0,1) against the largest other pair.

n	K	push fwd	rotate	leave alone	each move	δ_{01} / max other
3	8	0.859	0.521	0.004	0.001	0.42 / 0.60
3	12	0.902	0.467	0.065	0.001	0.73 / 0.38
4	8	0.915	0.484	0.036	0.002	0.44 / 0.35
4	12	0.921	0.373	0.193	0.001	0.24 / 0.21

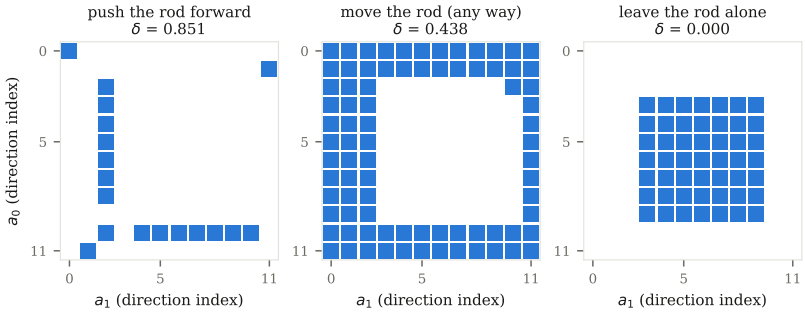

Figure A3: Feasible joint action sets for three instructions at one physical state; filled cells succeed.

zero while the full δ is about 0.5, and the pairwise mutual informations of Appendix E account for only 37–50% of the total dependence, against 88–104% for `push forward`. Much of the coupling in `rotate` therefore involves three or more robots at once, which pairwise edges cannot represent.

E Other training-free measures

On the same feasible sets (the 2-robot sets of Table 3 and the 3- and 4-robot sets at $K = 8$ and, for three robots, $K = 12$; 230 sets in all) we computed three alternatives: constraint tightness $|F|/K^n$; the total correlation of the uniform distribution over F, $\sum_i H(A_i) - \log|F|$, which is zero exactly when F is a product set; and, with two robots, a greedy upper bound on the number of rectangles needed to cover F. δ **ranks the sets almost exactly as total correlation does** (Spearman $\rho = 0.978$), so by the criterion fixed in advance δ's advantage over it is interpretability — its exact relation to what a factorised greedy policy can select (Proposition 1) — not a different ranking. Tightness is a different quantity ($\rho = -0.76$): the product-set control has tightness near 1 and `push forward` about 0.05–0.13. The rectangle bound separates the thin cross of `push forward` (5.2 rectangles on average) from the ring of `move any way` (2.5), which a fill ratio cannot. As pair selectors, pairwise mutual information picks out the pair behind the box in 50% of states at the settings where pairwise δ manages 20–40%, and both reach 100% with three robots at $K = 12$; neither meets the pre-registered 80% bar elsewhere.

F Geometry of the feasible sets

Fig. A3 shows why the numbers take the values they do.

For `push the rod forward`, F is a sparse *cross*: a union of two axis-aligned slices plus three isolated cells. Either p_1 commits to direction 2 while p_0 has latitude, or p_0 commits to direction 10 while p_1 does — a leader–follower structure in which one agent's choice is pinned and the other's is then constrained. This is a one-dimensional subset of a two-dimensional space, and it is not expressible as any product.

For `move the rod (any way)`, F is a *ring*: the four outermost direction bands always succeed, and interior directions succeed only if the partner drives into the rod. This is a disjunction, monotone but not rectangular.

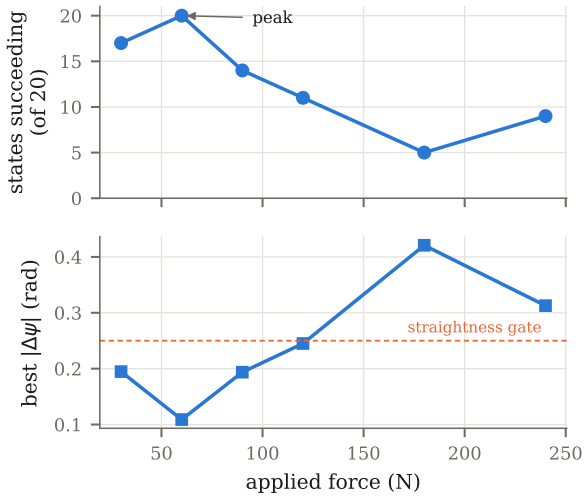

Figure A4: Single pusher on `push the rod forward`, second pusher removed: success over 20 states (top) and best achievable yaw (bottom) against force.

For `leave the rod alone`, F is a clean 7×7 rectangle. The control is non-trivial: five of twelve directions fail individually, yet all 49 combinations of the survivors succeed. δ is therefore not merely counting how many actions work.

G Threshold sensitivity

The success predicates are ours, so their thresholds are swept, over 8 randomised states at $K = 12$. For `push the rod forward`, $\Delta x \in [0.06, 0.15]$ crossed with $|\Delta\psi| \in [0.15, 0.40]$ gives 15 settings and $\delta \in [0.802, 0.852]$, a span of 0.051; for `leave the rod alone`, $\|\Delta p\| \in [0.01, 0.04]$ crossed with $\min(d_0, d_1) \in [0.06, 0.15]$ gives 9 settings and $\delta \in [0.000, 0.006]$. No setting produced an empty success set. The sweep covers two of the four instructions; `move any way` has a single threshold and `each just move` is a product set by construction.

The separation between instructions (≈ 0.85) is therefore roughly $17\times$ the threshold span. But this is the *less* exposed assumption: the discretisation K moves δ by 0.273 (Sec. 3.2), $5.4\times$ more than the thresholds do. We report the threshold sweep for completeness and rest no claim on it.

H Single-robot force sweep

Fig. A4 sweeps one pusher's force. The second pusher's body is deleted from the model, not merely left unpowered: an unpowered 1.5 kg damped box beside the rod acts as a passive brace, so that test would not have been a single-agent test.

Success is non-monotone in applied force — $17/20$ states at 30 N, $20/20$ at 60 N, then $14/20$, $11/20$, $5/20$ at 180 N and $9/20$ at 240 N. The mechanism is visible in the yaw: torque about the rod's centre grows with applied force, and past roughly 90 N the rod turns beyond the straightness gate. We state the yaw relationship carefully, because it is *not* monotone either ($0.195, 0.109, 0.194, 0.245, 0.421, 0.313$ rad): it is lowest where success peaks and rises through the interior, but falls at both ends of the sweep.

This is a counterexample to "more capability is never worse" on a continuous physical parameter. We do **not** claim it falsifies the capability models of (Fu et al., 2022) or (Kannan et al., 2023): those are defined over *team composition*, and say nothing about a single actuator's force magnitude. It bears on additive capability models only if one extends them to continuous parameters, which their authors do not.

I δ AND THE COST OF FACTORISED LEARNING

We pre-registered the hypothesis that δ predicts the penalty for factorised learning, with the kill condition that independent learners matching a joint learner would retire it. On the one-step bandit defined by each instruction's F (reward 1 inside F, 0 outside), independent Q-learners reach greedy success 1.000 at every δ, matching the joint learner exactly.

Relative overgeneralisation (Wei & Luke, 2016) needs miscoordination to be worse than a safe alternative, and a 0 reward for failure makes it free. We therefore pre-registered a second test in which failure costs $-c$, $c \in \{0.5, 1, 2\}$, with the criterion that the gap between joint and independent learners correlate positively with δ (Spearman, $p < 0.05$) and reach 0.10 on `push forward`. **It is not supported**: at every c both learners reach 1.000 on all 32 (state, instruction) sets, so the gap is identically zero. The mechanism is the same in both tests. Once one agent is greedy on a_1^*, its partner's value acquires a clean peak at a matching action, since every cell of F is a Nash equilibrium of the induced game and self-play finds one. A uniform penalty lowers every failing combination equally and creates no safe suboptimal action. **We therefore withdraw any claim that δ is an a-priori diagnostic of learning difficulty**; it characterises what a factorised greedy policy can select (Proposition 1), not how hard it is to find one success.

J PEAK RETURNS AND LATE DECLINE

Every method declines toward the final evaluation after an early peak, losing roughly half of its peak return: for the four baselines the seed-averaged curves in Fig. 4 crest between 75k and 150k and per-seed peaks average 53–107k, while the variants and the CPU re-run of DCG peak later (141–156k). Episode length rises back toward the horizon as return falls. We had attributed this decline to bootstrapping through time-limit truncation, fix (a). A pre-registered diagnostic does not bear that out: IQL with the fix reverted still loses 6.20 return units from its peak, against 10.32 with the fix, above the pre-registered bar of half, so the attribution is withdrawn and the decline is unexplained. The pre-registered final evaluation therefore compares methods partly by how late they decline. Two post-hoc secondary metrics give the same picture among the baselines: the mean of the last five evaluations (DCG 10.43, VDN 11.36, IQL 10.08, QMIX 5.68) and the normalised area under the curve (DCG 12.72, VDN 12.63, IQL 12.45, QMIX 6.99). At peak, too, DCG (19.49) and IQL (19.66) are not distinguishable ($p = 0.881$). Peaks are recorded below and are not a result.

J.1 PEAK VERSUS FINAL EVALUATION

For each seed the peak is the maximum raw test return over the twenty evaluation points (every 15k steps, 30 greedy test episodes each); the step of peak is the argmax position of that seed. All values are computed from the raw per-evaluation records, not from the console log, which prints a five-evaluation moving average. Sample standard deviations across the eight seeds.

Table A6: Final-evaluation and peak test return, `RodPushMulti`, 8 seeds.

method	final	sd	peak	sd	step of peak	ep. len at end
DCG	10.47	3.85	19.49	2.67	90k	23.8
VDN	10.73	4.90	17.72	3.18	98k	23.3
IQL	9.34	3.97	19.66	1.68	107k	23.8
QMIX	4.97	2.37	14.81	4.15	53k	25.0

Two statistics are easy to conflate. The *seed-averaged* curve in the Fig. 4 crests between 75k and 150k, whereas the *per-seed* argmax positions above average 53–107k: seeds peak at different times, and averaging the curves smears the crest later. Both are stated in Section 3.3.

J.2 WELCH TESTS ON THE PEAK (POST HOC)

DCG vs IQL: -0.17, $p = 0.881$. DCG vs VDN: $+1.77$, $p = 0.249$. DCG vs QMIX: $+4.68$, $p = 0.020$. The ordering of peaks (IQL $\approx$ DCG $>$ VDN $>$ QMIX) differs slightly from the ordering at the final evaluation (VDN $\approx$ DCG $\approx$ IQL $>$ QMIX), but the one comparison the paper turns on

— coordination graph against independent learner — is a null at the peak as it is at the endpoint. Had we switched to the peak after seeing it, the conclusion would not have changed; we did not switch because the protocol forbids choosing the metric after the data.

J.3 Cause of the decline

Episode length rises back toward the 25-step horizon as return falls. The earlier, void runs — which signalled time-limit truncation to the learner as a terminal state — improved monotonically. The corrected treatment (bootstrapping through truncation, Appendix A, fix (a)) is the standard one, but it allows value overestimation at the horizon state to propagate backward once ϵ reaches its floor of 0.05 at 50k steps. This attribution is a diagnosis, not a measured result: the single-seed ablation that would confirm it (reverting the fix for one IQL seed) was not run, because the fix will not be reverted in any case.

K Variant criteria in full

Device. After a driver fault the variants were trained on CPU while the baselines had been trained on GPU. The algorithm and code are identical, so as a control DCG was re-run on CPU at the same eight seeds: 10.41 against 10.47 on GPU ($p = 0.981$). The device does not move the result.

Every pre-registered criterion (Section 3.1.4) is reported here, on seeds 0–7. The final-return bar for both variants is DCG's mean plus one sample standard deviation, $10.47 + 3.85 = 14.33$. *LC-DCG* reaches 11.87 ± 4.37, short of the bar and not distinguishable from DCG ($+1.40$, 95% CI $[-3.03, 5.82]$). On sample efficiency it reaches DCG's final return after 28 ± 25k steps against DCG's 34 ± 24k, not faster by one standard deviation. It fails both criteria. Its framing criterion also fails, since DCG does not beat IQL (Section 3.3). *LS-DCG* reaches 14.26 ± 1.88 and misses the bar (H-1) by 0.07. It needs 39 ± 20k steps, so H-2 (sample efficiency at least DCG's) is not met either, although H-2 is not a failure condition. H-3 could not be evaluated on these seeds, because per-instruction returns were not logged; see Section 3.4.

LS-DCG's miss is narrow, and it has the highest mean and the smallest spread of any method in Table 4. Post hoc and uncorrected, it exceeds IQL by 4.91 ($[1.45, 8.37]$, $p = 0.010$) and DCG by 3.78 ($[0.41, 7.15]$, $p = 0.031$ against the GPU runs; $p = 0.096$ against the same-device CPU runs). Having seen this, we pre-registered a confirmatory replication on twelve fresh seeds and two controls before running them (Section 3.1.4). **The replication supports LS-DCG.** On seeds 8–19, all three arms on CPU, LS-DCG reaches 14.93 ± 4.11, DCG 11.49 ± 3.51 and IQL 8.40 ± 2.72. LS-DCG exceeds DCG by 3.44 (Welch $p = 0.039$) and IQL by 6.52 ($p = 0.0002$); both survive Holm correction over the two comparisons ($p = 0.039$ and 0.0004), so the pre-registered criterion is met. The margin over DCG is modest and close to the threshold, and it is one task.

The controls do not support the stated mechanism. The two pre-registered controls, on seeds 0–7 on CPU, ask whether the gain comes from removing the edge *where* $\delta \approx 0$. It does not. DCG with the edge removed for *every* instruction reaches 15.52 ± 4.57, and LS-DCG gated on the *wrong* instruction (`push forward`) reaches 13.74 ± 3.22; LS-DCG is not distinguishable from either (-1.26, $[-5.19, 2.67]$, $p = 0.488$; $+0.51$, $[-2.38, 3.40]$, $p = 0.705$). By the pre-registered rule we therefore make no claim that the measurement guides the gate. The edgeless DCG is, as a function class, the same average of per-robot utilities as VDN, yet it exceeds VDN by 4.78 ($[-0.30, 9.87]$, $p = 0.063$) and IQL by 6.17 ($[1.57, 10.77]$, $p = 0.012$). It differs from them in training settings rather than in coordination structure: a replay buffer of 500 episodes instead of 5000, DCG's learner, and no double Q-learning. The most plausible reading is that on this task the pairwise payoff term hurts learning, and that the DCG family's training settings help it. Neither effect involves δ.

L Observation noise

The pre-registered inference was that coupling-aware learners retain more return under noise than the independent learner, tested at $\sigma = 0.20$ (Table A7). **It is not supported.** At $\sigma = 0.20$ DCG is 5.27 below IQL ($p = 0.096$), and LS-DCG is level with it ($+0.15$, $p = 0.960$). No method degrades consistently with noise: IQL's mean rises from 9.34 clean to 13.22 at $\sigma = 0.20$, and only DCG falls

Table A7: Final-evaluation test return under observation noise σ; 3 seeds (clean: 8). Δ: change from clean.

method	clean	$\sigma = 0.05$	Δ	$\sigma = 0.20$	Δ
DCG (Böhmer et al., 2020)	10.47	11.70 ± 2.51	$+1.23$	7.96 ± 2.46	-2.52
LS-DCG (ours)	14.26	13.24 ± 1.57	-1.01	13.37 ± 3.54	-0.89
IQL	9.34	10.31 ± 1.18	$+0.96$	13.22 ± 3.29	$+3.88$
QMIX (Rashid et al., 2018)	4.97	7.00 ± 1.52	$+2.03$	6.12 ± 0.49	$+1.15$

at the larger σ. With three seeds against a between-seed spread of 2–5 return units, these differences are within noise and we draw no conclusion about robustness.

M Compute

Table A8: Seconds per 1k environment steps (between 5k and 30k steps, seed 0), one run at a time on an otherwise idle machine, and the implied time for a 300k-step run. CPU: AMD Ryzen 7 7840HS; GPU: RTX 4060 laptop.

	CPU s/1k	GPU s/1k	CPU min/300k	GPU min/300k
IQL	1.1	1.6	5.7	8.0
VDN	1.1	1.7	5.7	8.7
QMIX	1.3	1.8	6.5	9.0
QTRAN	1.4	1.9	7.2	9.5
DCG	3.9	8.3	19.4	41.4
LC-DCG	4.2	8.6	21.0	43.2
LS-DCG	4.0	8.5	20.0	42.4

The wall-clock times recorded during the main runs were measured with several runs sharing a device (four identical runs per GPU, or seven mixed runs on the CPU) and are not used for any comparison.

N Full list of limitations

The measure. (1) δ is the complement of a bounding-box fill ratio, a descriptor long established in image analysis (Rosin, 1999); we apply it to joint action sets and do not claim the quantity, and the representational remedies it motivates (Guestrin et al., 2001; Böhmer et al., 2020) and a learned coordination-requirement indicator (Wang et al., 2022) predate this work. (2) It is defined on a discretised action grid and its value depends on K and on the shape of F (Proposition 2); only the ranking of instructions is claimed. Enumeration costs K^n rollouts per state and does not scale to large teams. (3) A fill ratio conflates a thin set with a diffuse one of the same size, and δ ranks sets almost exactly as total correlation does, so its advantage is interpretability rather than a different ordering (Appendix E).

The task. (4) One object class per team size: a rod for two robots and a box for three and four; all learning experiments use two robots. The measured object is a Boolean array whose entries come from hand-written success predicates. (5) Those predicates are ours; a threshold sweep is reported (Appendix G) but they remain a modelling choice, and the partial-coupling predicate for three and four robots did not constrain the other robots as intended (Appendix D). (6) `each robot just move` is a product set by construction for two robots, so the instruction-dependence claim rests on `push forward` against `move any way`. (7) Fig. A3 shows one hand-chosen state. (8) `turn the rod` admitted no joint action in the enumeration environment, so only `push forward` can be compared across the two environments.

The learning comparison. (9) It uses shipped hyperparameters rather than a per-environment search (Papoudakis et al., 2021), and the DCG family ships with a replay buffer of 500 episodes against 5000 for the factorised methods, a difference that may explain part of the gain of the edgeless controls. (10) Five value-based learners were run; QPLEX (Wang et al., 2021), Weighted

QMIX (Rashid et al., 2020) and CASEC (Wang et al., 2022) were not. (11) The tests of δ against learning cost use a one-step bandit in which both learners saturate, the kind of constructed object the introduction criticises. (12) The uncoupled set $\mathcal{U}$ was specified by us before δ was measured on the learning environment, whose `independent` reward is additive by construction, so the edge-gating variant tests a prior the measurement later confirmed rather than a selection made by Algorithm 2. (13) The final evaluation falls where every method has declined from its peak, and the cause of the decline is not established (Appendix J). (14) The noise sweep uses three seeds, and per-instruction returns were logged only for runs started after the first eight-seed comparison. (15) The multi-instruction return averages three instructions whose reward scales differ, which inflates evaluation variance. (16) Compute is measured on one laptop, one short run per method and device (Appendix M). (17) `rware-tiny` was attempted and dropped because every method scored 0.000 at our step budget.