ACTING IN THE RIGHT BASIS: TEST-TIME COMPOSITIONAL GENERALISATION FOR OFFLINE GOAL-CONDITIONED RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Robotic manipulation goals combine several factors, such as the placements of different objects. Offline data cover each factor but rarely their combination, so a goal-conditioned agent must generalise compositionally. Test-time methods improve a frozen agent, the backbone, without new data, yet they compose along time. They chain dataset states by graph search or fine-tune on retrieved sub-trajectories. When no dataset state realises the goal combination, the chain keeps a long final hop. Retrieval then teaches nothing beyond the backbone value. We compose along factors instead. Our method, GC-FACT, shows the frozen policy one single-factor target at a time. Such a target copies the current state and moves one object to its goal placement. Play data contain many such changes. This works only if the policy can carry out each target. That depends on how the goal is split into units, which we call the basis. We prove that the serial plan is near optimal when units are atomic and weakly coupled. We also prove that the relative cost of each target depends on the basis, not only on the environment. On the Lights Out puzzles, per-cell targets are unreachable or costly. Per-press targets are cheap, yet monotone descent over them stalls. A linear solve over the binary field finds the exact plan. GC-FACT lifts three frozen backbones on nine OGBench manipulation tasks. Their average success rises from 34.4, 9.3 and 5.0 percent to 60.7, 45.0 and 32.6 percent. Graph search lowers all three averages.

1 INTRODUCTION

Offline goal-conditioned reinforcement learning trains one policy to reach any goal from a fixed dataset without reward labels (Schaul et al., 2015; Levine et al., 2020; Park et al., 2025). On robotic manipulation a goal fixes several factors at once, such as the placements of different objects. Play data move one object at a time, so they cover each factor of an evaluation goal but rarely the whole combination (Figure 1(a)). We call such goals *out of combination*. Reaching them requires compositional generalisation, which the trained policy often lacks. Test-time adaptation could supply it without new data, since it reuses the dataset and the learned value of the trained agent, the backbone.

Existing test-time methods compose along time. Test-time graph search chains dataset states into subgoals for the frozen backbone (Opryshko et al., 2026). Goal-conditioned test-time training temporarily fine-tunes the backbone on retrieved sub-trajectories (Bagatella et al., 2026). Both report large gains on maze navigation, where a goal is a single position. An out-of-combination goal, in contrast, is rarely realised by any dataset state. In the limit where none realises it, every chain of dataset states keeps a long final hop, and retrieval learns nothing beyond the backbone value (Proposition 5.2 and Figure 1(b)). Before this limit, dataset waypoints still fix the other objects at intermediate values that the goal does not require. On manipulation these temporal methods rarely improve the backbone and often degrade it (Figure 1(c) and Section 6.1).

We compose along factors instead of time. At each step our method, GC-FACT, picks the misplaced object that seems cheapest to fix. The frozen policy then sees the current state with only this object moved to its goal placement. Play data are full of such one-object changes, even if the target state as a whole never occurs in them. The plan never routes the agent through a dataset state (Figure 1(b),

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.

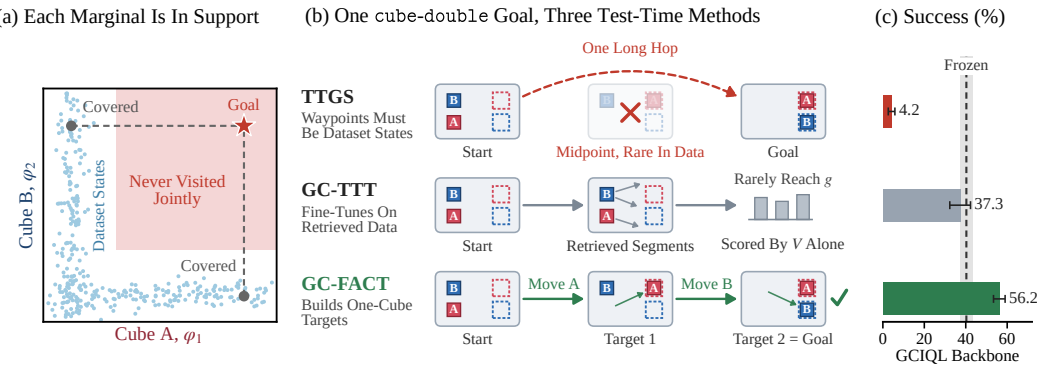

Figure 1: **Out-of-combination goals.** **(a)** Play data cover each goal factor but rarely the joint configuration (Definition 1). **(b)** A `cube-double` goal moves both cubes. TTGS routes only through dataset states, and the midpoint it needs is itself a rare combination, so one long hop remains (Proposition 5.2). GC-TTT fine-tunes on retrieved segments that rarely reach the goal, so only the backbone value scores them. GC-FACT builds its own targets and moves one cube at a time. **(c)** Success rate with the GCIQL backbone and the TTGS setting of Appendix E (Tables 1 and 9).

bottom row). GC-FACT needs no training and no graph. Whether the policy can execute every target depends on how we split the goal into units. We call this choice the basis.

The Lights Out puzzles of OGBench make this dependence explicit. A press toggles a cell together with its neighbours. If each unit is a cell, a target that fixes one cell is often unreachable. When reachable, it can cost more presses than the whole goal. The agent then falls below the frozen backbone. If each unit is a press, every target costs one press. Yet greedy search over presses, called monotone descent, halts when a single cell is wrong. A linear solve over the binary field $\mathbb{F}_2$ gives the exact plan without search. A frozen agent thus generalises compositionally only if it acts in the right basis. The units of this basis must be atomic. Every target must be reachable and no harder than the goal. The basis must also admit an efficient planner. This is the claim of the paper.

- We formalise out-of-combination goals, measure how closely OGBench approaches them (Table 6) and prove that, in the limit, every path through dataset states keeps a long final hop (Proposition 5.2).
- We define actionability by a factor cost ratio and prove that the serial factored plan is near optimal when units are atomic and weakly coupled (Theorem 5.4). The ratio depends on the basis (Proposition 5.5), and an actionable basis can still defeat monotone descent, which an exact solver avoids (Proposition 5.6 and Corollary 5.7).
- We implement this composition as GC-FACT, prove its fallback and name the regimes where its guarantees lapse (Propositions 5.8 and 5.9 and Remark B.1). It beats the frozen backbone beyond the seed spread on 20 of 27 pairs. The other seven fall in these regimes.

2 Related Work

Offline goal-conditioned RL. Offline GCRL learns to reach arbitrary goals from a fixed dataset (Lange et al., 2012; Levine et al., 2020; Liu et al., 2022), usually with goal relabelling (Kaelbling, 1993; Andrychowicz et al., 2017). Some methods imitate relabelled actions (Lynch et al., 2020; Ghosh et al., 2021) or fit a value by in-sample temporal difference, as GCIQL does with implicit Q-learning (Kostrikov et al., 2022). Other families fit contrastive (Eysenbach et al., 2022; Zheng et al., 2024) or quasimetric (Pitis et al., 2020; Wang and Isola, 2022; Wang et al., 2023; Myers et al., 2026b; Zheng et al., 2026) values, or use one value twice in a hierarchy (Park et al., 2023) that later work abstracts, flattens or shortens (Ahn et al., 2026; Zhou and Kao, 2026; Giammarino and Qureshi, 2026; Park et al., 2026a). Others reshape the goal representation (Ma et al., 2023; Park et al., 2024; 2026b) or the architecture (Lei et al., 2026; Bao et al., 2026). OGBench finds no single leader among them (Park et al., 2025). Each can serve as our backbone.

Compositional generalisation and factored goals. A goal-conditioned agent generalises compositionally when it reaches goals whose parts appear in the data but never together. Factored MDPs

expose state variables that planning and learning can exploit (Kearns and Koller, 1999; Guestrin et al., 2003). Object-centric agents represent a scene as a set of objects (Zadaianchuk et al., 2021; Li et al., 2020), and entity-centric agents generalise to more objects than seen in training (Haramati et al., 2024). Goal modules (Colas et al., 2019), disentangled skills (Hu et al., 2024), aligned task representations (Myers et al., 2026a) and latent goal analogies (Kim et al., 2026) add further structure during training. Task algebras combine the values of whole tasks without further learning (Nangue Tasse et al., 2020), and planners chain subgoals along time in graphs, learned models or diffusion samples (Eysenbach et al., 2019; Wang et al., 2024; Hong et al., 2023; Jain and Ravanbakhsh, 2024). GC-FACT instead composes the factors of a single goal at test time with a frozen backbone, and the coordinates of the factorisation decide whether this composition succeeds.

Test-time adaptation. TTGS searches a graph of dataset observations weighted by the learned value (Opryshko et al., 2026). Its authors report largest hop ratios of 0.04 to 0.22 on locomotion and 0.46 to 0.62 on manipulation. GC-TTT fine-tunes the backbone on retrieved sub-trajectories in a receding horizon (Bagatella et al., 2026). Its authors note that it needs reasonable values and data related to the current state and goal. They observe that, without a critic, relevant sub-trajectories of play data tend to share one return. Both diagnoses concern how far the data sit from the goal, which Proposition 5.2 ties to its combinatorial structure. We borrow the classical Lights Out algebra (Anderson and Feil, 1998) to show that factored composition can fail in the wrong basis.

3 BACKGROUND

3.1 OFFLINE GOAL-CONDITIONED REINFORCEMENT LEARNING AND TEMPORAL DISTANCE

A goal-conditioned problem is a controlled Markov process $M = (\mathcal{S}, \mathcal{A}, \mu, p)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, initial distribution μ and transition kernel $p(s' \mid s, a)$. Goals are states, and g is reached when the state enters its success set $\mathcal{G}(g) \subseteq \mathcal{S}$, which the environment tolerance defines. The backbone is a value V and a policy $\pi(a \mid s, g)$, trained offline on a fixed dataset $\mathcal{D}$ of trajectories with state set $\mathrm{supp}(\mathcal{D})$ and frozen at test time.

Under the reward $r(s, g) = -\mathbb{1}[s \notin \mathcal{G}(g)]$ and discount $\gamma \in (0, 1)$, a policy π has the value $V^\pi(s, g) = \mathbb{E}_\pi[\sum_{t \geq 0} \gamma^t r(s_t, g)]$. The optimal value $V^* = \max_\pi V^\pi$ defines a temporal distance,

$$d^*(s, g) = \log_\gamma\big(1 + (1 - \gamma)V^*(s, g)\big), \qquad \text{so that} \qquad -V^*(s, g) = \frac{1 - \gamma^{\,d^*(s,g)}}{1 - \gamma}. \tag{1}$$

In a deterministic process $d^*(s, g)$ is the length of a shortest path from s into $\mathcal{G}(g)$, and it is infinite when no path exists. We write $d^*(s, B)$ for the same quantity with a target set $B \subseteq \mathcal{S}$ in place of $\mathcal{G}(g)$, and $\hat{d}$ for its estimate from the backbone value V.

3.2 TEMPORAL TEST-TIME METHODS

Two prior methods compose along time. The first, TTGS, is also the parent planner that GC-FACT falls back to when a goal has no factor structure. TTGS samples vertices from $\mathrm{supp}(\mathcal{D})$, weights each edge by $\hat{d}$ and penalises edges longer than a trust region τ (Opryshko et al., 2026). A shortest path $(z_0, \ldots, z_L)$ between the vertices nearest to s_0 and g forms the subgoal chain. At each step the policy receives the furthest point of the chain whose predicted distance stays below a subgoal threshold T_s. The quality of the chain is its largest hop ratio,

$$\rho_{\mathrm{hop}}(s_0, g) = \frac{\max\{\hat{d}(s_0, z_0),\ \hat{d}(z_0, z_1), \ldots,\ \hat{d}(z_L, g)\}}{\hat{d}(s_0, g)}, \tag{2}$$

and ρ^*_{hop} denotes the same ratio of a given path with d^* in place of $\hat{d}$. GC-TTT retrieves the sub-trajectories $(s_1, \ldots, s_H)$ of $\mathcal{D}$ that start near the current state s (Bagatella et al., 2026). Near means $\hat{d}(s, s_1) < \xi$ for a retrieval radius $\xi > 0$. It ranks them by how well they seem to reach g under the backbone value. It then briefly fine-tunes the backbone on the best ones (Appendix F). Both methods thus need dataset states markedly closer to g than s_0.

3.3 FACTORED GOALS

Manipulation states carry object structure, which we write as a product (Figure 2(a)).

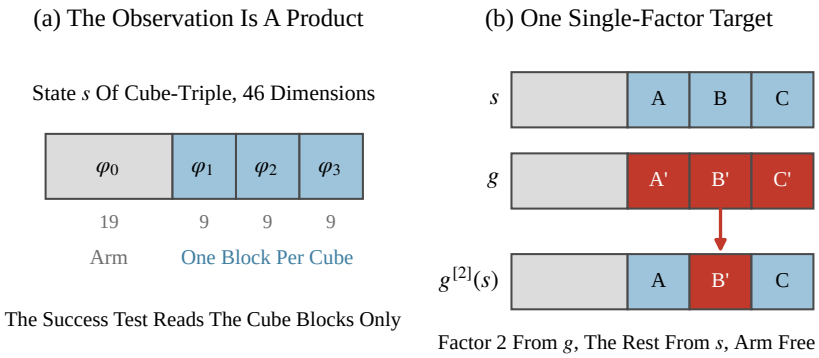

Figure 2: **A factored goal on `cube-triple`. (a)** The observation is an arm block plus one block per cube, and success reads the cube blocks alone (Assumption 3.1). **(b)** The single-factor target $g^{[2]}(s)$ of Equation (3) takes block 2 from the goal and the other cube blocks from s, with the arm block free. Only one unit must move, a change that the play data contain.

Assumption 3.1 (Product state space)**.** *There are maps $\phi_k : \mathcal{S} \to \mathcal{S}_k$ for $k = 0, \dots, K$ such that $s \mapsto (\phi_0(s), \dots, \phi_K(s))$ is a bijection onto $\mathcal{S}_0 \times \cdots \times \mathcal{S}_K$. Each object factor space $\mathcal{S}_k$, $k \geq 1$, carries an equivalence relation $\simeq$ encoding the environment tolerance, and $s' \in \mathcal{G}(g)$ if and only if $\phi_k(s') \simeq \phi_k(g)$ for $k = 1, \dots, K$.*

The body factor ϕ_0 describes the agent itself, such as the arm joints. Each object factor describes one controllable unit, such as a cube or a button, and $\Phi = \{\phi_k\}_{k=1}^{K}$ is a *factorisation* of order K. We write $\mathcal{U}(s, g) = \{k : \phi_k(s) \not\simeq \phi_k(g)\}$ for the unsatisfied factors and $K' = |\mathcal{U}(s, g)|$. Assumption 3.1 holds on the state-based OGBench manipulation tasks up to two idealisations. Their success test is a per-object tolerance (Park et al., 2025), which is not transitive, and the equivalence $\simeq$ idealises it (Table 7). The product space also contains configurations that collisions exclude.

4 GC-FACT: TEST-TIME COMPOSITION VIA SINGLE-FACTOR TARGETS

GC-FACT wraps a frozen backbone and changes only the goal that its policy sees.

4.1 CONSTRUCTING SINGLE-FACTOR TARGETS

A single-factor target takes factor k from the goal g and every other object factor from the state s,

$$g^{[k]}(s) = \{s' : \phi_k(s') \simeq \phi_k(g) \text{ and } \phi_j(s') \simeq \phi_j(s) \text{ for all } j \geq 1,\ j \neq k\}, \tag{3}$$

with the body factor free (Figure 2(b)). Each target thus moves one unit. We call the choice of units a *basis*. On cube tasks and `scene` one unit per object suffices. Each target then asks for a one-unit change that play data contain. On Lights Out, pressing cell j of an $R \times C$ grid maps the colouring $\phi(s) \in \mathbb{F}_2^n$, $n = RC$, to $\phi(s) \oplus A_j$, where $\oplus$ is addition in $\mathbb{F}_2$ and column A_j of the toggle matrix $A \in \mathbb{F}_2^{n \times n}$ marks cell j and its neighbours. No press flips a single cell. GC-FACT therefore uses one unit per press, the *press basis*, rather than one per cell, the *button basis* (Figure 3).

4.2 ONE CONTROL STEP

Algorithm 1 runs at every environment step. Lines 1 to 5 handle the two special cases of Proposition 5.8. *(i)* If the backbone predicts that g is at most T steps away, the policy receives g itself. *(ii)* Otherwise every unsatisfied factor proposes its single-factor target. On Lights Out the $\mathbb{F}_2$ solve of Corollary 5.7 proposes the presses of an exact plan instead. *(iii)* A precedence filter holds back targets whose object cannot move yet, such as a locked drawer. A patience rule demotes targets that stall. *(iv)* The remaining target B with the smallest $\hat{d}(s, B)$ is passed to the policy. The policy sees it as a copy

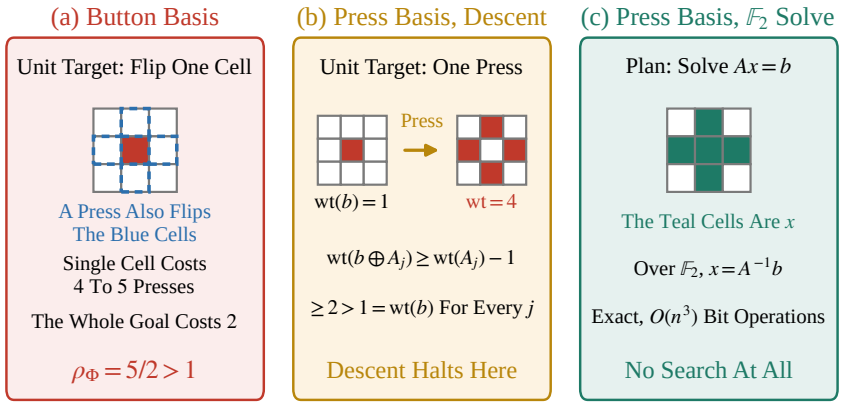

Figure 3: **Why the basis decides, on** 3×3**.** Panel (a) uses a goal that costs two presses, and panels (b) and (c) use one mismatched cell. **(a)** Button basis. A single-cell target costs four to five presses, so $\rho_\Phi = 5/2$ (Proposition 5.5(i)). **(b)** Press basis with monotone descent. Every press raises the mismatch from one cell to at least two (Proposition 5.6). **(c)** The solve of Corollary 5.7 returns the unique plan $x = A^{-1}b$, the five teal cells.

Algorithm 1 GC-FACT, one control step. Orange marks the basis-dependent step.

Require: frozen value V and policy π, factorisation Φ of order K, goal g, state s, threshold T

if Φ is undefined **then**
  return the parent planner unchanged // out of scope, Proposition 5.8
else if $K = 1$ **then**
  return $a \sim \pi(\cdot \mid s, g)$ // identity, Proposition 5.8
end if
$\mathcal{U} \leftarrow \{k : \phi_k(s) \not\simeq \phi_k(g)\}$, and **if** $\mathcal{U} = \emptyset$ or $\hat{d}(s, g) \leq T$ **return** $a \sim \pi(\cdot \mid s, g)$ // aim at g
$\mathcal{B}(s, g) \leftarrow$ ATOMICUNITS$(\Phi, \mathcal{U})$ // one unit per object, or the press basis
if press basis: keep $\{j : x_j = 1\}$ for a minimum-weight x with $Ax = \phi(s) \oplus \phi(g)$
$\mathcal{B}(s, g) \leftarrow$ FILTERPRECEDENCE$(\mathcal{B}(s, g), s, g)$ // drop blocked or stalled units
$B^\star \leftarrow \arg\min_{B \in \mathcal{B}(s,g)} \hat{d}(s, B)$, and **return** $a \sim \pi(\cdot \mid s, B^\star)$ // Theorem 5.4

of s in which only the factors that B fixes are overwritten. Repeated at every step, this greedy choice runs the unsatisfied factors in an order $\sigma = (k_1, \ldots, k_{K'})$. With $s^{(0)} = s$ and $s^{(i)} \in g^{[k_i]}(s^{(i-1)})$, its cost is

$$J_\sigma(s, g) = \sum_{i=1}^{K'} d^*\big(s^{(i-1)},\, g^{[k_i]}(s^{(i-1)})\big) \;\geq\; d^*(s, g). \tag{4}$$

The gap between the two sides comes only from interactions between units (Theorem 5.4 and Remark B.1). Appendix E gives the Lights Out fallbacks and all settings.

5 INTERPRETATION OF GC-FACT

5.1 TIME VERSUS FACTORS

We formalise the data coverage of Section 1, with proofs in Appendix B.

Definition 1 (Out-of-combination goal)**.** A goal g is *m-out-of-combination* for $\mathcal{D}$ if no dataset state satisfies more than m of its K factors. That is, $\max_{v \in \mathrm{supp}(\mathcal{D})} |\{k : \phi_k(v) \simeq \phi_k(g)\}| \leq m$. Each factor must still be attainable on its own. Thus each k has some $v \in \mathrm{supp}(\mathcal{D})$ with $\phi_k(v) \simeq \phi_k(g)$.

Table 6 in Appendix D measures both halves. Every factor of every goal appears in the data. Yet at most 0.36 percent of the sampled state-goal pairs with $K \geq 2$ match all factors. A statement about planners needs one more assumption. Every wrong factor must cost at least a fixed number of steps.

Assumption 5.1 (Separable factor cost)**.** *There is $c_{\min} > 0$ with $d^*(v, g) \geq |\mathcal{U}(v, g)|\, c_{\min}$ for all v.*

Proposition 5.2 (Dataset paths keep a long hop to an out-of-combination goal)**.** *Let g be m-out-of-combination, let Assumption 5.1 hold, and let $s_0 \notin \mathcal{G}(g)$ satisfy $d^*(s_0, g) < \infty$. Every path*

$(z_0, \ldots, z_L)$ *in* $\mathrm{supp}(\mathcal{D})$ *then satisfies*

$$\rho^*_{\mathrm{hop}}(s_0, g) \;\geq\; \frac{(K-m)\, c_{\min}}{d^*(s_0, g)}. \tag{5}$$

If $m < K$, no sub-trajectory that GC-TTT *retrieves reaches g, whatever the radius ξ. Its score then depends only on its length H and on the backbone value at the unseen goal.*

In words, the last hop of any dataset path must still fix at least $K - m$ factors. If every factor costs about the same, this hop covers at least a fraction $1 - m/K$ of the task. For $K \in [2, 4]$ and $m = 1$, that is half to three quarters of it. Opryshko et al. (2026) find that graph search stops helping above a hop ratio of about 0.4 (Figure 6 in Appendix D). Only `cube-quadruple` and `puzzle-4x6` have $m < K$ in the sampled data. The proposition thus describes a limit that the data approach.

GC-FACT avoids this hop by construction. Each single-factor target moves one unit and holds the others where they are. Play data contain many moves of this kind. The target state itself may never appear in them (Figure 1(b)). Two questions remain. How long is the chain of targets, and can the frozen policy execute each of them? We answer the first here and the second in Section 5.2. The length of the chain depends on how the units interact (Figure 4).

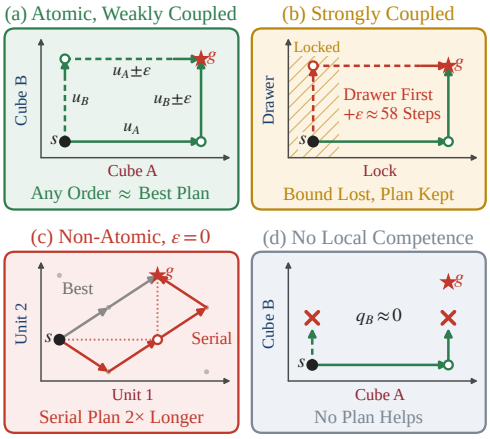

Figure 4: **When the guarantees hold.** Axes are units and u is a single-factor cost. Arrows are whole targets, in (c) single actions.

Assumption 5.3 (Atomic and weakly coupled units)**.** (i) Atomic units. *The dynamics are deterministic. Some optimal trajectory to the goal moves the unfinished units one at a time.* (ii) Weak coupling. *Finishing one unit changes the single-factor cost of any other unit by at most $\varepsilon \geq 0$ steps.*

Theorem 5.4 (The serial factored plan is near optimal under weak coupling)**.** *Under Assumptions 3.1 and 5.3, every order σ of the unfinished units satisfies*

$$0 \;\leq\; J_\sigma(s, g) - d^*(s, g) \;\leq\; K'(K'-1)\,\varepsilon. \tag{6}$$

The chain of targets is thus never shorter than the best plan and at most $K'(K'-1)\varepsilon$ steps longer. The bound holds for every order of the units (Figure 4(a)). With weak coupling the order hardly matters, so GC-FACT picks the cheapest target at each step. Hard preconditions, such as a locked drawer, fall outside the assumption. The precedence filter of Algorithm 1 handles them. With strong coupling the bound becomes loose. On `scene`, releasing a lock shortens the drawer and window targets by 57.8 and 62.2 steps on average (Figure 10). That is about the cost of a whole target. Appendix B.2 states the assumption formally and proves the theorem.

5.2 COMPOSITION REQUIRES THE RIGHT BASIS

Theorem 5.4 bounds the length of a plan. It does not say whether the frozen policy can execute each target. That depends on the basis of Section 4.1, the coordinates in which we split the goal into units. A useful basis never asks for a target that is harder than the goal itself.

Definition 2 (Factor cost ratio, actionable basis)**.** A basis assigns to each pair (s, g) a finite set $\mathcal{B}(s, g)$ of targets. Some sequence of targets, each taken from the basis at the current state, must reach $\mathcal{G}(g)$. The factor cost ratio of the basis compares the hardest target with the goal,

$$\rho_{\mathcal{B}}(s, g) = \max_{B \in \mathcal{B}(s,g)} \frac{d^*(s, B)}{d^*(s, g)}. \tag{7}$$

The basis is actionable at (s, g) if every target is reachable and $\rho_{\mathcal{B}}(s, g) \leq 1$. The factorisation induces the basis $\mathcal{B}_\Phi(s, g) = \{g^{[k]}(s) : k \in \mathcal{U}(s, g)\}$, whose ratio we write ρ_Φ.

The next result is the centre of the paper. On Lights Out, one task passes or fails this test depending only on the basis.

Proposition 5.5 (Actionability is a property of the basis)**.** *Consider Lights Out with the toggle matrix A and the residual $b = \phi(s) \oplus \phi(g)$ of Section 4.1. Let $b \neq 0$ and let the goal be reachable. Count the cost in presses.* (i) Button basis. *Let each cell be a unit. If some mismatched cell cannot be flipped on its own, then $\rho_\Phi = \infty$. If A is invertible, flipping cell k on its own takes $\mathrm{wt}(A^{-1}e_k)$ presses, where e_k marks cell k. The whole goal takes $\mathrm{wt}(A^{-1}b)$ presses. Hence $\rho_\Phi > 1$ as soon as one mismatched cell costs more than the goal.* (ii) Press basis. *Let each press be a unit. Every target then costs one press. Hence $\rho_\Psi = 1/w^*(b) \leq 1$, where $w^*(b)$ is the least number of presses that reach the goal.*

The environment, the data and the policy are the same in (i) and (ii). Only the coordinates change. The button basis also breaks Assumption 5.3(i), since one press moves three to five cells at once. Its serial plan can therefore be several times longer than the goal (Remark B.1 and Figure 4(c)). The press basis passes both tests. It is still not enough, because a planner must also find the right presses.

Proposition 5.6 (An actionable basis does not make monotone descent work)**.** *Consider a grid with at least two rows and two columns. At each step, monotone descent applies the press that most reduces the number of mismatched cells. It halts when no press reduces it. If exactly one cell is mismatched, every press leaves at least two cells mismatched. Monotone descent then halts before the goal, even when a plan exists.*

Corollary 5.7 (Exact planning in the press basis)**.** *Let $x \in \mathbb{F}_2^n$ record which cells are pressed an odd number of times. The presses of x reach the goal if and only if $Ax = b$. Gaussian elimination over $\mathbb{F}_2$ finds such an x in $O(n^3)$ bit operations whenever one exists. Enumerating the $2^{\dim \ker A}$ solutions yields a plan with the fewest presses. This plan is unique when A is invertible.*

This is the classical solution of Lights Out (Anderson and Feil, 1998), used here for a new purpose. Read together, Propositions 5.5 and 5.6 and Corollary 5.7 separate two needs of factored composition. The basis must keep every target cheap and reachable. The planner must then find targets that add up to the goal. On Lights Out only the press basis with the $\mathbb{F}_2$ solve meets both needs. These are also the grids where GC-FACT gains most (Section 6.2). Appendices B.3 and B.4 give the formal statements and proofs.

5.3 WHEN THE GUARANTEES LAPSE

GC-FACT has a safe fallback, and its guarantees have clear limits.

Proposition 5.8 (Exact degradation)**.** *Without a factor map, as for locomotion or pixel observations, line 2 of Algorithm 1 returns the parent planner. The two agents then act identically. With a single factor, the single-factor target equals the goal. Line 4 then queries the frozen backbone on the raw goal.*

Proposition 5.9 (Where the guarantees lapse)**.** (i) Strong coupling. *If $K'(K'-1)\varepsilon \geq d^*(s,g)$, Theorem 5.4 no longer bounds the plan.* (ii) No local competence. *Let q_k be the highest probability that the frozen policy finishes unit k within the step budget. The maximum runs over start states and over rules for choosing the goal input.* GC-FACT *then succeeds with probability at most $\min_{k \in \mathcal{U}(s_0,g)} q_k$, whatever its schedule, basis or solver.* (iii) Trivial factorisation. *If $K = 1$,* GC-FACT *is the identity (Proposition 5.8).*

Case (i) removes the bound but keeps the plan (Figure 4(b)). A coupled target usually stays reachable and only grows longer. Only cases (ii) and (iii) can rule out a gain. Case (ii) does so only when $\min_k q_k$ is close to the frozen success rate (Figure 4(d)). Section 6.3 finds each case where the proposition places it. Remark B.1 in Appendix B.8 adds a fourth case, units that one action moves together (Figure 4(c)). The button basis of Section 6.2 shows it.

6 EXPERIMENTS

Setup. We use the state-based OGBench manipulation suite (Park et al., 2025). It has four cube tasks, `scene` and four Lights Out puzzles on 3×3, 4×4, 4×5 and 4×6 grids. Their goals approach the out-of-combination limit (Table 6). Success is averaged over five evaluation goals with 50 rollouts each. We report the mean and standard deviation over evaluation runs from up to six training seeds. We call this standard deviation the seed spread. It mixes training seeds with repeated evaluations of the same checkpoint. The backbones GCIQL (Kostrikov et al., 2022; Park et al., 2025), HIQL (Park

Table 1: **OGBench manipulation, success rate (%).** Mean ± standard deviation over one to eight evaluation runs from up to six training seeds. **Orange** marks the best mean of a block when it exceeds the underlined runner-up by more than the larger of their standard deviations. Rows marked † have one factor, where GC-FACT is the identity (Proposition 5.8). Averages use unrounded means.

	GCIQL			HIQL			OTA		
Environment	frozen	+TTGS	+GC-FACT	frozen	+TTGS	+GC-FACT	frozen	+TTGS	+GC-FACT
cube-single†	70.3 ±4.2	38.5 ±6.5	66.9 ±9.8	13.1 ±3.1	11.4 ±3.2	13.7 ±1.7	11.3 ±2.2	13.2 ±2.5	10.5 ±2.1
cube-double	40.3 ±2.9	4.2 ±1.6	**56.2** ±2.8	6.2 ±0.5	2.1 ±0.7	**11.7** ±2.4	2.4 ±1.2	1.1 ±0.5	**5.4** ±1.9
cube-triple	2.1 ±0.9	0.3 ±0.5	**6.3** ±2.3	4.0 ±1.6	1.1 ±0.7	**7.9** ±3.0	1.6 ±1.2	0.7 ±0.6	**4.3** ±2.2
cube-quadruple	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.1 ±0.2	0.0 ±0.0	0.1 ±0.2	0.2 ±0.2	0.0 ±0.0	0.0 ±0.0
scene	52.2 ±5.0	17.5 ±7.5	56.9 ±2.4	34.3 ±3.4	9.5 ±3.9	**53.9** ±4.0	22.5 ±5.5	5.8 ±2.6	**49.3** ±8.1
puzzle-3x3	94.7 ±2.6	99.6 ±0.3	99.4 ±0.5	12.2 ±1.5	8.8 ±3.0	**70.8** ±2.7	4.1 ±0.9	3.5 ±1.4	**7.2** ±2.4
puzzle-4x4	25.6 ±3.4	28.6 ±4.7	**92.7** ±2.6	7.6 ±3.8	4.2 ±3.0	**47.0** ±15.6	2.3 ±1.4	1.1 ±0.2	**17.2** ±7.1
puzzle-4x5	12.9 ±2.3	2.8 ±3.7	**79.1** ±1.9	3.3 ±1.8	0.4 ±0.7	**99.9** ±0.2	0.4 ±0.0	0.8 ±0.6	**99.2** ±1.1
puzzle-4x6	11.3 ±2.6	1.1 ±1.1	**89.3** ±4.1	3.2 ±1.6	0.4 ±0.1	**100.0** ±0.0	0.4 ±0.6	0.4 ±0.3	**99.9** ±0.2
Average	34.4	21.4	**60.7**	9.3	4.2	**45.0**	5.0	3.0	**32.6**

et al., 2023) and OTA (Ahn et al., 2026) are trained with their reference configurations and frozen. We compare with TTGS on all three backbones and with GC-TTT on GCIQL and HIQL (Appendix F). Every planner in a block runs on the same checkpoint. GC-FACT shares one configuration across all tasks and backbones (Table 8).

6.1 MAIN RESULTS

Factored composition helps where the conditions hold. GC-FACT raises the average success of all three backbones (Table 1). GCIQL rises from 34.4 to 60.7 percent, HIQL from 9.3 to 45.0 and OTA from 5.0 to 32.6. GC-FACT exceeds the frozen backbone by more than the larger standard deviation on 20 of the 27 pairs. The other seven pairs belong to the three regimes of Proposition 5.9. No pair falls below the backbone by more than the spread. On `puzzle-4x5` and `puzzle-4x6` GC-FACT reaches 79.1 to 100.0 percent with every backbone, from at most 12.9 percent frozen. Monotone descent completes at most 0.18 percent of the reachable residuals there (Table 4). The $\mathbb{F}_2$ solve supplies exactly the plan that the backbone lacks (Figure 13).

Temporal composition does not help. TTGS lies below the frozen backbone on 21 of the 27 pairs. On 13 of them the gap exceeds the larger standard deviation. TTGS exceeds this spread only on `puzzle-3x3` with GCIQL, where the frozen agent already reaches 94.7 percent. Its losses reach 36.1 points and occur even on `cube-single`, whose goals the data cover (Table 6). Our TTGS uses $T_s = 48$, so it mostly steers the policy toward dataset states. With T_s tuned per task, Opryshko et al. (2026) report gains of at most two points on four of these tasks. Retrieval fails differently. GC-TTT stays within 3.0 points of the backbone on 14 of its 16 pairs (Table 9). The second clause of Proposition 5.2 predicts this in the limit.

Execution caps an exact plan. On 3×3 and 4×4 the exact plans yield 99.4 and 92.7 percent with GCIQL. With OTA they yield only 7.2 and 17.2 percent. Exact press targets nevertheless produce about 98 percent of the OTA steps (Figure 9). This gap is the local competence of Proposition 5.9(ii). HIQL and OTA already abstract along time inside the backbone (Park et al., 2023; Ahn et al., 2026). Yet they gain 19.6 and 26.8 points on `scene` and 96.6 and 98.8 on `puzzle-4x5`. A temporal abstraction inside the backbone thus does not replace composition along factors.

6.2 THE BASIS IS THE DECIDING VARIABLE

Figure 5 reports exact properties of the toggle matrix. They hold for every backbone and seed without a rollout. A single-cell target costs four to five presses on 3×3 and eight to twelve on 4×5 and 4×6. None is reachable on 4×4. The button basis therefore asks for an unreachable or costlier target on 41.0 to 100 percent of the reachable residuals. Monotone descent completes at most 9.01 percent of them, while the $\mathbb{F}_2$ solve completes all.

Table 2 closes the loop with rollouts on OTA. It fixes the environment, the checkpoint and the schedule, and it varies only the coordinates. The button basis falls below the frozen backbone on all four grids and to zero on 4×5 and 4×6, consistent with the long serial plans of Remark B.1. Monotone descent stays within 1.6 points of the backbone on three grids. It gains 6.8 points on 4×5,

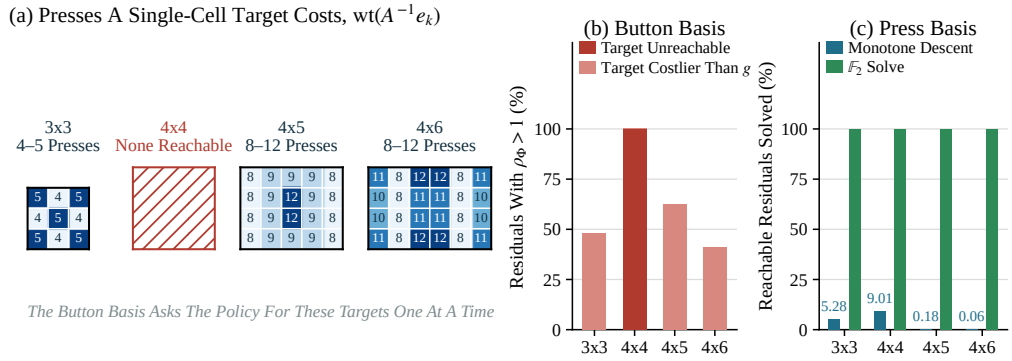

Figure 5: **Exact basis analysis. (a)** Press cost $\mathrm{wt}(A^{-1}e_k)$ of each single-cell target, none reachable on 4×4. **(b)** Residuals with $\rho_\Phi > 1$. **(c)** Residuals that monotone descent and the $\mathbb{F}_2$ solve complete.

Table 2: **Changing only the basis.** OTA backbone, success rate (%), mean ± standard deviation over up to six training seeds. The last three rows realise Proposition 5.5(i), Proposition 5.6 and Corollary 5.7. The frozen and TTGS rows come from the ablation runs and differ from Table 1 by less than one point.

Coordinates	3×3	4×4	4×5	4×6
frozen backbone	4.0 ±1.1	2.5 ±0.9	0.7 ±0.5	0.5 ±0.3
+TTGS, parent planner	4.2 ±0.8	1.6 ±0.2	0.8 ±0.3	0.5 ±0.2
button basis	2.9 ±1.0	0.5 ±0.5	0.0 ±0.0	0.0 ±0.0
press basis, descent	3.2 ±0.3	4.1 ±2.1	7.5 ±1.5	1.7 ±0.2
press basis, $\mathbb{F}_2$ solve	**6.8** ±1.5	**16.6** ±5.5	**99.6** ±0.4	**100.0** ±0.0

whose first goal is one of the two goals that descent reaches (Table 5). The $\mathbb{F}_2$ solve lifts 4×5 from 0.7 to 99.6 percent and 4×6 from 0.5 to 100.0 percent. It gains 2.8 and 14.1 points on 3×3 and 4×4, where the OTA checkpoints rarely complete an exact plan.

6.3 SCOPE, ABLATION AND COST

The three regimes of Proposition 5.9 appear where the theory places them (Figure 8). Case (i) is `scene`, where the coupling of Section 5.1 makes Equation (6) vacuous. The plan still helps, by 4.7, 19.6 and 26.8 points on the three backbones. The lost guarantee shows in the spread instead, since `scene` has the widest per-goal spread on OTA, from -38 to $+82$ points (Figure 11). Case (ii) is consistent with `cube-quadruple`, where no agent exceeds 0.2 percent, although its estimated OTA factor cost ratio is only 0.20. Case (iii) is `cube-single`, where GC-FACT is the identity and the differences of -3.4, $+0.6$ and -0.8 points lie inside the seed spread.

Table 10 ablates one component at a time on OTA. No change exceeds the spread of the reported configuration, so the gain comes from the factored targets rather than from these safeguards and modules. On navigation, which has no factor map, GC-FACT matches its parent planner within 3.6 points (Table 11). It needs 0.1 s of setup and 13.5 ms per step on `cube-triple`, against 65.6 s and 16.1 ms for TTGS (Table 12).

7 CONCLUSION

Offline goal-conditioned agents face goals that the data cover factor by factor but rarely as a whole. Test-time methods that compose along time rarely help. GC-FACT composes along factors. It shows a frozen policy one single-factor target at a time. This works under three conditions. Units must be atomic. Targets must be reachable and no harder than the goal. The basis must admit efficient planning. Lights Out meets all three only after a change of coordinates and an $\mathbb{F}_2$ solve. There the gains are largest. A frozen agent generalises compositionally at test time when it acts in the right basis. Three design choices limit the scope: state observations, the known toggle matrix and hand-written preconditions. Learning the factor maps and the atomic basis from data is the natural next step.

AI USE STATEMENT

We used generative AI tools to assist with English writing, translation from internal drafts, language polishing, and LaTeX/table formatting. We did not use generative AI tools to formulate the research idea, design the methodology or experiments, implement the proposed method, generate experimental results, or interpret the results. All AI-assisted content was reviewed and edited by the authors, who take full responsibility for the final submission.

ETHICS STATEMENT

This work studies test-time composition in offline goal-conditioned reinforcement learning. All experiments run in simulation on the public OGBench benchmark. They involve no human subjects and no personal data, and we release no new dataset. In the reported configuration, GC-FACT learns no parameters. It only changes the goal inputs of a frozen policy. Its risks are therefore those of better robot control in general. A deployment on physical robots would still need standard safety measures. A frozen policy can fail on goals outside its competence, and Section 5.3 names these regimes. We see no further ethical concern specific to this work.

REPRODUCIBILITY STATEMENT

Section 4 and Algorithm 1 describe GC-FACT step by step. Appendix E gives the factor layouts (Table 7) and the full configuration (Table 8). It also describes the Lights Out fallbacks and the settings of the parent planner. Section 5 states every assumption, and Appendix B gives complete proofs of all results. The diagnostics of Appendix C are exact. They depend only on the toggle matrix and the OGBench evaluation goals, so anyone can check them without training. Appendix D describes how we measure data coverage. Section 6 and Appendix F give the evaluation protocol, the baseline settings, the ablations and the compute cost. All experiments use the public OGBench datasets and evaluation goals, and every backbone uses its reference configuration. Upon publication, we commit to release the full source code, configuration files, and evaluation scripts.

REFERENCES

Hongjoon Ahn, Heewoong Choi, Jisu Han, and Taesup Moon. Option-aware temporally abstracted value for offline goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 38:99833–99861, 2026.

Marlow Anderson and Todd Feil. Turning lights out with linear algebra. *Mathematics magazine*, 71 (4):300–303, 1998.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Neural Information Processing Systems (NeurIPS)*, 2017.

Marco Bagatella, Mert Albaba, Jonas Hübotter, Georg Martius, and Andreas Krause. Test-time offline reinforcement learning on goal-related experience. In *Forty-third International Conference on Machine Learning*, 2026.

Erdemt Bao, Xing Lei, and Jun Chen. Nftr: From provable mode-averaging to geodesic subgoal selection in offline goal-conditioned rl. *arXiv preprint arXiv:2607.07855*, 2026.

Cédric Colas, Pierre Fournier, Mohamed Chetouani, Olivier Sigaud, and Pierre-Yves Oudeyer. Curious: intrinsically motivated modular multi-goal reinforcement learning. In *International conference on machine learning*, pages 1331–1340. PMLR, 2019.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. Contrastive learning as goal-conditioned reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Manon Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations*, 2021.

Vittorio Giammarino and Ahmed H Qureshi. Goal reaching with eikonal-constrained hierarchical quasimetric reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

Dan Haramati, Tal Daniel, and Aviv Tamar. Entity-centric reinforcement learning for object manipulation from pixels. In *International Conference on Learning Representations*, volume 2024, pages 29980–30014, 2024.

Mineui Hong, Minjae Kang, and Songhwai Oh. Diffused task-agnostic milestone planner. *Advances in Neural Information Processing Systems*, 36:387–405, 2023.

Jiaheng Hu, Zizhao Wang, Peter Stone, and Roberto Martín-Martín. Disentangled unsupervised skill discovery for efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 37:76529–76552, 2024.

Vineet Jain and Siamak Ravanbakhsh. Learning to reach goals via diffusion. In *International Conference on Machine Learning*, pages 21170–21195. PMLR, 2024.

Leslie Pack Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1993.

Michael Kearns and Daphne Koller. Efficient reinforcement learning in factored mdps. In *IJCAI*, volume 16, pages 740–747, 1999.

Junseok Kim, Dohyeong Kim, Mineui Hong, and Songhwai Oh. Compositional transduction with latent analogies for offline goal-conditioned reinforcement learning. In *Forty-third International Conference on Machine Learning*, 2026.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*, pages 45–73. Springer, 2012.

Xing Lei, Jincheng Wang, Xuetao Zhang, and Donglin Wang. QHyer: Q-conditioned hybrid attention-mamba transformer for offline goal-conditioned RL. In *Forty-third International Conference on Machine Learning*, 2026.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Richard Li, Allan Jabri, Trevor Darrell, and Pulkit Agrawal. Towards practical multi-object manipulation using relational reinforcement learning. In *2020 ieee international conference on robotics and automation (icra)*, pages 4051–4058. IEEE, 2020.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*, 2022.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on robot learning*, pages 1113–1132. Pmlr, 2020.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *The Eleventh International Conference on Learning Representations*, 2023.

Vivek Myers, Bill Zheng, Anca Dragan, Kuan Fang, and Sergey Levine. Temporal representation alignment: Successor features enable emergent compositionality in robot instruction following. *Advances in Neural Information Processing Systems*, 38:149934–149961, 2026a.

Vivek Myers, Bill Zheng, Benjamin Eysenbach, and Sergey Levine. Offline goal-conditioned reinforcement learning with quasimetric representations. *Advances in Neural Information Processing Systems*, 38:19654–19679, 2026b.

Geraud Nangue Tasse, Steven James, and Benjamin Rosman. A boolean task algebra for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:9497–9507, 2020.

Evgenii Opryshko, Junwei Quan, Claas A Voelcker, Yilun Du, and Igor Gilitschenski. Test-time graph search for goal-conditioned reinforcement learning. In *Forty-third International Conference on Machine Learning*, 2026.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. In *International Conference on Machine Learning (ICML)*, 2024.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pages 94937–94982, 2025.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable. *Advances in Neural Information Processing Systems*, 38:8350–8389, 2026a.

Seohong Park, Deepinder Mann, and Sergey Levine. Dual goal representations. In *The Fourteenth International Conference on Learning Representations*, 2026b.

Silviu Pitis, Harris Chan, Kiarash Jamali, and Jimmy Ba. An inductive bias for distances: Neural nets that respect the triangle inequality. In *International Conference on Learning Representations*, 2020.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning (ICML)*, 2015.

Mianchu Wang, Rui Yang, Xi Chen, Hao Sun, Meng Fang, and Giovanni Montana. GOPlan: Goal-conditioned offline reinforcement learning by planning with learned models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Tongzhou Wang and Phillip Isola. On the learning and learnability of quasimetrics. In *International Conference on Learning Representations*, 2022.

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning (ICML)*, 2023.

Andrii Zadaianchuk, Maximilian Seitzer, and Georg Martius. Self-supervised visual reinforcement learning with object-centric representations. In *International Conference on Learning Representations*, 2021.

Bill Zheng, Vivek Myers, Benjamin Eysenbach, and Sergey Levine. Scaling goal-conditioned reinforcement learning with multistep quasimetric distances. In *The Fourteenth International Conference on Learning Representations*, 2026.

Chongyi Zheng, Ruslan Salakhutdinov, and Benjamin Eysenbach. Contrastive difference predictive coding. In *International Conference on Learning Representations*, volume 2024, pages 47577–47601, 2024.

John Zhou and Jonathan Kao. Flattening hierarchies with policy bootstrapping. *Advances in Neural Information Processing Systems*, 38:82975–83007, 2026.

CONTENTS OF APPENDIX

A Notation **14**

B Proofs **15**

B.1 Proof of Proposition 5.2 15

B.2 Proof of Theorem 5.4 15

B.3 Proof of Proposition 5.5 16

B.4 Proof of Proposition 5.6 17

B.5 Proof of Corollary 5.7 17

B.6 Proof of Proposition 5.8 17

B.7 Proof of Proposition 5.9 18

B.8 Non-atomic units 18

C Rollout-free basis diagnostics **19**

D Measuring the out-of-combination assumption **20**

E Implementation details **21**

F Additional results **22**

G Qualitative rollouts in OGBench **28**

A NOTATION

Table 3 collects the symbols used in the proofs. Symbols fixed in Section 3 reappear only where a proof adds structure to them.

Table 3: Notation used in the proofs.

Symbol	Meaning
$M = (\mathcal{S}, \mathcal{A}, \mu, p)$	Controlled Markov process, without reward.
$\mathcal{G}(g)$	Success set of goal g, defined by the environment tolerance.
$d^*(s, B)$	Temporal distance from s to a target set B, defined through the optimal value as in Equation (1). It is a shortest-path length in a deterministic process. We abbreviate $d^*(s, g) = d^*(s, \mathcal{G}(g))$.
$\hat{d}$	The same quantity estimated from the frozen backbone value.
$\mathcal{D}$, $\mathrm{supp}(\mathcal{D})$	Offline dataset and the set of states it contains.
$(z_0, \ldots, z_L)$, ξ	Path of dataset states searched by TTGS, and retrieval radius of GC-TTT.
$\Phi = \{\phi_k\}_{k=1}^{K}$	Factorisation of order K. The body factor ϕ_0 is free in every target set.
$\simeq$	Equivalence of factor values at the environment tolerance.
$g^{[k]}(s)$	Single-factor target of Equation (3), with factor k in its goal class and every other object factor in its current class.
$\mathcal{U}(s, g)$, K'	Set of unsatisfied factors and its size.
m	Largest number of goal factors that a dataset state satisfies jointly, Definition 1.
σ, J_σ	Schedule over $\mathcal{U}(s, g)$ and its serial plan cost, Equation (4).
ε, $c_{\min}$	Coupling constant of Assumption 5.3(ii) and per-factor cost floor of Assumption 5.1.
$\mathcal{B}(s, g)$, $\rho_{\mathcal{B}}$	Basis and its factor cost ratio, Equation (7). We write ρ_Φ for the factor basis, which is the button basis on Lights Out, ρ_Ψ for the press basis, and ρ_{hop} for the temporal analogue of Equation (2).
q_k, T	Local competence on factor k in Proposition 5.9(ii), and the threshold below which Algorithm 1 aims at the raw goal.
Lights Out (Section 5.2)	
$\mathbb{F}_2$	The field with two elements. Its addition is written $\oplus$.
$n = RC$	Number of cells on an $R \times C$ grid.
$A \in \mathbb{F}_2^{n \times n}$	Toggle matrix. Column A_j indicates cell j and its grid neighbours.
$b = \phi(s) \oplus \phi(g)$	Residual between the current and the target colouring.
$x \in \mathbb{F}_2^n$	Press vector, where $x_j = 1$ means that cell j is pressed an odd number of times.
$\mathrm{wt}(\cdot)$, $\mathrm{supp}(\cdot)$	Hamming weight and set of nonzero coordinates of a vector.
$\mathrm{col}(A)$, $\ker A$	Column space and kernel over $\mathbb{F}_2$.
$w^*(b)$	Minimum weight of a solution of $Ax = b$, infinite when $b \notin \mathrm{col}(A)$.
e_k	The k-th standard basis vector of $\mathbb{F}_2^n$.

B PROOFS

B.1 PROOF OF PROPOSITION 5.2

Let $v \in \mathrm{supp}(\mathcal{D})$. By Definition 1 at most m factors of v agree with g, so $|\mathcal{U}(v, g)| \geq K - m$. Assumption 5.1 then gives

$$d^*(v, g) \;\geq\; |\mathcal{U}(v, g)|\, c_{\min} \;\geq\; (K - m)\, c_{\min}. \tag{8}$$

Hop bound. Take any path $(z_0, \ldots, z_L)$ with $z_\ell \in \mathrm{supp}(\mathcal{D})$. Its final hop runs from z_L to g, and Equation (8) with $v = z_L$ gives $d^*(z_L, g) \geq (K - m)c_{\min}$. The numerator of ρ^*_{hop} is at least this final hop. The denominator $d^*(s_0, g)$ is positive because $s_0 \notin \mathcal{G}(g)$, and it is finite by assumption. Hence

$$\rho^*_{\mathrm{hop}}(s_0, g) \;\geq\; \frac{d^*(z_L, g)}{d^*(s_0, g)} \;\geq\; \frac{(K - m)c_{\min}}{d^*(s_0, g)},$$

which is Equation (5). The bound holds for every path, so no choice of vertices or edge weights removes it.

Selection bound. Assume $m < K$, fix the current state s and let $\xi > 0$ be any retrieval radius. A retained sub-trajectory $(s_1, \ldots, s_H)$ consists of dataset states, so $|\mathcal{U}(s_h, g)| \geq K - m \geq 1$ for every h. By Assumption 3.1 a state with an unsatisfied factor lies outside $\mathcal{G}(g)$. Hence no s_h lies in $\mathcal{G}(g)$, whatever the value of ξ. The sparse reward then gives $r(s_h, g) = -1$ for every h, and the H-step return becomes

$$\widehat{V} = -\frac{1 - \gamma^{H-1}}{1 - \gamma} + \gamma^{H-1} V(s_H, g).$$

The first term depends on the sub-trajectory only through its length H. The second term is the backbone value at an out-of-combination goal, which is the quantity that test-time adaptation tries to improve. The filter therefore ranks the retained set by the length and by the backbone value alone. The same conclusion holds for any ranking computed from the backbone value at g, such as the minimum value-derived distance to g that the released implementation of GC-TTT uses. □

B.2 PROOF OF THEOREM 5.4

Formal statement. Assumption 5.3 reads formally as follows. *(i)* The transition kernel is deterministic. Some optimal trajectory from s to $\mathcal{G}(g)$ splits into K' contiguous segments, and along each segment every object factor but one stays in its equivalence class. *(ii)* Completing one factor changes the single-factor cost of any other by at most $\varepsilon \geq 0$. That is, $|d^*(y', g^{[j]}(y')) - d^*(y, g^{[j]}(y))| \leq \varepsilon$ for every state y, every $y' \in g^{[k]}(y)$ and every $j \neq k$. The bound of Theorem 5.4 is uniform over schedules, so the order matters only through hard preconditions. Part (i) excludes them, and the precedence filter of Algorithm 1 handles them in practice. The bound becomes empty when ε is comparable to a single-factor cost.

Proof. Write $K' = |\mathcal{U}(s, g)|$ and $\sigma = (k_1, \ldots, k_{K'})$. If $K' = 0$, then $s \in \mathcal{G}(g)$ and both sides of Equation (6) vanish, so let $K' \geq 1$. By Assumption 5.3(i) the process is deterministic, so d^* is a shortest-path length. The same assumption provides an optimal trajectory, so $d^*(s, g) < \infty$. For a state y, let $u_k(y) = d^*(y, g^{[k]}(y))$ denote the single-factor cost of factor k.

A telescoping bound. Call states $y^{(0)} = s, y^{(1)}, \ldots, y^{(K')}$ a chain of completions with order $(j_1, \ldots, j_{K'})$ if the j_l are distinct and $y^{(l)} \in g^{[j_l]}(y^{(l-1)})$ for every l. For $l < i$ the factors j_l and j_i differ, so Assumption 5.3(ii) gives $|u_{j_i}(y^{(l)}) - u_{j_i}(y^{(l-1)})| \leq \varepsilon$. This bound on a difference also means that one of the two costs is finite exactly when the other is. Summing over $l = 1, \ldots, i-1$ and using the triangle inequality for real numbers yields

$$\left|u_{j_i}(y^{(i-1)}) - u_{j_i}(s)\right| \;\leq\; (i - 1)\, \varepsilon, \tag{9}$$

and $u_{j_i}(y^{(i-1)})$ is finite exactly when $u_{j_i}(s)$ is.

Lower bound on d^*. Take the optimal trajectory of Assumption 5.3(i). Say that a segment changes factor k if ϕ_k leaves its class somewhere along the segment. Every factor of $\mathcal{U}(s, g)$ must change, because the trajectory starts outside its goal class and ends inside it. Each segment changes at most

one factor, so counting shows that the K' segments change the K' unsatisfied factors one each and that no other factor changes. Let $\kappa = (\kappa_1, \dots, \kappa_{K'})$ be the resulting order of $\mathcal{U}(s, g)$. Let $y^{(i)}$ be the state at the end of segment i, with $y^{(0)} = s$, and let δ_i be the length of segment i. After segment i the factor κ_i never changes again, and the trajectory ends in $\mathcal{G}(g)$, so $\phi_{\kappa_i}(y^{(i)}) \simeq \phi_{\kappa_i}(g)$. Every other object factor keeps its class along segment i, so $y^{(i)} \in g^{[\kappa_i]}(y^{(i-1)})$. The states $y^{(i)}$ thus form a chain of completions with order κ. Segment i is one path into $g^{[\kappa_i]}(y^{(i-1)})$, hence $\delta_i \geq u_{\kappa_i}(y^{(i-1)})$, and the segments partition the trajectory, so $d^*(s, g) = \sum_i \delta_i$. Each $u_{\kappa_i}(y^{(i-1)}) \leq \delta_i$ is finite, so Equation (9) shows that $u_k(s)$ is finite for every $k \in \mathcal{U}(s, g)$. Its left inequality gives

$$d^*(s, g) \;\geq\; \sum_{i=1}^{K'} u_{\kappa_i}(y^{(i-1)}) \;\geq\; \sum_{k \in \mathcal{U}(s,g)} u_k(s) - \varepsilon \, \frac{K'(K'-1)}{2}. \tag{10}$$

Upper bound on J_σ. Let $s^{(i)}$ be the endpoint of a shortest path from $s^{(i-1)}$ into $g^{[k_i]}(s^{(i-1)})$. The argument holds for every such choice. Assume that $s^{(0)}, \dots, s^{(i-1)}$ exist. They form the start of a chain of completions with order σ, so Equation (9) and the finiteness of $u_{k_i}(s)$ show that $u_{k_i}(s^{(i-1)})$ is finite. The target is therefore reachable and $s^{(i)}$ exists. By induction the whole serial plan exists, and the right inequality of Equation (9) gives

$$J_\sigma(s, g) = \sum_{i=1}^{K'} u_{k_i}(s^{(i-1)}) \;\leq\; \sum_{k \in \mathcal{U}(s,g)} u_k(s) + \varepsilon \, \frac{K'(K'-1)}{2}. \tag{11}$$

Feasibility. By Equation (3), stage i puts factor k_i into the class of $\phi_{k_i}(g)$ and keeps every other object factor in its class. Since $\simeq$ is transitive, a factor completed at an earlier stage keeps its goal class, and the factors outside $\mathcal{U}(s, g)$ never leave theirs. Hence $s^{(K')} \in \mathcal{G}(g)$ by Assumption 3.1. The concatenated shortest paths reach $\mathcal{G}(g)$ from s in $J_\sigma(s, g)$ steps, so $J_\sigma(s, g) \geq d^*(s, g)$.

Subtracting Equation (10) from Equation (11) gives $J_\sigma(s, g) - d^*(s, g) \leq K'(K'-1)\varepsilon$. The argument used no property of σ, so the bound is uniform over schedules. Setting $\varepsilon = 0$ makes every schedule optimal. □

B.3 Proof of Proposition 5.5

Formal statement. The targets of a basis in Definition 2 are sets $B \subseteq \mathcal{S}$. A basis is *α-actionable* at (s, g) if $\rho_{\mathcal{B}}(s, g) \leq \alpha$ and $d^*(s, B) < \infty$ for every $B \in \mathcal{B}(s, g)$, so an actionable basis is 1-actionable. A ratio above one means that some target is a larger problem than the goal itself. An infinite distance means that the environment cannot produce the target. Both conditions concern the pair $(M, \mathcal{B})$ and no algorithm. In full, Proposition 5.5 reads as follows. Consider Lights Out on an $R \times C$ grid, with n, $\phi(s)$, A and $\oplus$ as in Section 4.1. Let $b = \phi(s) \oplus \phi(g)$ be a nonzero residual in the column space $\mathrm{col}(A)$. Let e_k be the k-th standard basis vector, $\mathrm{supp}(\cdot)$ the set of nonzero coordinates and $\mathrm{wt}(\cdot)$ the Hamming weight. Assume that each press costs c steps and that only presses change the colouring. *(i)* Let ϕ_k be the colour of cell k. Then $\mathcal{U}(s, g) = \mathrm{supp}(b)$, and the target of unit k has residual e_k. If $e_k \notin \mathrm{col}(A)$ for some $k \in \mathrm{supp}(b)$, then $\rho_\Phi = \infty$. If A is invertible, then

$$\rho_\Phi(s, g) = \frac{\max_{k \in \mathrm{supp}(b)} \mathrm{wt}(A^{-1} e_k)}{\mathrm{wt}(A^{-1} b)}. \tag{12}$$

(ii) Let the units be the presses, so unit j targets the states with colouring $\phi(s) \oplus A_j$. Every target of this basis Ψ costs one press, and $\rho_\Psi(s, g) = 1/w^*(b) \leq 1$ with $w^*(b) = \min\{\mathrm{wt}(x) : Ax = b\}$.

Proof. A press of cell j maps the colouring $\phi(s)$ to $\phi(s) \oplus A_j$, since it toggles cell j and its grid neighbours and toggling is addition in $\mathbb{F}_2$ (Park et al., 2025). By assumption nothing else changes the colouring. Presses commute, and pressing a cell twice restores its colours. The colouring reached from s by a press sequence is therefore $\phi(s) \oplus Ax$, where $x \in \mathbb{F}_2^n$ records the parity of the presses on each cell. A sequence with parity x contains at least $\mathrm{wt}(x)$ presses, and pressing each cell of $\mathrm{supp}(x)$ once attains this number. Reaching a colouring at residual b' from s therefore costs $c\, w^*(b')$ steps, where $w^*(b') = \min\{\mathrm{wt}(x) : Ax = b'\}$. This cost is infinite when $b' \notin \mathrm{col}(A)$.

(i) Button basis. Here ϕ_k returns the colour of cell k, so $\mathcal{U}(s, g) = \mathrm{supp}(b)$. By Equation (3) the target $g^{[k]}(s)$ differs from the colouring of s in cell k only, so its residual is e_k. Since $b \in \mathrm{col}(A)$ and

$b \neq 0$, the goal cost $d^*(s, g) = c\, w^*(b)$ is finite and positive. If $e_k \notin \mathrm{col}(A)$ for some $k \in \mathrm{supp}(b)$, then $w^*(e_k) = \infty$, that target is unreachable and $\rho_\Phi = \infty$. If A is invertible, every system $Ax = b'$ has the unique solution $A^{-1}b'$. Hence $d^*(s, g^{[k]}(s)) = c\ \mathrm{wt}(A^{-1}e_k)$ and $d^*(s, g) = c\ \mathrm{wt}(A^{-1}b)$. Substituting both into Equation (7) gives Equation (12), in which c cancels. The ratio exceeds one exactly when $\max_{k \in \mathrm{supp}(b)} \mathrm{wt}(A^{-1}e_k) > \mathrm{wt}(A^{-1}b)$.

(ii) Press basis. The target of unit j has residual A_j, and the single press $x = e_j$ reaches it at cost c. No cheaper plan exists, because a plan of weight zero leaves the colouring unchanged and $A_j \neq 0$. Every target of Ψ is therefore reachable at cost exactly c, so the numerator of Equation (7) is c. The first paragraph gives $d^*(s, g) = c\, w^*(b)$. Since b is a nonzero element of $\mathrm{col}(A)$, we have $1 \leq w^*(b) < \infty$, so $\rho_\Psi(s, g) = 1/w^*(b) \leq 1$. When A is invertible, $w^*(b) = \mathrm{wt}(A^{-1}b)$. Both parts use the same M, the same frozen policy and the same data, so only the basis differs between them. □

B.4 Proof of Proposition 5.6

Formal statement. Let $R, C \geq 2$. Monotone descent applies at every step the press j that minimises $\mathrm{wt}(b \oplus A_j)$, with the smallest index among ties, only when this press strictly decreases $\mathrm{wt}(b)$, and it halts otherwise. The tie rule fixes the exact fractions in Table 4. For a single mismatched cell, $b = e_i$, every press strictly increases the mismatch,

$$\mathrm{wt}(b \oplus A_j) \ \geq\ \mathrm{wt}(A_j) - 1 \ \geq\ 2 \ >\ 1 = \mathrm{wt}(b) \qquad \text{for every } j. \tag{13}$$

Monotone descent thus halts before the goal, although a plan exists whenever $e_i \in \mathrm{col}(A)$.

Proof. Column A_j indicates cell j and its grid neighbours. Its weight is 5 for an interior cell, 4 on an edge and 3 at a corner, so $\mathrm{wt}(A_j) \geq 3$ on every grid with $R, C \geq 2$. Let $b = e_i$ and fix any j. Addition over $\mathbb{F}_2$ cancels a coordinate exactly when both operands have it, so

$$\mathrm{wt}(b \oplus A_j) = \mathrm{wt}(A_j) + \mathrm{wt}(b) - 2\,\big|\{i\} \cap \mathrm{supp}(A_j)\big|.$$

The intersection has at most one element, because b has a single nonzero coordinate. Hence $\mathrm{wt}(b \oplus A_j) \geq \mathrm{wt}(A_j) + 1 - 2 = \mathrm{wt}(A_j) - 1 \geq 2 > 1 = \mathrm{wt}(b)$, which is Equation (13). Monotone descent therefore has no admissible press at $b = e_i$. Since $b \neq 0$, the current configuration is not the goal, and the search halts before reaching it. By Corollary 5.7 a plan nevertheless exists whenever $e_i \in \mathrm{col}(A)$. □

B.5 Proof of Corollary 5.7

The first paragraph of Appendix B.3 shows that the presses of x reach the goal if and only if $Ax = b$. A linear system over a field is solvable if and only if $b \in \mathrm{col}(A)$, and Gaussian elimination on the augmented matrix $[A \mid b]$ decides this. Over $\mathbb{F}_2$ the elimination uses $O(n^3)$ bit operations, since it performs $O(n^2)$ row additions of cost $O(n)$ each. If the system is consistent, back substitution returns a particular solution x_0. By linearity the solution set is $x_0 \oplus \ker A$. It has $2^{\dim \ker A}$ elements, so enumeration finds a minimum-weight element in $O(2^{\dim \ker A} n)$ operations. When A is invertible, the kernel is trivial and the unique solution is $A^{-1}b$. Then $\mathrm{col}(A) = \mathbb{F}_2^n$, so every target configuration is reachable. □

B.6 Proof of Proposition 5.8

Suppose first that Assumption 3.1 does not hold, for instance because the observation is an image or a locomotion state. No factor map then exists, and line 2 of Algorithm 1 returns the parent planner before any target is built. The agent under GC-FACT is therefore the parent agent. Now let $K = 1$. By Equation (3) the target $g^{[1]}(s)$ constrains only ϕ_1, and it constrains it to the class of $\phi_1(g)$. Assumption 3.1 states that the success set depends on the object factors alone, so $g^{[1]}(s) = \mathcal{G}(g)$. The single-unit query and the raw query therefore coincide, and line 4 returns $\pi(\cdot \mid s, g)$. In both cases GC-FACT produces the same action distribution as the agent it runs on, at every state. From the same initial distribution the two agents thus induce the same trajectory distribution. □

B.7 Proof of Proposition 5.9

Formal statement. In case (ii) a rule chooses the goal input of π from the history, and q_k is the largest probability, over all start states and all such rules, that factor k reaches its goal class within the step budget.

(i) Suppose $K'(K'-1)\varepsilon \geq d^*(s,g)$. The right-hand side of Equation (6) is then at least $d^*(s,g)$. The bound thus allows $J_\sigma(s,g)$ to reach $2d^*(s,g)$ and says nothing about the quality of the serial plan. The condition holds whenever $K' \geq 2$ and $\varepsilon \geq d^*(s,g)/K'$, since then $K'(K'-1)\varepsilon \geq (K'-1)\, d^*(s,g) \geq d^*(s,g)$. The bound also says nothing about feasibility, and the targets of $\mathcal{B}_\Phi(s,g)$ may remain reachable. In `scene`, releasing a lock changes the single-factor costs of the drawer and the window by amounts comparable to a factor cost (Figure 10). This is the regime described above.

(ii) Success requires every factor $k \in \mathcal{U}(s_0,g)$ to reach its goal class within the step budget. Under GC-FACT the agent acts through $\pi(\cdot \mid s, B)$, where Algorithm 1 chooses the target B from the history. GC-FACT is therefore one of the rules over which q_k is maximised. The probability that factor k reaches its goal class is then at most q_k, and the success probability is at most $\min_{k\in\mathcal{U}(s_0,g)} q_k$. The schedule, the basis and the solver only change how B is chosen, so the bound holds for all of them.

(iii) This case follows directly from Proposition 5.8. □

B.8 Non-atomic units

Remark B.1 (Non-atomic units). Theorem 5.4 also needs Assumption 5.3(i), which fails when one action moves several units. The button basis on Lights Out shows the cost. For invertible A, completing one cell leaves the mismatch of every other cell unchanged, so each single-cell cost stays $c\ \mathrm{wt}(A^{-1}e_k)$ and $\varepsilon = 0$. The serial plan nevertheless costs $J_\sigma(s,g) = c\sum_{k\in\mathrm{supp}(b)} \mathrm{wt}(A^{-1}e_k)$, against $d^*(s,g) = c\ \mathrm{wt}(A^{-1}b)$. On task 5 of the 4×5 grid every single-cell target is cheaper than the goal, with $\rho_\Phi = 0.60$, yet the serial plan needs 92 presses against 20. The press basis restores Assumption 5.3(i), since each press changes one coordinate of $A^{-1}\phi(s)$. With $\varepsilon = 0$, Theorem 5.4 then makes every schedule optimal in the model of Proposition 5.5.

C ROLLOUT-FREE BASIS DIAGNOSTICS

This appendix reports the exact computation behind Propositions 5.5 and 5.6 and Corollary 5.7 and Figure 5. Every entry is a property of the toggle matrix and of the evaluation goals that OGBench fixes. None of it depends on a checkpoint, a seed or a rollout, so the algebraic claim of Section 5.2 can be checked without training.

Table 4 gives the per-grid quantities. On 3×3, 4×5 and 4×6 the toggle matrix is invertible, so every configuration is reachable and the press plan is unique. The button basis pays for this reachability with single-cell targets of four to twelve presses, which drive ρ_Φ above one on 41.0 to 62.4 percent of the residuals. On 4×4 the kernel has dimension four. Only 2^{12} of the 2^{16} configurations are then reachable from a given one, none of the single-cell targets is among them, and $\rho_\Phi = \infty$ on every residual. The last column extends Proposition 5.6 beyond its single-cell statement. Monotone descent completes at most 9.01 percent of the reachable residuals, and at most 0.18 percent on the two largest grids. A search that also accepts presses of equal weight and backtracks solves far more, 61.1 and 96.3 percent of the residuals on 3×3 and 4×4, but it enumerates configurations and offers no polynomial bound.

Table 4: **Algebraic diagnostics of the four OGBench Lights Out grids.** $\dim\ker A$ is the dimension of the space of quiet patterns. A single-cell target is reachable only when $e_k \in \mathrm{col}(A)$. The ratio column gives the fraction of reachable residuals with $\rho_\Phi > 1$. It uses every nonzero reachable residual, which number 511, 4095, $2^{20}-1$ and $2^{24}-1$ on the four grids. The last column gives the fraction of the same residuals that monotone descent (Proposition 5.6) completes. All entries are exact and need no rollout.

Grid	n	$\dim\ker A$	**reachable single-cell targets**	$\mathrm{wt}(A^{-1}e_k)$	$\rho_\Phi > 1$	**monotone descent solves**
3×3	9	0	$9/9$	4 to 5	48.1%	5.28%
4×4	16	4	$0/16$	unreachable	100.0%	9.01%
4×5	20	0	$20/20$	8 to 12	62.4%	0.18%
4×6	24	0	$24/24$	8 to 12	41.0%	0.06%

Table 5 applies the same computation to the five evaluation goals of each grid. The press column gives the length of the minimum-weight plan of Corollary 5.7, the ratio gives ρ_Φ at that residual, and the mark records whether monotone descent reaches the goal. Two patterns are visible. On the invertible grids the button-basis ratio exceeds one exactly on the one or two goals with the shortest plans of their grid. The per-cell decomposition thus harms most where the plan is shortest. On 4×4 the ratio is infinite on every goal. Monotone descent reaches two of the twenty goals, which are the shortest plans on 3×3 and on 4×5.

Table 5: **The twenty OGBench evaluation goals, planned exactly.** $\mathrm{wt}(b)$ counts the cells whose colour must change. The press column is $w^*(b)$, the weight of a minimum-weight solution of $Ax = b$, which on 4×4 is chosen among 2^4 solutions. The ratio is ρ_Φ of Equation (7), which is infinite on 4×4 because no single-cell target is reachable there. ✓ marks the goals that monotone descent reaches.

	task 1		task 2		task 3		task 4		task 5	
Grid	$\mathrm{wt}(b)$	presses	$\mathrm{wt}(b)$	presses	$\mathrm{wt}(b)$	presses	$\mathrm{wt}(b)$	presses	$\mathrm{wt}(b)$	presses
3×3	6	2	1	5	8	8	5	9	3	7
4×4	16	4	4	6	10	6	8	6	5	7
4×5	12	4	20	10	16	14	2	16	10	20
4×6	14	6	24	8	18	12	12	16	12	24
button-basis ratio ρ_Φ, and whether monotone descent reaches the goal										
3×3	2.50, ✓		1.00, ✗		0.62, ✗		0.56, ✗		0.71, ✗	
4×4	∞, ✗		∞, ✗		∞, ✗		∞, ✗		∞, ✗	
4×5	2.25, ✓		1.20, ✗		0.86, ✗		0.75, ✗		0.60, ✗	
4×6	2.00, ✗		1.50, ✗		1.00, ✗		0.75, ✗		0.46, ✗	

D MEASURING THE OUT-OF-COMBINATION ASSUMPTION

Definition 1 is a statement about the data, so it can be checked before any agent is trained. For every environment we sample 10^5 dataset states and pair each with the five evaluation goals. With the environment's success tolerance we count the goal factors that each state matches. Marginal coverage is the fraction of the K factors of a goal that some sampled state matches. Joint coverage counts the state-goal pairs in which the state matches at least two factors and those in which it matches all K. Definition 1 holds with $m < K$ when marginal coverage is complete and no pair matches all K factors. Proposition 5.2 then applies with m equal to the largest number of factors that any state of $\mathrm{supp}(\mathcal{D})$ matches. A sample of 10^5 states can only underestimate this maximum.

Table 6 confirms the first half everywhere and the second only approximately. Marginal coverage is complete on all nine environments. Joint coverage stays below 0.4 percent of the pairs on every environment with $K \geq 2$, and it is zero on `cube-quadruple` and `puzzle-4x6`. Hence the sample gives $m = K - 1$ on these two environments and $m = K$ on the others. On the cube tasks even partial agreement is rare, since fewer than 0.3 percent of the pairs match two factors. On `scene` and the puzzles the count of partial matches carries little information. Several of their factors take only two values, such as the lock states of `scene` and the lights of the puzzles, so a random state already matches many of them. The sampled m makes the bound of Equation (5) nontrivial only on `cube-quadruple` and `puzzle-4x6`, and the full dataset could lower even these two bounds to zero. For this reason Section 5.1 treats Definition 1 as a limiting case.

Figure 6 plots the bound of Equation (5) for $m = 1$ against the largest hop ratios that Opryshko et al. (2026) measure.

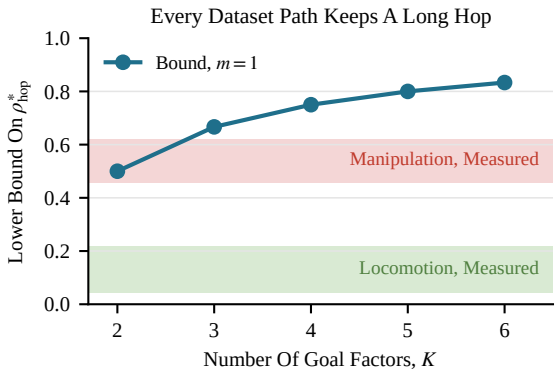

Figure 6: **The hop bound of Proposition 5.2.** The bound of Equation (5) for $m = 1$ and $d^*(s_0, g) = Kc_{\min}$, which equals $1 - 1/K$, against the largest hop ratios that Opryshko et al. (2026) measure on locomotion and manipulation.

Table 6: **Definition 1 measured on the data.** Each environment pairs 10^5 sampled dataset states with its five evaluation goals, and a factor matches when it lies within the success tolerance. Marginal coverage is the fraction of goal factors that some sampled state matches. The next two columns count the state-goal pairs, out of 5×10^5, in which the state matches at least two and all K factors of the goal. m is the largest number of factors that any sampled state matches jointly. The last column is the bound of Equation (5) with $d^*(s_0, g)$ set to $Kc_{\min}$.

Environment	K	marginal coverage	pairs matching ≥ 2 factors	pairs matching all K factors	m	bound on ρ^*_{hop}
`cube-single`	1	100%	n/a	11,205 (2.241%)	1	0.00
`cube-double`	2	100%	394 (0.079%)	394 (0.079%)	2	0.00
`cube-triple`	3	100%	986 (0.197%)	24 (0.005%)	3	0.00
`cube-quadruple`	4	100%	1,459 (0.292%)	0	3	0.25
`scene`	5	100%	310,054 (62.011%)	1,802 (0.360%)	5	0.00
`puzzle-3x3`	9	100%	490,283 (98.057%)	1,061 (0.212%)	9	0.00
`puzzle-4x4`	16	100%	499,882 (99.976%)	5 (0.001%)	16	0.00
`puzzle-4x5`	20	100%	499,992 (99.998%)	1 (< 0.001%)	20	0.00
`puzzle-4x6`	24	100%	500,000 (100.000%)	0	23	0.04

E IMPLEMENTATION DETAILS

Factor maps. On the state-based OGBench manipulation tasks the observation concatenates a proprioceptive block and one block per object (Park et al., 2025). The factor maps ϕ_k of Assumption 3.1 are the corresponding slices, which Table 7 lists. The test $\simeq$ uses the environment success tolerance in observation units. GC-FACT reads no privileged simulator state, and the goal is the goal observation that the benchmark provides. Each layout is derived once per environment and checked against the observation dimension of the dataset. A mismatch disables GC-FACT instead of guessing a layout. Figure 7 draws one control step of Algorithm 1.

Table 7: **Factor layouts derived from the OGBench observation.** The body factor ϕ_0 is the proprioceptive block and is free in every target set. Tolerances are the environment success criteria converted into observation units by the scalers OGBench applies when it builds the observation.

Family	obs. dim	$\dim \phi_0$	K	block per unit	test on the factor
`cube-{single,double,triple,quadruple}`	28/37/46/55	19	1, 2, 3, 4	9	position within 0.40
`scene`	40	19	5	9, 4, 2	position within 0.40/0.72/0.60, lock state match
`puzzle-{3x3,4x4,4x5,4x6}`	55/83/99/115	19	9, 16, 20, 24	4	discrete state match

Domain knowledge. The toggle matrix A of Proposition 5.5 encodes that a press flips a cell and its grid neighbours. It is the only model of the transition function that GC-FACT uses, and the offline agent would otherwise have to learn it. We state this choice explicitly rather than fold it into the generic method, in the spirit of the domain-specific Euclidean distance that Opryshko et al. (2026) report. The cube and scene results use no transition model. They do use hand-derived factor layouts (Table 7) and the hand-written preconditions of the precedence filter below, whose removal changes the outcome by at most 2.1 points (Table 10). Learning the factor maps and the atomic basis from data is the open problem that Section 7 names.

Schedule and precedence. The schedule of line 10 in Algorithm 1 is greedy on the value-derived distance and is recomputed at every environment step. Once $\hat{d}(s, g) \leq T$, the policy receives the raw goal. A unit whose distance fails to decrease for a patience window is demoted for a cooldown window, which stops the agent from repeating a move it cannot execute. The precedence filter encodes only hard preconditions that the factorisation exposes. A cube that is the base of a goal stack must be placed before the cube above it, and on `scene` a locked drawer or window cannot move. There the filter offers an unlock target for every locked slider that must move and withholds that slider until it is unlocked. It sets the final lock states only after every movable unit is aligned. On Lights Out the candidates are the presses of the $\mathbb{F}_2$ plan that the patience filter has not demoted. When the solve returns no plan, or the filter has demoted every press of the plan, the candidates become the presses of the monotone descent of Proposition 5.6 from the current residual, which Figure 9 labels greedy press fallbacks. When the filter has demoted these too, or the descent cannot take a first step, the candidates become the single-cell targets $g^{[k]}(s)$ of the factorisation, which Figure 9 counts as single-factor targets. No single press reaches such a target (Proposition 5.5), so these targets serve only as a last resort.

Optional modules. Five mechanisms are available and disabled in the reported configuration. Group symmetrisation averages the value and the action over a sample of the relabelling group of the identical cubes, whose order is $K!$. Value-column repair collects the single-factor distances on a bank of anchor states and denoises this matrix by a low-rank truncation. It then reads the joint distance as the sum of the per-factor distances, which is the additive rule that Theorem 5.4 justifies when ε is small. Unit blending replaces the single query of line 10 by a weighted average over the best few units. The weights are proportional to $\exp(-\lambda(\hat{d} - \hat{d}_{\min})/\bar{d})$, where $\hat{d}_{\min}$ and $\bar{d}$ are the smallest and the mean selected distances and λ sets the sharpness. Evacuation resolves the swap and cycle deadlocks of the cube tasks, in which every unaligned cube is blocked by another. It moves one blocker to a free location drawn from the anchor states. Last-layer test-time training selects transitions near the current state and takes a few gradient steps on the final actor layer. It resets the weights at the end of the episode. Table 10 reports what each module changes.

Hyperparameters. Table 8 lists the configuration of GC-FACT, which is identical for every environment and every backbone. TTGS, both as the baseline and as the parent planner, uses the trust region $\tau = 24$ and the subgoal threshold $T_s = 48$ on every manipulation environment and on eight of the nine navigation environments, and $\tau = 36$ and $T_s = 72$ on `humanoidmaze-giant-navigate`.

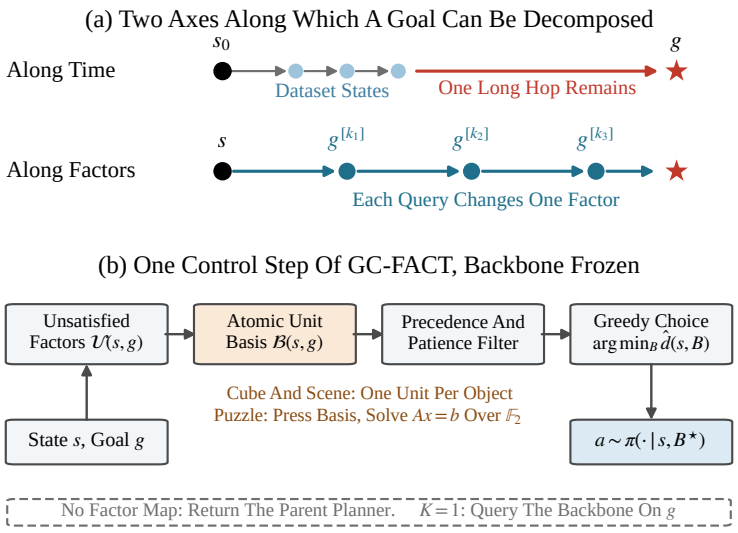

Figure 7: **GC-FACT in one picture. (a)** A temporal decomposition routes through dataset states. When these states never satisfy the goal, Proposition 5.2 shows that one hop keeps at least a fraction $(K - m)c_{\min}/d^*(s_0, g)$ of the task. A factored decomposition issues one single-factor target at a time, and each such target asks for a change of the kind the backbone met during training. **(b)** One control step. The basis supplies the candidate targets, and on Lights Out the solve of Corollary 5.7 selects which of them to issue. The precedence and patience filter drops the blocked ones, and the greedy rule picks one target for a single policy query. Theorem 5.4 permits this rule, since its bound holds for every schedule. Without a factor map GC-FACT returns the parent planner, and with $K = 1$ it queries the backbone on the raw goal (Proposition 5.8).

On manipulation T_s is the same setting as the threshold T of GC-FACT, and GC-FACT never calls the parent planner there. The backbone hyperparameters are the reference OGBench values for GCIQL and HIQL and the published values for OTA. At setup GC-FACT draws 4000 random dataset states and keeps 256 of them as anchor states, which is the setup time of Table 12. Only the optional value-column repair and evacuation read the anchors, so they play no role in the reported configuration. Each step scores at most eight candidate targets in one value query of fixed shape, so that the compiled value function is reused across steps. Shorter candidate lists are padded, and a padded entry is never chosen. Longer lists occur only on Lights Out, where the first eight entries are scored. For the $\mathbb{F}_2$ plan these are the presses with the lowest cell index. The plan is recomputed after every press, so on the invertible grids this cap changes the order of the presses but not the plan.

F ADDITIONAL RESULTS

This appendix holds the evidence that supports Section 6 but does not fit in the main text. Each entry is tied to a statement of Section 5.

The retrieval baseline. Table 9 adds GC-TTT (Bagatella et al., 2026) on GCIQL and HIQL for the eight manipulation environments other than `puzzle-4x4`. The original rule of Bagatella et al. (2026) scores a retained sub-trajectory by $\widehat{V} = \sum_{h=1}^{H-1} \gamma^{h-1} r(s_h, g) + \gamma^{H-1} V(s_H, g)$. It keeps those above a fixed percentile. We run the critic-based variant of the released implementation with the fixed hyperparameters that its authors report for GCIQL. Every 100 environment steps the policy is reset to its pre-trained weights. A dataset state counts as relevant when its value-derived distance from the current state is below 20 steps, which corresponds to a value threshold of -18 at $\gamma = 0.99$. The method keeps the ten trajectories whose best state, taken at least ten steps after their first relevant state, lies closest to g under $\hat{d}$. The backbone's own update then takes 100 steps with learning rate 3×10^{-4} on their transitions, with the actor goal set to g. Once no retained state satisfies g, this ranking depends on the backbone value at g alone, which is the situation of the second clause of Proposition 5.2. GC-TTT stays within 3.0 points of the frozen backbone on 14 of its 16 pairs. Its

Table 8: GC-FACT configuration, shared by every environment and every backbone. The settings of the parent planner follow the environment and are given in the Hyperparameters paragraph of Appendix E.

Hyperparameter	**Value**
Reported configuration	
Distance mode	value-derived, Equation (1)
Threshold T for aiming at the raw goal	48
Candidate targets scored per step	at most 8
Patience and cooldown, environment steps	40 and 60
Schedule	greedy on $\hat{d}$
Precedence filter	on
Press-basis solver	exact over $\mathbb{F}_2$
Body factor of a target set	taken from the current state
Optional modules, disabled unless stated	
Group symmetrisation	off
Value-column repair, rank and anchors	off, 2 and 256
Unit blending, units and sharpness λ	off, 2 and 8.0
Evacuation of a blocking cube	off
Last-layer test-time training, steps	off, 50 if enabled
Evaluation protocol	
Rollouts per evaluation goal	50
Evaluation goals per environment	5
Training seeds	up to 6

failure thus differs from that of TTGS, which subtracts. Both exceptions involve GCIQL. GC-TTT gains 5.0 points on `puzzle-3x3` and loses 22.6 on `cube-single`. There the goal is covered by the data, and fine-tuning on ten retrieved trajectories degrades a policy that already succeeds in 70 percent of the episodes.

Table 9: **The retrieval baseline on OGBench manipulation, success rate (%).** Same protocol and marking as Table 1, whose frozen and +GC-FACT columns are repeated. GC-TTT fine-tunes the backbone on retrieved sub-trajectories. The second clause of Proposition 5.2 expects this selection to be uninformative when the retained data never reach the goal, the limit that the data approach. `puzzle-4x4` was not run with GC-TTT.

	GCIQL backbone			**HIQL backbone**		
Environment	frozen	+GC-TTT	+GC-FACT	frozen	+GC-TTT	+GC-FACT
`cube-single`†	70.3 ±4.2	47.7 ±2.7	66.9 ±9.8	13.1 ±3.1	10.1 ±1.5	**13.7** ±1.7
`cube-double`	40.3 ±2.9	37.3 ±5.0	**56.2** ±2.8	6.2 ±0.5	5.3 ±0.9	**11.7** ±2.4
`cube-triple`	2.1 ±0.9	2.9 ±0.9	**6.3** ±2.3	4.0 ±1.6	3.9 ±1.4	**7.9** ±3.0
`cube-quadruple`	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.1 ±0.2	0.2 ±0.4	0.1 ±0.2
`scene`	52.2 ±5.0	49.7 ±3.1	**56.9** ±2.4	34.3 ±3.4	35.8 ±0.8	**53.9** ±4.0
`puzzle-3x3`	94.7 ±2.6	**99.7** ±0.4	**99.4** ±0.5	12.2 ±1.5	14.0 ±4.5	**70.8** ±2.7
`puzzle-4x5`	12.9 ±2.3	12.3 ±3.4	**79.1** ±1.9	3.3 ±1.8	3.2 ±1.0	**99.9** ±0.2
`puzzle-4x6`	11.3 ±2.6	10.5 ±2.3	**89.3** ±4.1	3.2 ±1.6	1.6 ±1.4	**100.0** ±0.0

Component ablation. Table 10 changes one mechanism at a time on OTA. It uses `cube-triple` and `scene`, the two environments where the basis is actionable without a change of coordinates. No row moves the outcome by more than the spread of the reported configuration, which is 2.1 points on `cube-triple` and 7.4 on `scene`. The first block removes or alters one component. Dropping the precedence filter costs 0.1 and 2.1 points, and dropping the patience window costs 0.0 and 5.2. Filling the free body factor from the goal instead of the current state costs 0.4 and 2.8. The value-derived cost of a locked unit already contains the cost of releasing its lock (Figure 10), which is consistent with the small effect of the explicit filter. The filter and the patience window thus act as safeguards rather

Table 10: **Component ablation.** OTA backbone, success rate (%), mean ± standard deviation over up to six training seeds, one flag changed per row. Parentheses give the change from the reported configuration, computed from unrounded means. The first block removes a component of the reported configuration, and the second enables an optional module. The evacuation row comes from a single run. The frozen and reported rows come from the ablation runs and differ from Table 1 by at most 1.2 points. No row changes the outcome by more than the spread of the reported configuration.

Variant	`cube-triple`	`scene`
frozen backbone	1.4 ±0.9	21.3 ±5.5
GC-FACT, reported configuration	4.2 ±2.1	49.0 ±7.4
− precedence filter	4.2 ±2.0 (−0.1)	47.0 ±4.8 (−2.1)
− patience	4.2 ±1.9 (+0.0)	43.8 ±3.7 (−5.2)
body factor from g	3.8 ±2.6 (−0.4)	46.2 ±6.1 (−2.8)
+ symmetrisation	4.5 ±2.6 (+0.2)	48.6 ±6.1 (−0.4)
+ value-column repair	4.2 ±2.0 (−0.1)	48.6 ±7.0 (−0.4)
+ unit blending	4.1 ±2.0 (−0.2)	47.2 ±4.5 (−1.8)
+ evacuation	3.6 (−0.6)	46.0 (−3.0)
+ last-layer training	4.1 ±2.0 (−0.2)	48.7 ±5.4 (−0.3)

than as the source of the gain. The second block enables the five optional modules of Appendix E, which change the outcome by -3.0 to $+0.2$ points. The reported configuration therefore disables all five, and the released implementation keeps them as switches.

Navigation parity. Table 11 reports the nine state-based navigation environments. Locomotion observations carry no factor map, so Proposition 5.8 applies and GC-FACT runs the same code path as TTGS. The two right columns of each block are therefore the same computation. They agree within 3.6 points everywhere and within the seed spread on 17 of the 18 pairs. The remaining pair, GCIQL on `antmaze-medium-navigate`, differs by 2.2 points. Differences of this size lie within the run-to-run variation of the evaluation, since separate evaluations of the same frozen checkpoint already differ by several points (Figure 11). We include the table to verify Proposition 5.8 and claim nothing from it.

Wall-clock cost. Table 12 gives the timing that Section 6.3 summarises.

Table 11: **Navigation parity, success rate (%).** Locomotion has no factor map, so GC-FACT delegates every call to its parent planner, and the last two columns of each block run the same code. Their differences, at most 3.6 points, lie within the run-to-run variation of the evaluation. The table verifies Proposition 5.8 and makes no performance claim.

	GCIQL backbone			**HIQL backbone**		
Environment	frozen	+TTGS	+GC-FACT	frozen	+TTGS	+GC-FACT
`pointmaze-medium-navigate`	53.4 ±2.5	88.6 ±3.7	88.6 ±3.8	70.8 ±2.2	89.6 ±4.3	89.6 ±5.0
`pointmaze-large-navigate`	35.6 ±2.8	86.0 ±10.2	86.0 ±9.1	40.0 ±0.0	86.8 ±2.7	86.8 ±2.1
`pointmaze-giant-navigate`	0.0 ±0.0	44.8 ±6.8	44.8 ±6.8	43.6 ±10.8	70.9 ±4.1	70.9 ±4.1
`antmaze-medium-navigate`	76.0 ±2.2	83.4 ±1.4	81.2 ±1.1	96.4 ±2.3	94.0±1.9	94.0 ±1.5
`antmaze-large-navigate`	34.7 ±3.0	57.6 ±4.5	59.2 ±2.8	91.5 ±1.9	93.6 ±1.6	93.8 ±0.5
`antmaze-giant-navigate`	0.0 ±0.0	2.4 ±1.1	3.4 ±0.3	66.4 ±5.8	69.0 ±0.4	69.5 ±3.8
`humanoidmaze-medium-navigate`	32.9 ±2.5	60.0 ±4.0	56.4 ±2.8	94.6 ±0.8	94.0 ±3.6	94.0 ±5.3
`humanoidmaze-large-navigate`	1.8 ±0.8	11.8 ±3.1	9.2 ±5.1	48.2 ±4.7	79.0 ±3.4	81.2 ±2.8
`humanoidmaze-giant-navigate`	0.1 ±0.2	0.2 ±0.3	0.4 ±0.0	12.0 ±3.8	86.0 ±4.6	87.4 ±2.8

Table 12: **Wall-clock cost on `cube-triple` on the same hardware.** Mean ± standard deviation over the runs on this environment. Setup is the time before the first rollout. It covers subsampling and graph construction for TTGS, and for GC-FACT only the draw of anchor states described in Appendix E, since GC-FACT builds no graph on manipulation. Per step is the time of choosing the goal input and querying the policy per environment step. The last column is the wall clock of the whole job, which also restores the model and evaluates the frozen backbone in the same process. The frozen backbone runs without the planner loop and is not instrumented.

Method	Setup (s)	Per step (ms)	Five-goal job (s)
frozen backbone	none	n/a	n/a
+ TTGS	65.6 ±10.5	16.1 ±0.4	6646 ±159
+ GC-FACT	0.1 ±0.1	13.5 ±0.7	5035 ±31

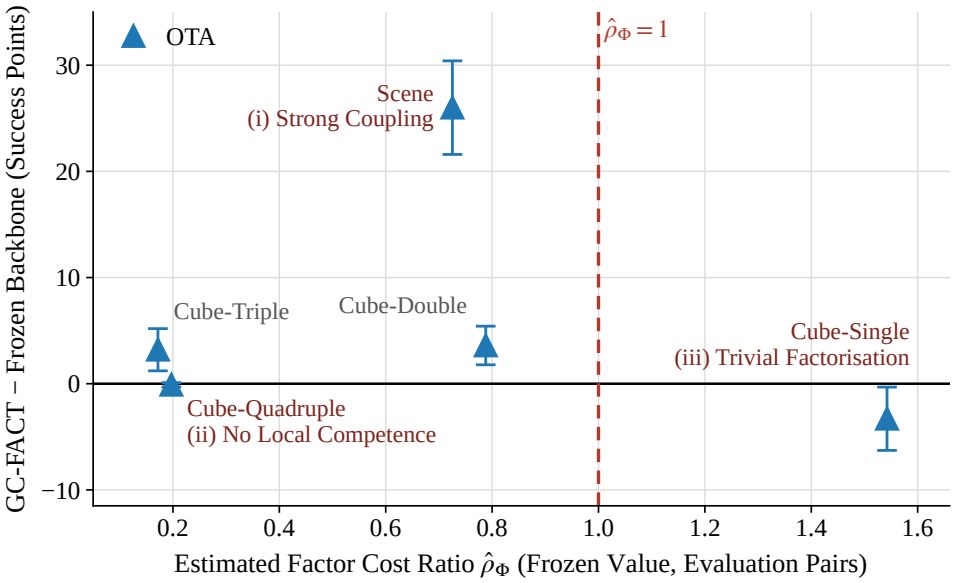

Figure 8: **The three regimes of Proposition 5.9 along the factor cost ratio.** OTA backbone. The horizontal axis is the estimate $\hat{\rho}_\Phi$, which is Equation (7) with $\hat{d}$ in place of d^*, computed from the frozen value of one checkpoint and averaged over the five evaluation pairs. The vertical axis is the gain of GC-FACT over the frozen evaluation of the same run, as mean ± standard deviation over six seeds. This paired estimate differs from the difference of means in Table 1 by at most 2.5 points. Every environment with $\hat{\rho}_\Phi < 1$ gains, except `cube-quadruple`, which stays at zero within its spread. The size of the gain is not ordered by $\hat{\rho}_\Phi$. The strongly coupled `scene`, case (i), gains most, while `cube-quadruple`, case (ii), gains nothing at an estimated ratio of 0.20. For `cube-single`, case (iii), the exact ratio is one, since $g^{[1]}(s) = \mathcal{G}(g)$ (Proposition 5.8). Its estimate exceeds one only because the target query copies the arm block from s, a query that GC-FACT never issues when $K = 1$.

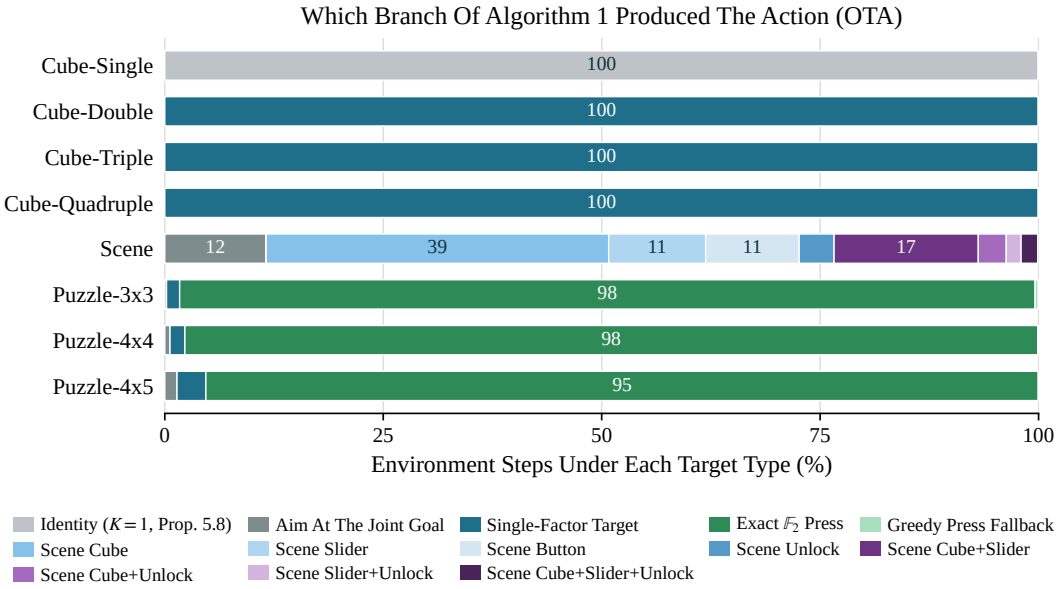

Figure 9: **Which branch of Algorithm 1 produced each action.** OTA backbone, fraction of environment steps per target type, pooled over seeds. `cube-single` spends every step on the identity branch of Proposition 5.8, and the other cube tasks spend theirs on single-factor targets. On the puzzles, exact $\mathbb{F}_2$ press targets produce 97.9, 97.6 and 95.3 percent of the steps on 3×3, 4×4 and 4×5. The rest are the aim at the raw goal and the two fallbacks of Appendix E: greedy presses, and single-cell targets shown as single-factor targets, which take 1.5 to 3.3 percent of the steps. On `scene`, 11.6 percent of the steps aim at the raw goal. The other steps are labelled by the unit types offered to the greedy rule, and an unlock target is among them in 10.8 percent of the steps. `puzzle-4x6` is omitted.

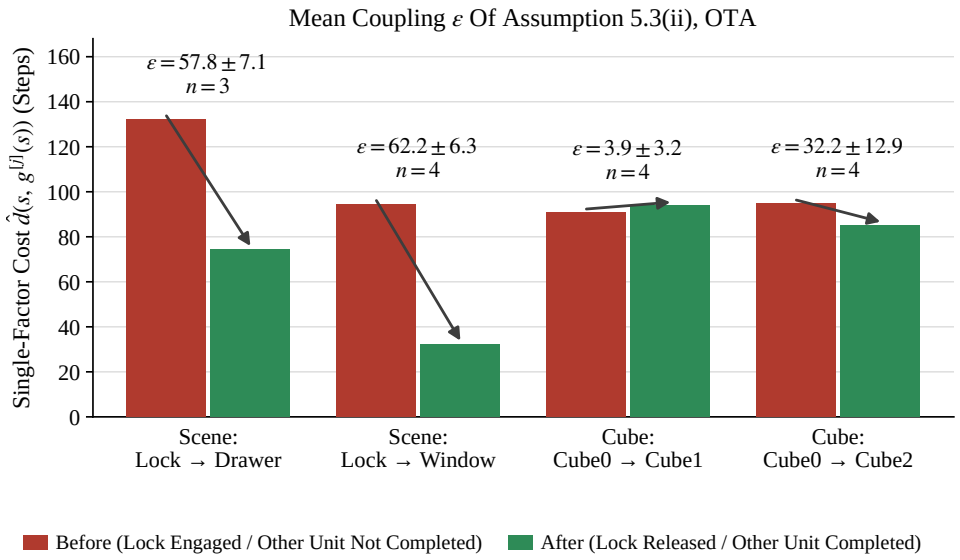

Figure 10: **The mean coupling of Assumption 5.3(ii), measured with the OTA value.** Each pair of bars gives the value-derived cost $\hat{d}(s, g^{[j]}(s))$ of a single-factor target before and after a lock is released or another unit is completed. Costs are averaged over the probed evaluation goals, whose number appears above each pair. ε is the mean absolute change, which lower-bounds the worst-case constant of Assumption 5.3(ii), and each arrow points from the cost before the change to the cost after it. Releasing a lock lowers the drawer and window costs on every probed goal, by 57.8 and 62.2 steps on average. This change is of the order of a single-factor cost and makes Equation (6) vacuous on `scene`. On `cube-triple` the change varies in sign across goals, so ε exceeds the difference of the bar heights. This pattern points to estimation error in $\hat{d}$ rather than to a precondition.

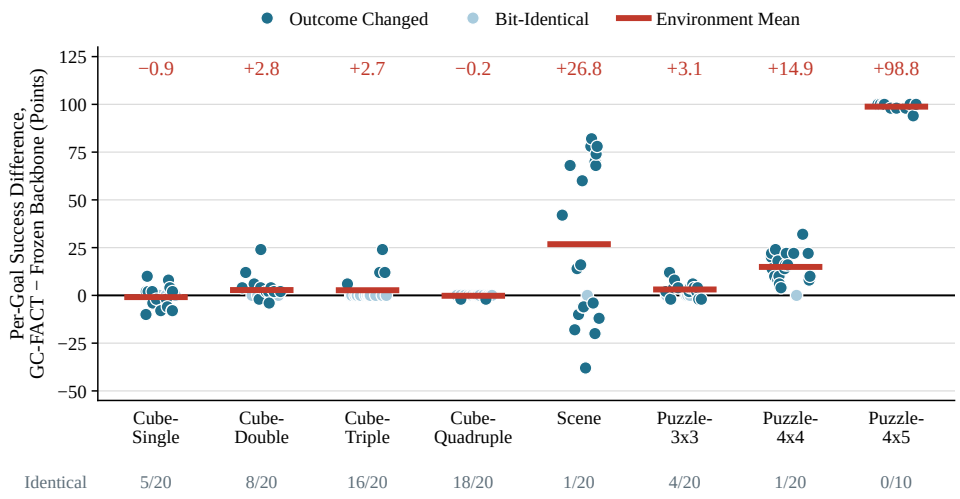

Figure 11: **Every evaluation goal of every seed.** OTA backbone, difference in success rate between GC-FACT and the frozen backbone for each goal and seed. The red bar and the number above each column give the environment mean. Light points mark pairs whose two success rates coincide, and the bottom row counts them out of all pairs of the environment. `puzzle-4x5` separates cleanly, with a mean of 98.8 points over its 10 pairs, and `puzzle-4x4` gains on 19 of its 20 pairs. `scene` spreads widest, from a loss of 38 points to a gain of 82. On `cube-single` GC-FACT is the identity, yet only 5 of the 20 pairs coincide, and the others spread over ± 10 points. The frozen backbone and GC-FACT are evaluated in separate runs, so this spread is the evaluation noise against which the other columns should be read. `puzzle-4x6` is omitted.

G QUALITATIVE ROLLOUTS IN OGBENCH

Figures 12 and 13 show rollouts of the OTA backbone in the OGBench simulator. For each environment we take the single video episode that the evaluation code renders for task 1 with training seed 0, and we show it without selection. Each frame stacks the goal above the current scene. The rows of a figure share the checkpoint and the task, and the columns are evenly spaced frames from the first to the last step of the episode. An episode ends when the agent reaches the goal or the step limit. The frames illustrate one goal and do not replace the statistics of Tables 1 and 2 or the step counts of Figure 9.

On `cube-triple`, TTGS hands the policy dataset states as waypoints, while GC-FACT hands it one single-factor target at a time (Section 4.1). All three agents reach the step limit without the goal configuration. A single failed episode is the expected outcome here, since no OTA agent exceeds 4.3 percent on this task (Table 1).

On `puzzle-4x5`, GC-FACT asks for the presses of the $\mathbb{F}_2$ plan one at a time (Corollary 5.7), and it sets every light to its goal colour after about 440 steps. The frozen backbone and TTGS, which routes the policy through dataset states, still leave mismatched lights when their episodes reach the step limit. The episode thus reproduces the gap of Table 1, where GC-FACT succeeds in 99.2 percent of the episodes and neither of the other two agents exceeds 0.8 percent.

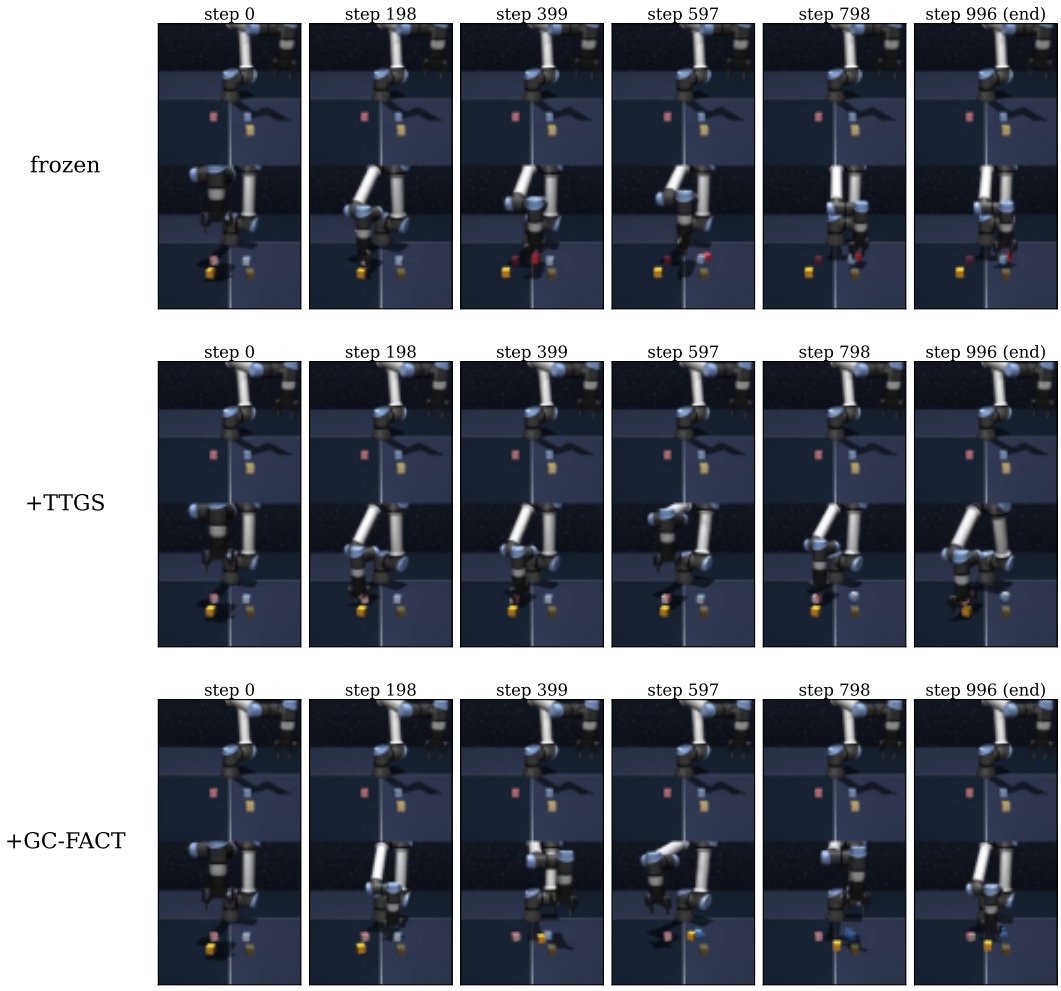

Figure 12: **Qualitative rollouts on `cube-triple`.** OTA backbone, training seed 0, the rendered episode of task 1, shown without selection. Each frame shows the goal on top and the current scene below. Rows show the frozen backbone, TTGS and GC-FACT on the same checkpoint and task. Columns are evenly spaced frames, and the last column is the final step of the episode. Frames are recorded every third step, so the step labels are approximate.

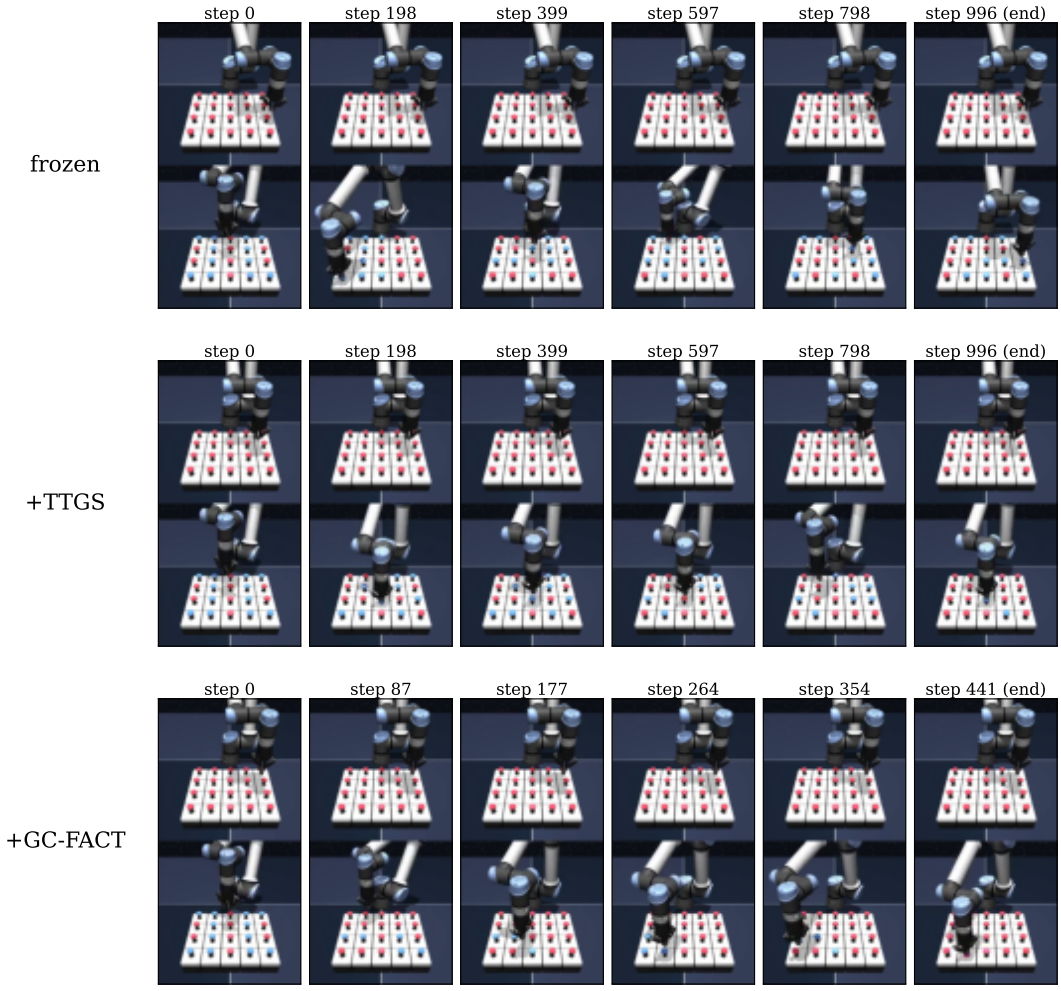

Figure 13: **Qualitative rollouts on `puzzle-4x5`.** Same protocol and layout as Figure 12. GC-FACT ends its episode at the goal after about 440 steps, while the frozen backbone and TTGS reach the step limit.