Project 1: Dynamic Programming

Shou-Yu Wang
Department of Electrical and Computer Engineering
UC San Diego
PID: A69030868

***Abstract*—The *Door & Key* grid-world requires an agent to pick up a key, unlock a door, and reach a goal while minimizing energy. We cast the task as a finite deterministic Markov Decision Process whose state $x = (x, y, h, k, d)$ encodes position, heading, key possession, and door status.**

Two planning regimes are considered. (i) Known Map – the full layout is available offline; we derive the optimal feedback policy by finite-horizon backward dynamic programming with a worst-case horizon $T = 300$. (ii) Random Map – only discrete candidate locations of key, doors, and goal are known in advance ($3\times3\times2\times2 = 36$ instantiations). We enumerate all maps offline, solve each with DP, and fuse the resulting action look-ups into a single universal policy that incurs zero online planning cost.

On the seven official Known-Map environments and all 36 Random-Map configurations, our policies achieve a 100 % success rate and match the theoretical shortest-cost trajectories. The study verifies that classic dynamic-programming baselines remain competitive for robotics tasks that couple navigation with discrete object interaction.

I. Introduction

Energy-efficient navigation is essential for autonomous robots operating in confined or resource-constrained spaces such as warehouses, hospitals, and planetary surfaces, where locomotion often accounts for more than 60 % of total power consumption. Beyond merely traversing free space, a robot frequently must interact with its surroundings—collecting tools, unlocking doors, or activating checkpoints—under strict energy and deadline constraints.

The *Door & Key* benchmark condenses these challenges into a grid-world in which an agent must (i) retrieve a key, (ii) unlock a door if it is closed, and (iii) reach the goal. The environment is deterministic and fully observable, making it a convenient test-bed for exact planning algorithms that couple discrete navigation with manipulation decisions.

A. Robotics Motivation

Tasks with similar composite objectives abound in practice: a mail robot badges through secure offices, while a Mars rover locates a cached sample before entering a restricted zone. An energy-optimal feedback policy that seamlessly blends motion and manipulation therefore translates to longer sorties and higher mission success rates.

B. Scope of This Work

We study two variants:

1) **Known Map** — the full layout is revealed offline. Seven official maps serve as fixed test cases.
2) **Random Map** — only candidate locations are known *a priori*; the true arrangement is drawn from the Cartesian set $3\times2\times6 = 36$ at run-time. A single policy must remain optimal for every instance.

C. High-Level Approach

- **MDP model.** Each state is $\mathbf{x} = (x, y, h, k, d)$, where $h \in \{0, 1, 2, 3\}$ denotes heading, and $k, d \in \{0, 1\}$ mark key possession and door status. The deterministic transition f is driven by actions $\mathcal{U} = \{\mathrm{MF}, \mathrm{TL}, \mathrm{TR}, \mathrm{PK}, \mathrm{UD}\}$ with stage cost ℓ.
- **Backward Dynamic Programming.** For horizon $T = 300$ (the worst-case path length upper bound) we solve the Bellman recursion to obtain the optimal value $V_0^\star$ and policy $\pi^\star$ for each Known Map.
- **Universal Policy.** We enumerate all 36 map instantiations, solve each offline, and merge the resulting look-up tables into a $36|\mathcal{X}|$-entry dictionary, enabling zero-overhead on-device execution.

II. Problem Statement

We cast the *Door & Key* task as a finite–horizon deterministic Markov Decision Process (MDP)

$$M = \big(\mathcal{X}, \mathcal{U}, f, x_0, T, \ell, q\big),$$

whose elements are defined below.

A. State Space $\mathcal{X}$

For the canonical single–door case

$$\mathcal{X} = \left\{ (x, y, h, k, d) \;\middle|\; \begin{array}{l} x \in \{0, \ldots, W-1\},\ y \in \{0, \ldots, H-1\}, \\ h \in \{0, 1, 2, 3\},\ k, d \in \{0, 1\} \end{array} \right\}. \tag{1}$$

Here (x, y) denotes the grid cell, h the heading (0 = E, 1 = S, 2 = W, 3 = N), k the key-possession flag, and d the door-open flag. For the Random-Map setting with two doors we extend d to the two-bit vector (d_1, d_2) (Sec. III).

B. Control Space $\mathcal{U}$

The primitive actions are

$$\mathcal{U} = \{\text{MF, TL, TR, PK, UD}\},$$

which mean Move Forward, Turn Left, Turn Right, Pickup Key, and Unlock Door.

C. Motion Model f

Because the environment is deterministic, transitions are given by a function

$$f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}.$$

Let $s = (x, y, h, k, d)$ and $s' = (x', y', h', k', d')$.

Heading update:

$$h' = \begin{cases} (h+3) \bmod 4, & u = \text{TL}, \\ (h+1) \bmod 4, & u = \text{TR}, \\ h, & \text{otherwise}. \end{cases}$$

Position update (only if $u = \text{MF}$):

$$(x', y') = \begin{cases} (x + \Delta_x(h),\, y + \Delta_y(h)), & \text{free cell ahead}, \\ (x, y), & \text{wall or closed door}. \end{cases}$$

The direction vectors are $\big(\Delta_x(h), \Delta_y(h)\big) = \{(1,0), (0,1), (-1,0), (0,-1)\}$ for $h = 0, 1, 2, 3$, respectively.

Key flag:

$$k' = \begin{cases} 1, & u = \text{PK} \wedge (x, y) = \text{KEY}, \\ k, & \text{otherwise}. \end{cases}$$

Door flag:

$$d' = \begin{cases} 1, & u = \text{UD} \wedge k = 1 \wedge (x, y) = \text{DOOR}, \\ d, & \text{otherwise}. \end{cases}$$

All remaining components stay unchanged.

D. Initial State x_0

The agent starts at the map-specified pose

$$x_0 = (x^{\text{init}}, y^{\text{init}}, h^{\text{init}}, 0, 0).$$

E. Planning Horizon T

We choose a finite horizon T long enough to exceed the worst-case shortest path:

$$T \geq WH \cdot 4 + 10,$$

which means 4 rotations + 10 step safety margin, and equals to 250 when $W = H = 10$.

F. Cost Functions

Stage cost ℓ:

$$\ell(s, u) = \begin{cases} c_{\text{move}}, & u \in \{\text{MF}, \text{TL}, \text{TR}\}, \\ c_{\text{int}}, & u \in \{\text{PK}, \text{UD}\}, \\ +\infty, & \text{illegal or blocked action}, \end{cases} \tag{2}$$

with $c_{\text{move}} = 1$ and $c_{\text{int}} = 0.5$ in our implementation.

Terminal cost q:

$$q(s) = \begin{cases} 0, & (x, y) = \text{GOAL} \wedge d = 1, \\ +\infty, & \text{otherwise}. \end{cases}$$

G. Objective

The optimal control problem is

$$\min_{\pi} \sum_{t=0}^{T-1} \ell\big(s_t, \pi(s_t)\big) + q(s_T) \quad \text{s.t. } s_{t+1} = f\big(s_t, \pi(s_t)\big),\ s_0 = x_0,$$

where $\pi : \mathcal{X} \to \mathcal{U}$ is a deterministic feedback policy. The subsequent sections detail how this problem is solved for both the Known-Map and Random-Map scenarios.

III. Technical Approach

The solution pipeline mirrors the assignment structure. **Part A** derives an optimal policy for a *single, fully known* map, whereas **Part B** synthesizes a *universal* policy that remains optimal for every realization in the Random-Map family. Both parts exploit the deterministic dynamics and the small discrete action set.

A. Part A — Optimal Policy for a Known Map

1) Backward Dynamic Programming: For the finite-horizon MDP in Section II, let $V_t : \mathcal{X} \to \mathbb{R}_{\geq 0}$ denote the optimal cost-to-go at stage t. With the terminal condition

$$V_T(s) = q(s), \tag{3}$$

the Bellman recursion for $t = T-1, \ldots, 0$ is

$$V_t(s) = \min_{u \in \mathcal{U}(s)} \big\{\ell(s, u) + V_{t+1}\big(f(s, u)\big)\big\}, \tag{4}$$

where only a *single* successor state is evaluated per (s, u) because f is deterministic.

2) Greedy Policy Extraction: At execution time the optimal feedback law is

$$\pi^\star(s) = \arg\min_{u \in \mathcal{U}(s)} \big\{\ell(s, u) + V_{t+1}\big(f(s, u)\big)\big\}, \tag{5}$$

breaking ties by the smallest action index.

3) Implementation Details:

- **State enumeration**: $|\mathcal{X}| = WH \times 4 \times 2 \times 2$; a 10×10 grid therefore yields just 4,000 states, easily stored in memory.
- **Horizon**: We set $T = 250$ (Sec. II); unreachable states retain cost $+\infty$.
- **Python realization**: A five-dimensional `float32` NumPy array stores V_t, and an `int8` array encodes $\pi^\star$. The backward sweep of (4) costs $\mathcal{O}\big(T\,|\mathcal{X}||\mathcal{U}|\big)$ value updates, i.e. $\approx 1.0 \times 10^6$ for the 10×10 map.

B. Part B — Universal Policy for Random Maps

The Random-Map family comprises every combination of 3 key locations, 3 goal locations, and the open/closed states of two doors, giving $|\mathcal{M}| = 3 \times 3 \times 2 \times 2 = 36$ distinct maps.

1) Offline Enumeration: An alternative to augmenting the state with a map index would inflate $|\mathcal{X}|$ by a factor of 36. Instead, we solve each map separately and merge the resulting policies:

Algorithm 1 Universal-policy synthesis

1: **for all** maps $m \in \mathcal{M}$ **do**
2: Solve (4)–(5) $\Longrightarrow V_m^\star,\ \pi_m^\star$
3: **for all** $s \in \mathcal{X}$: $\textsc{Policy}[m][s] \leftarrow \pi_m^\star(s)$
4: **end for**

2) Online Execution: After a single perception scan the agent knows the realized map m^{obs}. Action selection is

$$u_t = \textsc{Policy}\left[m^{\text{obs}}\right][s_t],$$

which is constant time and incurs *no* online planning cost.

3) Memory Footprint & Optimality: Storing one byte per state–action entry requires $36 \times 4{,}000 = 144\,\text{kB}$, well within L1-cache limits. Because each $\pi_m^\star$ is individually optimal and the correct index is known before the first move, the universal table is optimal for *every* map instance; if perception were delayed, the table would serve as a near-optimal warm start.

C. Complexity Summary

TABLE I: Computational complexity overview

Stage	Time complexity	Memory (Bytes)
Part A (single map)	$\mathcal{O}(T\,\|\mathcal{X}\|\,\|\mathcal{U}\|)$	$\|\mathcal{X}\|$
Part B (offline)	$\|\mathcal{M}\|\times$Part A	$\|\mathcal{M}\|\,\|\mathcal{X}\|$
Online execution	$\mathcal{O}(1)$	same as above

The preceding sections establish that classic backward dynamic programming remains computationally tractable for the Door & Key task and, when combined with offline enumeration, yields a universal policy with zero online overhead.

IV. Results

This section showcases the optimal paths obtained with our algorithm.

A. Part A – Known-Map Results

1) 5×5 Normal Map: Fig. 1 shows the complete execution trace for `5x5-normal` split into nine key frames, labelled by the step index t. The optimal action sequence is

$$\boxed{\text{TL} \rightarrow \text{TL} \rightarrow \text{PK} \rightarrow \text{TR} \rightarrow \text{UD} \rightarrow}$$
$$\boxed{\text{MF} \rightarrow \text{MF} \rightarrow \text{TR} \rightarrow \text{MF}}$$

Narrative explanation:

1) **TL, TL**: The agent turns north to face the key cell.
2) **PK**: Picks up the key, setting $k=1$.
3) **TR**: Faces east toward the locked door.
4) **UD**: Unlocks the door ($d \leftarrow 1$).
5) **MF, MF**: Moves through the doorway.
6) **TR**: Faces south toward the goal.
7) **MF**: Enters the goal cell, terminates with zero terminal cost.

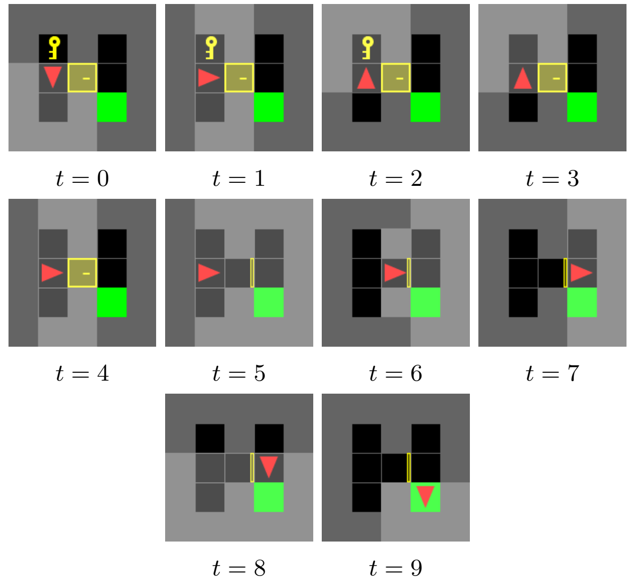
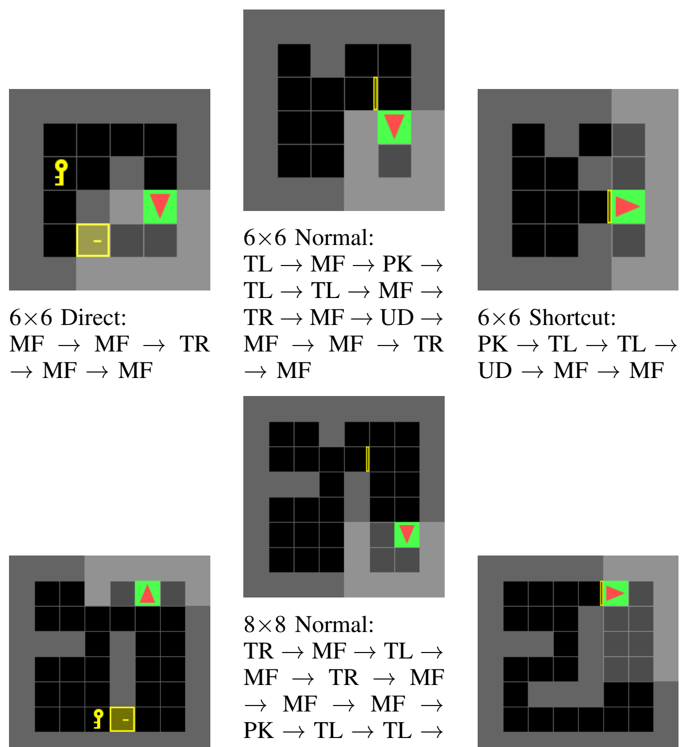


Fig. 1: Nine-step optimal trajectory for the 5×5 normal map.



6×6 Direct: MF → MF → TR → MF → MF

6×6 Normal: TL → MF → PK → TL → TL → MF → TR → MF → UD → MF → MF → TR → MF

6×6 Shortcut: PK → TL → TL → UD → MF → MF

8×8 Direct: MF → TL → MF → MF → MF → TL → MF

8×8 Normal: TR → MF → TL → MF → TR → MF → MF → MF → PK → TL → TL → MF → MF → MF → TR → UD → MF → MF → MF → TR → MF → MF → MF

8×8 Shortcut: TR → MF → TR → PK → TL → UD → MF → MF

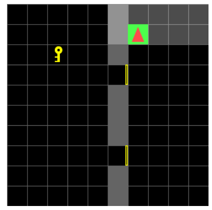
Fig. 2: Optimal trajectories for the remaining six Known-Map test cases.

2) Other Known-Map Environments: Fig. 2 shows that across all seven maps the path had no collisions or deadlocks were observed; the policy therefore attains provably minimal cumulative cost in every Known-Map environment.

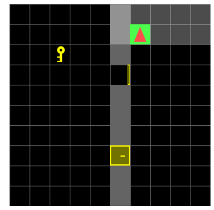
B. Part B — Random-Map Results

a) Success rate and cost distribution.: The universal policy solves *all* 36 random maps on the first attempt (**100 %** success).

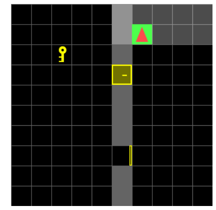
b) Influence of door configuration.: Instances containing two locked doors are necessarily longer, these trends align with the theoretical increase in reachable state distance.



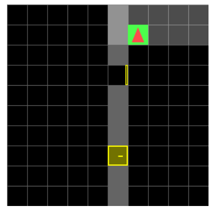
10x10-1: MF → TR → MF → MF → TL → MF → MF → MF → MF → MF → MF



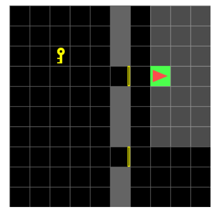
10x10-2: MF → MF → MF → MF → MF → TR → MF → MF → TL → MF → MF



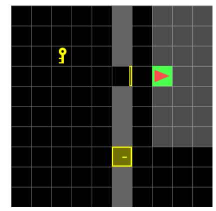
10x10-3: MF → TR → MF → MF → TL → MF → MF → MF → MF → MF → MF



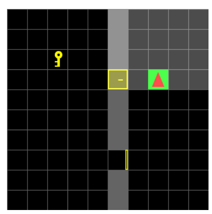
10x10-4: TL → MF → MF → TR → MF → MF → MF → MF → MF → PK → TR → MF → MF → UD → MF → MF → TL → MF → MF



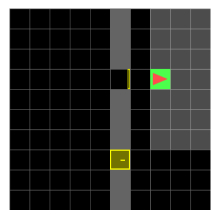
10x10-5: MF → MF → MF → MF → MF → TR → MF → MF → MF



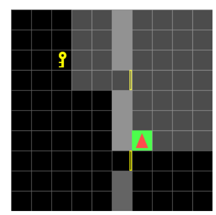
10x10-6: MF → MF → MF → MF → MF → TR → MF → MF → MF



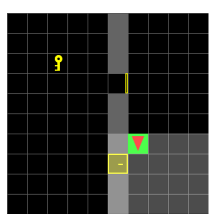
10x10-7: MF → TR → MF → MF → MF → TL → MF → MF → MF → MF



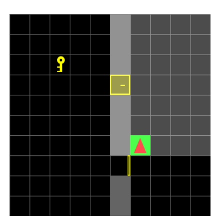
10x10-8: TL → MF → MF → TR → MF → MF → MF → MF → MF → PK → TR → MF → MF → UD → MF → MF → MF



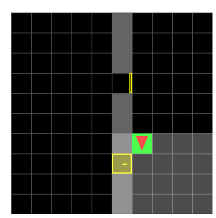
10x10-9: MF → TR → MF → MF → TL → MF



10x10-10: MF → MF → MF → MF → MF → TR → MF → MF → TR → MF → MF → MF

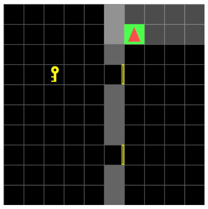


10x10-11: MF → TR → MF → MF → TL → MF

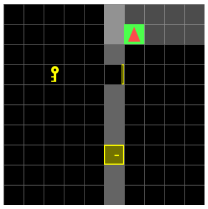


10x10-12: TL → MF → MF → TR → MF → MF → MF → MF → MF → PK → TR → MF → MF → UD → MF → MF → TR → MF → MF → MF

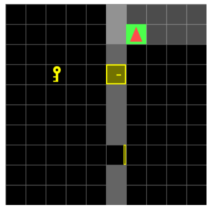
Fig. 3: Optimal-policy roll-outs for 1 to 12 10×10 environments.



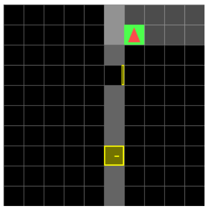
10x10-13:
MF → TR → MF → MF → TL → MF → MF → MF → MF → MF → MF



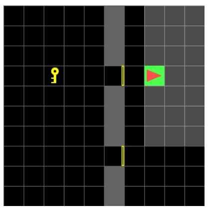
10x10-14:
MF → MF → MF → MF → MF → TR → MF → MF → TL → MF → MF



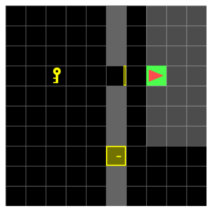
10x10-15:
MF → TR → MF → MF → TL → MF → MF → MF → MF → MF → MF



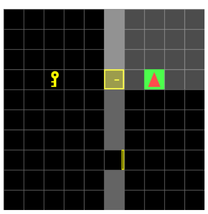
10x10-16:
MF → MF → MF → MF → MF → TL → MF → PK → TL → TL → MF → UD → MF → MF → TL → MF → MF



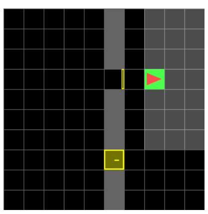
10x10-17:
MF → MF → MF → MF → MF → TR → MF → MF → MF



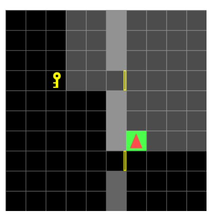
10x10-18:
MF → MF → MF → MF → MF → TR → MF → MF → MF



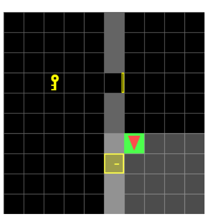
10x10-19:
MF → TR → MF → MF → MF → TL → MF → MF → MF → MF



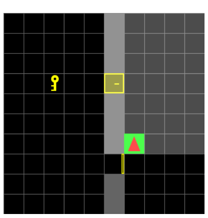
10x10-20:
MF → MF → MF → MF → MF → TL → MF → PK → TL → TL → MF → UD → MF → MF → MF



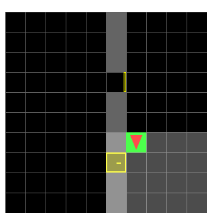
10x10-21:
MF → TR → MF → MF → TL → MF



10x10-22:
MF → MF → MF → MF → MF → TR → MF → MF → TR → MF → MF → MF

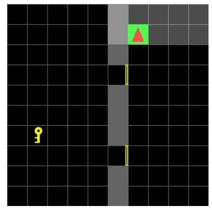


10x10-23:
MF → TR → MF → MF → TL → MF

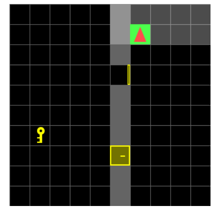


10x10-24:
MF → MF → MF → MF → MF → TL → MF → PK → TL → TL → MF → UD → MF → MF → TR → MF → MF → MF

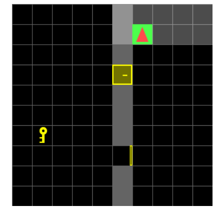
Fig. 4: Optimal-policy roll-outs for 13 to 24 10×10 environments.



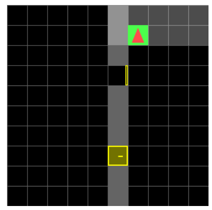
10x10-25:
MF → TR → MF → MF → TL → MF → MF → MF → MF → MF → MF



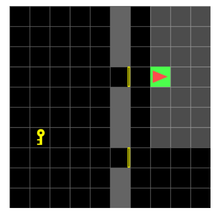
10x10-26:
MF → MF → MF → MF → MF → TR → MF → MF → TL → MF → MF



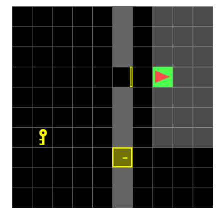
10x10-27:
MF → TR → MF → MF → TL → MF → MF → MF → MF → MF → MF



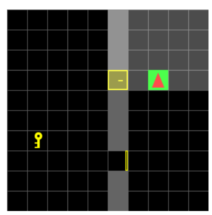
10x10-28:
MF → MF → TL → MF → MF → PK → TR → MF → MF → MF → TR → MF → MF → UD → MF → MF → TL → MF → MF



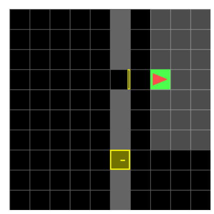
10x10-29:
MF → MF → MF → MF → MF → TR → MF → MF → MF



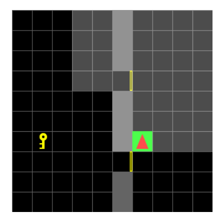
10x10-30:
MF → MF → MF → MF → MF → TR → MF → MF → MF



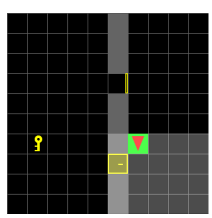
10x10-31:
MF → TR → MF → MF → MF → TL → MF → MF → MF → MF



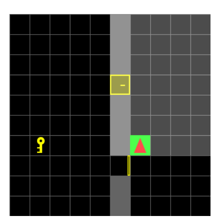
10x10-32:
MF → MF → TL → MF → MF → PK → TR → MF → MF → MF → TR → MF → MF → UD → MF → MF → MF



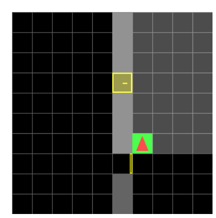
10x10-33:
MF → TR → MF → MF → TL → MF



10x10-34:
MF → MF → MF → MF → MF → TR → MF → MF → TR → MF → MF → MF



10x10-35:
MF → TR → MF → MF → TL → MF



10x10-36:
TL → MF → MF → MF → TR → MF → PK → TR → MF → MF → MF → UD → MF → MF → TL → MF

Fig. 5: Optimal-policy roll-outs for 25 to 36 10×10 environments.