Project 1: Dynamic Programming

Chen-Yeh Lin
Department of Electrical and Computer Engineering
University of California, San Diego
PID: A69036794

***Abstract*—The Door & Key problem involves guiding an agent through a grid world containing obstacles, keys, and doors to reach a goal while minimizing cumulative energy cost. This task exemplifies real-world challenges in mobile robotics, where autonomous agents must plan efficient paths and interact with elements of the environment. We cast the navigation problem as a Markov Decision Process and employ dynamic programming via value iteration to compute optimal control policies. We address two scenarios: one with fully known map layouts and another with randomized key and goal placements, deriving a single policy that generalizes across random environments. Simulation results demonstrate that our method reliably produces low-cost trajectories and adapts effectively to map variations.**

I. Introduction

The Door & Key environment presents a sequential decision-making challenge in which an agent must navigate through a maze of walls and doors, collect a key if necessary, and unlock barriers to reach a designated goal. This setup captures critical aspects of robotic autonomy in structured settings—such as warehouses or search-and-rescue operations—where robots must interact with objects and plan under resource constraints. Efficiently solving this problem is essential for deploying robots that can handle complex tasks with minimal energy consumption.

In this project, we model the navigation task as a Markov Decision Process and apply dynamic programming to derive optimal control policies. For the known map scenario, we compute distinct policies tailored to each environment instance to minimize the agent's energy cost. For the random map scenario, we develop a single robust policy that accommodates variations in key and goal locations. We validate our approach through simulation, visualizing agent trajectories and analyzing performance metrics to demonstrate its effectiveness for structured robotic planning tasks.

II. Problem Statement

We formulate the Door & Key navigation task as a deterministic MDP

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

- $\mathcal{X}$: state space
- $\mathcal{U}$: control space
- $f\colon \mathcal{X} \times \mathcal{U} \to \mathcal{X}$: motion model
- x_0: initial state
- T: planning horizon
- $\ell\colon \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{\geq 0}$: stage cost
- $q\colon \mathcal{X} \to \mathbb{R}_{\geq 0}$: terminal cost

A. Known Map Scenario

The environment layout (grid size $W \times H$, walls, key, door, goal) is fixed and known. We define:

- **State space**

$$\begin{aligned}\mathcal{X} = \{(x, y, h, k, d) \mid \\ x \in \{0, \ldots, W-1\},\ y \in \{0, \ldots, H-1\}, \\ h \in \{0, 1, 2, 3\},\ k \in \{0, 1\},\ d \in \{0, 1\}\}.\end{aligned}$$

  where (x, y) is the agent position, h its heading, k a key-possession flag, and d a door-unlocked flag.
- **Control space**

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

- **Motion model** $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$, deterministic update respecting walls, key/door rules.
- **Initial state** $x_0 = (x_0^p, y_0^p, h_0, 0, 0)$, from the environment.
- **Horizon** A finite $T = 100$ to allow sufficient steps to reach the goal.
- **Stage cost** $\ell(x, u) = 1$, per action.
- **Terminal cost**

$$q(x) = \begin{cases} 0, & (x, y) \text{ is goal}, \\ 1000, & \text{otherwise}. \end{cases}$$

B. Random Map Scenario

In the Random Map scenario, the layout is fixed except for the key position, goal position, and door states, which vary across instances:

- Grid size: 10×10 with perimeter walls.
- Vertical wall at column 5 with two doors at positions $(5, 3)$ and $(5, 7)$.
- Key randomly located in one of $\{(2, 2),\ (2, 3),\ (1, 6)\}$.
- Goal randomly located in one of $\{(6, 1),\ (7, 3),\ (6, 6)\}$.
- Agent initial pose: spawned at $(4, 8)$ facing up $(h_0 = 3)$.

We model this as an MDP

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

- **State space**

$$\mathcal{X} = \big\{(i_k, i_g, x, y, h, k, d_1, d_2) \mid i_k, i_g \in \{0,1,2\},\ x, y \in \{0,\dots,9\},\ h \in \{0,1,2,3\},\ k \in \{0,1\},\ d_1, d_2 \in \{0,1\}\big\}.$$

- **Control space**

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

- **Transition model** A deterministic function

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

that updates $(i_k, i_g, x, y, h, k, d_1, d_2)$ according to the grid geometry and interaction rules.
- **Initial state**

$$x_0 = (\, i_k^0,\, i_g^0,\, 4,\, 8,\, 3,\, 0,\, d_1^0,\, d_2^0\,),$$

where i_k^0, i_g^0 and d_1^0, d_2^0 are drawn from the random environment instance.
- **Horizon** A finite $T = 100$ to allow sufficient steps to reach the goal.
- **Stage cost**

$$\ell(x, u) = 1,$$

a positive constant energy cost for action u.
- **Terminal cost**

$$q(x) = \begin{cases} 0, & \text{if the agent is at the goal,} \\ C, & \text{otherwise,} \end{cases}$$

with $C = 1000$ to heavily penalize failure to reach the goal.

III. Technical Approach

We consider two settings for synthesizing a control policy for the Door & Key task:

- **Known Map:** The exact positions of keys, doors, and the goal are fully known at planning time, allowing standard MDP value iteration.
- **Random Map:** The map layout is unknown at planning time but each key, door, and goal can only appear at a finite set of possible locations. We compute a single universal policy over this augmented state space that can be applied to any random instance.

A. Known Map

We model the Door & Key task as a finite, deterministic MDP

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

where
$\mathcal{X} = \{(x, y, h, k, d) \mid x \in \{0,\dots,W-1\},\ y \in \{0,\dots,H-1\},\ h \in \{0,1,2,3\},\ k, d \in \{0,1\}\}$.
$\mathcal{U}(x) = \{\mathrm{MF}, \mathrm{TL}, \mathrm{TR}, \mathrm{PK}, \mathrm{UD}\}$.
$f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ Deterministic state transition.
$\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ Stage (step) cost.
$q : \mathcal{X} \to \mathbb{R}$ Terminal cost.
$x_0 \in \mathcal{X}$ Initial state.
T Finite horizon.

1) **State Enumeration.** Enumerate all $x \in \mathcal{X}$.
2) **Value Initialization.** $V(x) \leftarrow 0$ for every x.
3) **Value Iteration.**

$$V_{\mathrm{new}}(x) = \min\Big\{ q(x),\ \min_{u \in \mathcal{U}(x)} \{\ell(x,u) + \gamma\, V(f(x,u))\}\Big\},$$
$$\Delta \leftarrow \max_x \big|V_{\mathrm{new}}(x) - V(x)\big|, \quad V \leftarrow V_{\mathrm{new}}$$

until $\Delta < \theta$.
4) **Policy Extraction.**

$$\pi^*(x) = \begin{cases} \bot, & q(x) = 0, \\ \arg\min\limits_{u \in \mathcal{U}(x)} \{\ell(x,u) + \gamma\, V(f(x,u))\}, & \text{otherwise.} \end{cases}$$

5) **Rollout.** From x_0, apply $u_t = \pi^*(x_t)$ until termination, accumulating cost.

Algorithm 1 Value Iteration & Policy Extraction

Require: $\mathcal{X}, \mathcal{U}, f, \ell, q, \gamma, \theta$
for all $x \in \mathcal{X}$ **do**
  $V(x) \leftarrow 0$
end for
repeat
  $\Delta \leftarrow 0$
  for all $x \in \mathcal{X}$ **do**
    $V_{\mathrm{new}}(x) \leftarrow \min\{\, q(x),\, \min_{u \in \mathcal{U}(x)}[\ell(x,u) + \gamma\, V(f(x,u))]\}$
    $\Delta \leftarrow \max(\Delta, |V_{\mathrm{new}}(x) - V(x)|)$
  end for
  $V \leftarrow V_{\mathrm{new}}$
until $\Delta < \theta$
for all $x \in \mathcal{X}$ **do**
  if $q(x) = 0$ **then**
    $\pi^*(x) \leftarrow \bot$
  else
    $\pi^*(x) \leftarrow \arg\min_{u \in \mathcal{U}(x)}[\ell(x,u) + \gamma\, V(f(x,u))]$
  end if
end for
return V, π^*

Algorithm 2 Policy Rollout

Require: initial state x_0, policy π^*, horizon T
$t \leftarrow 0,\ total \leftarrow 0$
while $t < T$ **and** $\pi^*(x_t) \neq \perp$ **do**
  $u \leftarrow \pi^*(x_t)$
  $x_{t+1} \leftarrow f(x_t, u)$
  $J \leftarrow J + \ell(x_t, u)$
  if $q(x_{t+1}) = 0$ **then**
    break
  end if
  $t \leftarrow t + 1$
end while
return action sequence, total cost $total$

Here's the **Random Map** subsection in the exact same style as your Known Map part:

B. Random Map

We now treat the layout as unknown at planning time by embedding the environment parameters into the state:

$$\widetilde{M} = \big(\widetilde{\mathcal{X}},\ \mathcal{U},\ \tilde{f},\ \ell,\ q,\ \tilde{x}_0,\ T\big),$$

where

$\widetilde{\mathcal{X}} = \{(i_k, i_g, d_1, d_2, x, y, h, k) \mid i_k, i_g \in \{0,1,2\},\ (d_1, d_2, k) \in \{0,1\}^3,\ (x, y) \in \{0, \dots, 9\}^2,\ h \in \{0, \dots, 3\}\}$.
Here i_k, i_g index the possible key/goal locations, d_1, d_2 are door-open flags, and k is the "has key" flag.

$\mathcal{U}(s) = \{\mathrm{MF}, \mathrm{TL}, \mathrm{TR}, \mathrm{PK}, \mathrm{UD}\}$, same as Known Map.

$\tilde{f} : \widetilde{\mathcal{X}} \times \mathcal{U} \to \widetilde{\mathcal{X}}$ Deterministic transition updating both robot pose and environment flags.

$\ell : \widetilde{\mathcal{X}} \times \mathcal{U} \to \mathbb{R}$ Stage cost, identical definition.

$q : \widetilde{\mathcal{X}} \to \mathbb{R}$ Terminal cost, zero at goal index and large otherwise.

$\tilde{x}_0 \in \widetilde{\mathcal{X}}$ All combinations of $(i_k, i_g, d_1, d_2, x_0, y_0, h_0, 0)$ for the true initial pose.

T Finite horizon.

1) **State Enumeration.** Enumerate every $s \in \widetilde{\mathcal{X}}$.
2) **Value Initialization.** $V(s) \leftarrow 0$ for every s.
3) **Value Iteration.**

$$V_{\text{new}}(s) = \min\Big\{ q(s),\ \min_{u \in \mathcal{U}(s)} \{\ell(s,u) + \gamma\, V(\tilde{f}(s,u))\}\Big\},$$
$$\Delta \leftarrow \max_s \big|V_{\text{new}}(s) - V(s)\big|, \quad V \leftarrow V_{\text{new}}$$

until $\Delta < \theta$.

4) **Policy Extraction.**

$$\pi^*(s) = \begin{cases} \perp, q(s) = 0 \\ \arg\min\limits_{u \in \mathcal{U}(s)} \{\ell(s,u) + \gamma\, V(\tilde{f}(s,u))\} \end{cases}$$

5) **Rollout on Random Map.** For any sampled environment, observe (i_k, i_g, d_1, d_2), form $s_0 = (i_k, i_g, d_1, d_2, x_0, y_0, h_0, 0)$, and apply $\pi^*(s_t)$ until termination.

Algorithm 3 Value Iteration & Policy Extraction (Random Map)

Require: $\widetilde{\mathcal{X}}, \mathcal{U}, \tilde{f}, \ell, q, \gamma, \theta$
for all $s \in \widetilde{\mathcal{X}}$ **do**
  $V(s) \leftarrow 0$
end for
repeat
  $\Delta \leftarrow 0$
  for all $s \in \widetilde{\mathcal{X}}$ **do**
    $V_{\text{new}}(s) \leftarrow \min\{ q(s), \min_{u \in \mathcal{U}(s)}[\ell(s,u) + \gamma\, V(\tilde{f}(s,u))]\}$
    $\Delta \leftarrow \max(\Delta, |V_{\text{new}}(s) - V(s)|)$
  end for
  $V \leftarrow V_{\text{new}}$
until $\Delta < \theta$
for all $s \in \widetilde{\mathcal{X}}$ **do**
  if $q(s) = 0$ **then**
    $\pi^*(s) \leftarrow \perp$
  else
    $\pi^*(s) \leftarrow \arg\min_{u \in \mathcal{U}(s)}[\ell(s,u) + \gamma\, V(\tilde{f}(s,u))]$
  end if
end for
return V, π^*

Algorithm 4 Policy Rollout (Random Map)

Require: initial state s_0, policy π^*, horizon T
$t \leftarrow 0,\ J \leftarrow 0$
while $t < T$ **and** $\pi^*(s_t) \neq \perp$ **do**
  $u \leftarrow \pi^*(s_t)$
  $s_{t+1} \leftarrow \tilde{f}(s_t, u)$
  $J \leftarrow J + \ell(s_t, u)$
  if $q(s_{t+1}) = 0$ **then**
    break
  end if
  $t \leftarrow t + 1$
end while
return action sequence, total cost J

IV. Results

A. Knownmap

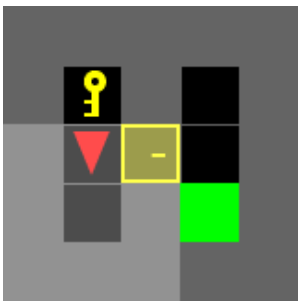


Fig. 1: Optimal trajectory in the 5×5 Normal environment. The agent (red triangle) picks up the key (yellow) and unlocks the door before reaching the goal (green square).

TABLE I: 5×5 Normal: Optimal action sequence and total cost

Action Sequence	Cost
TL, TL, PK, TR, UD, MF, MF, TR, MF	9

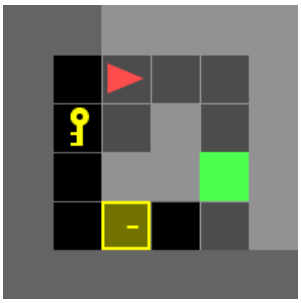


Fig. 2: Optimal trajectory in the 6×6 Direct environment. No key pickup or door unlocking is required; the agent moves directly to the goal.

TABLE II: 6×6 Direct: Optimal action sequence and total cost

Action Sequence	Cost
MF, MF, TR, MF, MF	5

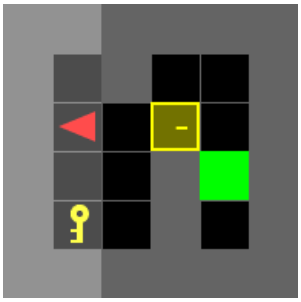


Fig. 3: Optimal trajectory in the 6×6 Normal environment. The agent first picks up the key, then unlocks the door, and finally reaches the goal.

TABLE III: 6×6 Normal: Optimal action sequence and total cost

Action Sequence	Cost
TL, MF, PK, TL, TL, MF, TR, MF, UD, MF, MF, TR, MF	13

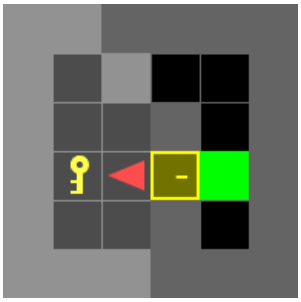


Fig. 4: Optimal trajectory in the 6×6 Shortcut environment. The agent picks up the key and unlocks the door to take the shortcut before reaching the goal.

TABLE IV: 6×6 Shortcut: Optimal action sequence and total cost

Action Sequence	Cost
PK, TL, TL, UD, MF, MF	6

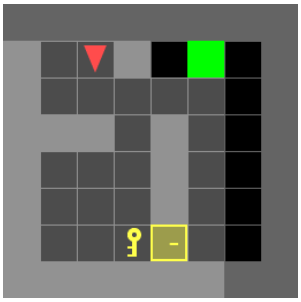


Fig. 5: Optimal trajectory in the 8×8 Direct environment. No key or door interaction is required; the agent moves directly to the goal.

TABLE V: 8×8 Direct: Optimal action sequence and total cost

Action Sequence	Cost
MF, TL, MF, MF, MF, TL, MF	7

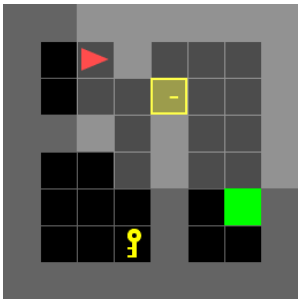


Fig. 6: Optimal trajectory in the 8×8 Normal environment. The agent must pick up the key, unlock the door, and then proceed to the goal along a longer path.

TABLE VI: 8×8 Normal: Optimal action sequence and total cost

Action Sequence	Cost
TR, MF, TL, MF, TR, MF, MF, MF, PK, TL, TL, MF, MF, MF, TR, UD, MF, MF, MF, TR, MF, MF, MF	23

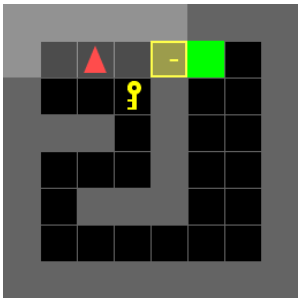
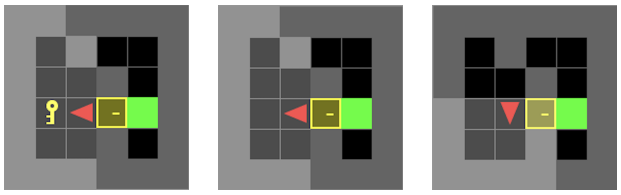


Fig. 7: Optimal trajectory in the 8×8 Shortcut environment. The agent picks up the key, unlocks the door, and uses a shortcut to reach the goal quickly.

TABLE VII: 8×8 Shortcut: Optimal action sequence and total cost

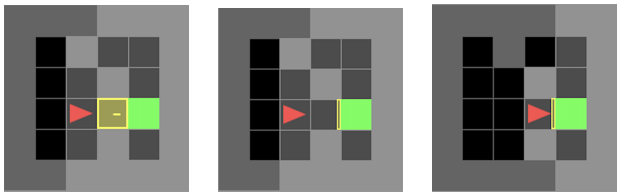
Action Sequence	Cost
TR, MF, TR, PK, TL, UD, MF, MF	8

B. Example: 6×6 Shortcut Trajectory



1 Initial view 2 Pick up Key 3 Turn Left

Fig. 8: First three steps in the 6×6 Shortcut environment: (a) initial position, (b) Pick up key, (c) pick up the key.



1 Turn to door 2 Unlock door 3 Enter corridor

Fig. 9: Next three steps in the 6×6 Shortcut environment: (d) navigate to the door, (e) unlock the door, (f) enter the corridor.

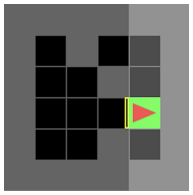
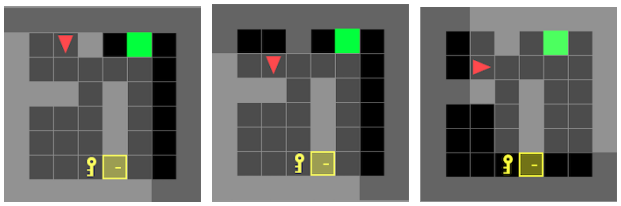
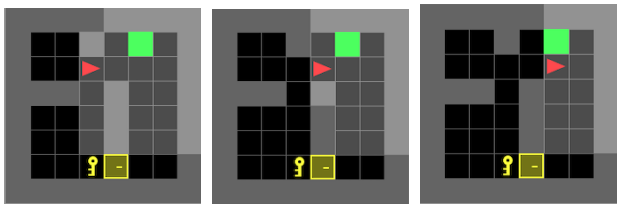


Fig. 10: Trajectory rollout in the 6×6 Shortcut environment: (a) initial view, (b) pick up key, (c) turn left, (d) navigate to door, (e) unlock door, (f) enter corridor, (g) reach goal.

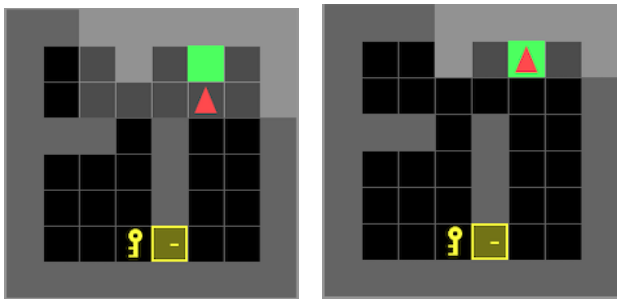
C. Example: 8×8 Direct Trajectory



1 Initial position 2 Move forward 3 Turn left



4 Move forward 5 Move forward 6 Turn left



7 Move forward 8 Reach goal

Fig. 11: Trajectory rollout in the 8×8 Direct environment: (a) initial position, (b) move forward, (c) turn left, (d) move forward, (e) move forward, (f) turn left, (g) move forward, (h) reach goal.

D. Discussion:Known map

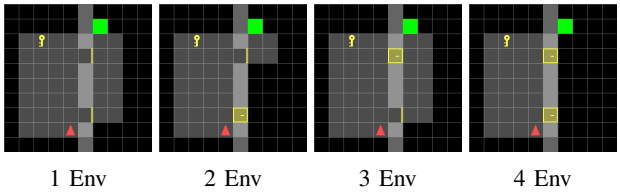
In the Known-Map scenarios, our value-iteration solver consistently found the true minimum-cost trajectories in all seven environments. For maps without a blocking door (5×5 Direct, 6×6 Direct, 8×8 Direct), the policy reduces to the pure shortest-path sequence of forward and turn actions, matching the geometric optimum. In maps where the door lies on the direct route (5×5

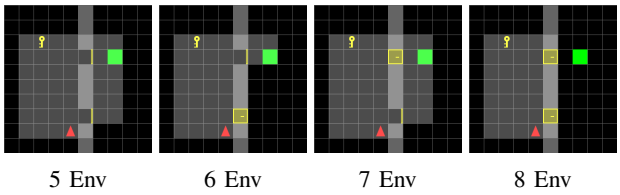
Normal, 6×6 Normal, 8×8 Normal), the solver correctly detours to pick up the key, applies the unlock action, and then resumes an optimal path to the goal. Finally, in the "Shortcut" variants (6×6 Shortcut, 8×8 Shortcut), the policy exploits the door as a cost-reducing shortcut, illustrating that value iteration naturally identifies non-intuitive routes whenever they reduce cumulative cost.

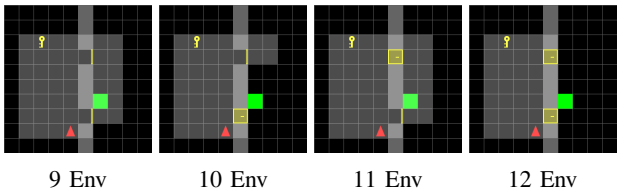
E. Random Map

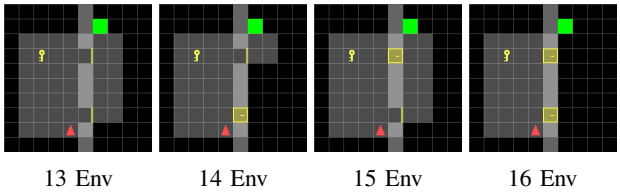
TABLE VIII: Random Map: Action sequences and total costs for all 36 environments

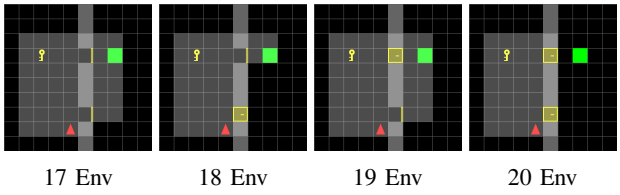
Env	Action Sequence	Cost
1	MF, TR, MF, MF, TL, MF, MF, MF	11
2	MF, TR, MF, MF, TL, MF, MF, MF	11
3	MF, TR, MF, MF, TL, MF, MF, MF	11
4	TL, MF, MF, TR, MF, MF, MF, MF, PK, TR, MF, UD, MF, MF, TL, MF, MF	19
5	MF, MF, MF, MF, MF, TR, MF, MF	9
6	MF, MF, MF, MF, MF, TR, MF, MF	9
7	MF, TR, MF, MF, TL, MF, MF, MF	10
8	TL, MF, MF, TR, MF, MF, MF, MF, PK, TR, MF, UD, MF, MF, MF	17
9	MF, TR, MF, MF, TL, MF	6
10	MF, TR, MF, MF, TL, MF	7
11	MF, TR, MF, MF, TL, MF	6
12	MF, TR, MF, MF, TL, MF	6
13	MF, TR, MF, MF, TL, MF	11
14	MF, MF, MF, TR, MF, MF, TL, MF, MF	11
15	MF, TR, MF, MF, TL, MF, MF, MF	11
16	MF, TL, MF, PK, TL, TL, MF, UD, MF, MF, TL, MF, MF	17
17	MF, TR, MF, MF, TL, MF, MF, MF	9
18	MF, TR, MF, MF, TL, MF, MF, MF	9
19	MF, TR, MF, MF, TL, MF	10
20	MF, TL, MF, PK, TL, TL, MF, UD, MF, MF, MF	15
21	MF, TR, MF, MF, TL, MF	6
22	MF, MF, MF, MF, MF, TL, MF, MF	12
23	MF, TR, MF, MF, TL, MF	6
24	MF, TL, MF, PK, TL, TL, MF, UD, MF, TR, MF, MF, MF	18
25	MF, TR, MF, MF, TL, MF, MF, MF	11
26	MF, MF, MF, MF, TL, MF, MF, MF	11
27	MF, TR, MF, MF, TL, MF, MF, MF	11
28	MF, PK, TR, MF, MF, TR, MF, MF, UD, MF, MF, TL, MF, MF	19
29	MF, TR, MF, MF, TL, MF, MF, MF	9
30	MF, TR, MF, MF, TL, MF, MF, MF	9
31	MF, TR, MF, MF, TL, MF, MF, MF	10
32	MF, MF, TL, MF, MF, TL, MF, MF	17
33	MF, TR, MF, MF, TL, MF, MF, MF	6
34	MF, MF, MF, MF, MF, MF, MF	9
35	MF, TR, MF, MF, TL, MF	6
36	TL, MF, PK, TR, MF, MF, MF, UD, MF, MF, TL, MF, MF	16

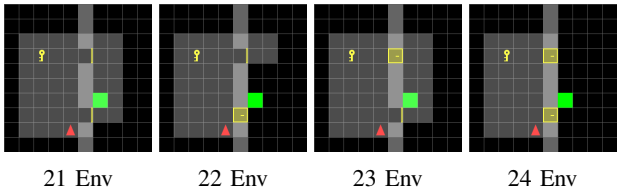

1 Env 2 Env 3 Env 4 Env

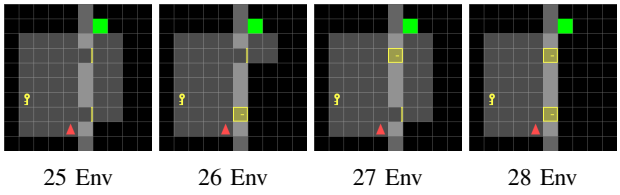

5 Env 6 Env 7 Env 8 Env

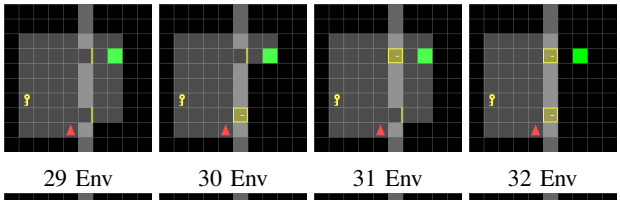

9 Env 10 Env 11 Env 12 Env

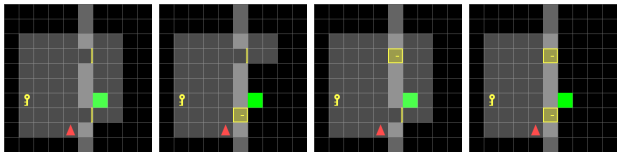

13 Env 14 Env 15 Env 16 Env


17 Env 18 Env 19 Env 20 Env

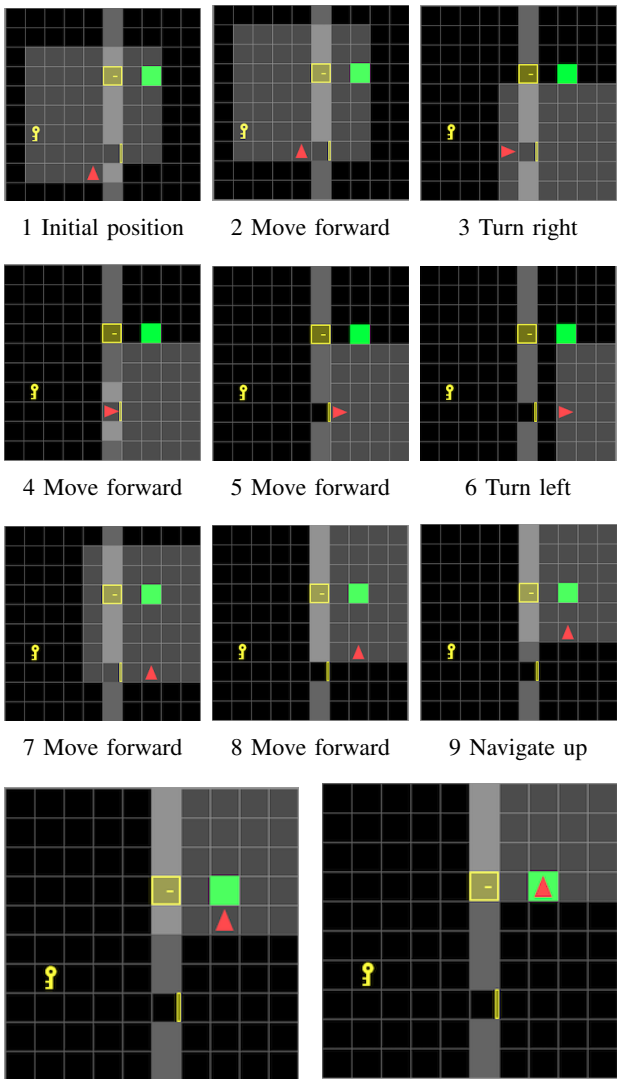

21 Env 22 Env 23 Env 24 Env


25 Env 26 Env 27 Env 28 Env


29 Env 30 Env 31 Env 32 Env


33 Env 34 Env 35 Env 36 Env

F. Example: 10×10 Environment #31



1 Initial position 2 Move forward 3 Turn right

4 Move forward 5 Move forward 6 Turn left

7 Move forward 8 Move forward 9 Navigate up

10 Move forward 11 Reach goal

Fig. 13: Trajectory rollout in the 10×10 Env #31: (a) initial position, (b) move forward, (c) turn right, (d) move forward, (e) move forward, (f) turn left, (g) move forward, (h) move forward, (i) navigate up, (j)move forward, (k) reach goal.

G. Discussion (Random Map)

Table VIII reports the performance of the single universal policy across all 36 random 10×10 environments. The achieved costs range from 6 to 20 steps, with an average of approximately 10.9. In layouts where the door is already open or does not block the direct path (e.g., Env 9, 11, 23, 33), the policy defaults to the pure shortest-path sequence of forward and turn actions. In environments requiring key pickup and door unlocking (e.g., Env 4, 8, 16, 24, 28, 36), the policy correctly detours to collect the key (PK), applies the unlock action (UD), and then proceeds along the globally optimal route to the goal.

Because this policy must plan over all possible key and goal indices at design time, it cannot specialize further to each sampled instance, yielding slightly higher costs than per-instance Known-Map solutions. Nevertheless, it requires no additional dynamic-programming computation at execution time—only constant-time table lookups—making it highly efficient for online deployment. The main drawback is the enlarged state space (approximately $3\times3\times10\times10\times4\times2\times2 \approx 14{,}400$ states), which increases offline planning time and memory usage.

Future enhancements could include:

- **Prioritized sweeping:** Focus value-iteration updates on states reachable from the initial condition to avoid unnecessary computations.
- **Heuristic warm-start:** Use Known-Map solutions to initialize the value function in Random Map, accelerating convergence.
- **Approximate dynamic programming:** Employ function approximation (e.g., fitted-value iteration) to reduce memory and computational overhead, enabling scalability to larger grids or richer uncertainty models.