



دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر



گزارش تمرین شماره ۳

درس یادگیری تعاملی

پاییز ۱۴۰۱

نام و نام خانوادگی

مهیار ملکی

شماره دانشجویی

۸۱۰۱۰۰۴۷۶

چکیده.....	۴
مسئله ۱ – آشنایی با MDP.....	۵
هدف سوال.....	۵
سوال ۱.....	۵
سوال ۲.....	۶
مسئله ۲ – پیاده‌سازی دستی.....	۷
هدف سوال.....	۷
سوال ۱.....	۷
توضیحات.....	۷
نتیجه.....	۱۵
سوال ۲.....	۱۶
توضیحات.....	۱۶
نتیجه.....	۱۸
سوال ۳.....	۱۹
توضیحات.....	۱۹
نتیجه.....	۲۳
سوال ۴.....	۲۴
روند اجرای کد پیاده‌سازی.....	۲۴
مسئله ۳ – شبیه‌سازی.....	۲۵
هدف سوال.....	۲۵
توضیحات پیاده‌سازی.....	۲۶

نتایج.....	۳۰
بخش اول.....	۳۰
بخش دوم.....	۳۳
بخش سوم.....	۳۶
روند اجرای کد پیاده‌سازی.....	۴۱
منابع.....	۴۲

چکیده

هدف از این تمرین آشنایی با مسائل MDP، مدل‌سازی و حل آنهاست. بدین منظور در سوال اول به مدل‌سازی یک مسئله دنیای می‌پردازیم. در سوال دوم یک مسئله MDP را به صورت دستی با استفاده از هر دو الگوریتم policy iteration و value iteration حل و بررسی می‌کنیم. و در نهایت در سوال آخر به پیاده سازی کامپیوتری این الگوریتم‌ها و حل یک مسئله نسبتاً دشوارتر خواهیم پرداخت.

مسئله ۱ – آشنایی با MDP

هدف سوال

در این بخش با بررسی چند مسئله دنیای واقعی، برای آنها مدل MDP ارائه خواهیم داد. همچنین مجموعه استیت‌ها، اکشن‌ها، پاداش و انتقال بین استیت‌ها را در مسائل مشخص خواهیم کرد.

سوال ۱

• اکشن‌ها

در اینجا با یک recommender system سر و کار داریم لذا اعمال ما پیشنهادهایی است که برنامه به کاربر می‌دهد. به عنوان مثال در این سوال و برای برنامه فیتنس ورزش‌ها، تمرین‌ها و همچنین غذاهای مختلفی که برنامه به کاربر پیشنهاد می‌دهد، همان اکشن‌ها می‌باشند.

• استیت‌ها

در یک recommender system پیشنهادهایی که به کاربر داده می‌شود (همان اکشن‌ها که در قسمت قبل به آن اشاره شد)، می‌تواند بر اساس مواردی چون اطلاعات ورودی کاربر مانند قد یا وزن یا سن یا حتی هدف کاربر (کاهش وزن، تناسب اندام، بدن‌سازی و ...) و موارد دیگر و یا بر اساس n تا پیشنهاد قبلی باشد. لذا هر استیت را می‌توان مجموعه‌ای از این موارد در نظر گرفت. به عنوان مثال برای استیت k می‌توان مجموعه زیر را در نظر گرفت:

[food_3, food_2, food_1, exercise_3, exercise_2, exercise_1, age, weight, age, bodybuilding, ...]

• پاداش

پاداش می‌تواند موارد مختلفی را شامل شود مانند میزان کاهش وزن، میزان رضایت کاربر، میزان علاقه کاربر، میزان چربی سوزی و ... این موارد باید به طور مرتب در برنامه ثبت شوند تا برنامه فیدبک مورد نیاز خود را از کاربر دریافت کند.

• انتقال بین استیت‌ها

همانطور که گفته شد با هر پیشنهاد جدید توسط برنامه، وارد استیت جدیدی خواهیم شد. ولی امکان و احتمال این انتقال تنها با تعامل با کاربر بدست خواهد آمد. برای اینکه کاربر توانایی انجام ورزش یا رژیم جدید را دارد یا خیر را تنها خود عامل می‌تواند مشخص کند. البته این امر می‌تواند به تدریج توسط برنامه یاد گرفته شود.

🌈 مزایا و معایب نسبت به توصیه‌گر معمولی

مزایا: برنامه به طور دائم از کاربر فیدبک می‌گیرد و پیشنهادات خود را براساس این اطلاعات دریافتی بروزرسانی می‌کند. همچنین فضای تعاملی برنامه می‌تواند برای کاربران جذابتر باشد.

معایب: سهل‌انگاری یا فراموشی کاربران در فیدبک دادن و بروزرسانی اطلاعات خود باعث می‌شود اختلال در عملکرد برنامه خواهد شد. همچنین پیشنهادات اولیه برنامه بدلیل کمبود دانش آن، ممکن است خوشایند کاربران نباشد.

سوال ۲

مسئله ۲ – پیاده‌سازی دستی

هدف سوال

در این سوال به مدلسازی و حل یک مسئله MDP با استفاده از الگوریتم‌های value iteration و policy iteration به صورت دستی خواهیم پرداخت تا گام به گام با نحوه عملکرد این الگوریتم‌ها در مسائل MDP با شرایط مختلف و همچنین اثر تغییر هایپرپارامترها آشنا شویم.

سوال ۱

توضیحات

همان‌طور که در کتاب نیز قابل مشاهده است، شبه کد الگوریتم policy iteration به شرح شکل ۱ می‌باشد. در این قسمت با شرط ضریب discount برابر با صفر به پیاده‌سازی دستی این الگوریتم می‌پردازیم.

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
 $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$
2. Policy Evaluation
Loop:
 $\Delta \leftarrow 0$
Loop for each $s \in \mathcal{S}$:
 $v \leftarrow V(s)$
 $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s)) [r + \gamma V(s')]$
 $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)
3. Policy Improvement
 $policy_stable \leftarrow true$
For each $s \in \mathcal{S}$:
 $old_action \leftarrow \pi(s)$
 $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a) [r + \gamma V(s')]$
If $old_action \neq \pi(s)$, then $policy_stable \leftarrow false$
If $policy_stable$, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

شکل ۱- شبه کد الگوریتم policy iteration

در ابتدا ارزش هر خانه را با عدد صفر به طوری که در جدول ۱ قابل مشاهده است، مقداردهی اولیه می‌کنیم. همچنین سیاست اولیه نیز طوری تنظیم می‌شود که در ابتدا احتمال انتخاب تمام حرکات به صورت یکسان (25%) باشد. سپس با توجه پاداشی که از محیط می‌گیریم و ارزش هر خانه، الگوریتم را پیاده می‌کنیم.

جدول ۱- ارزش هر خانه

0	0	Hell	0
Obstacle	0	0	0
0	0	0	Goal

جدول ۲- سیاست اولیه (احتمال انتخاب هر اکشن در هر خانه)

Action State	Left	Down	Right	Up
0	0.25	0.25	0.25	0.25
1	0.25	0.25	0.25	0.25
Hell	0	0	0	0
3	0.25	0.25	0.25	0.25
Obstacle	0	0	0	0
5	0.25	0.25	0.25	0.25
6	0.25	0.25	0.25	0.25
7	0.25	0.25	0.25	0.25
8	0.25	0.25	0.25	0.25
9	0.25	0.25	0.25	0.25
10	0.25	0.25	0.25	0.25
Goal	0	0	0	0

• تکرار اول:

Policy Evaluation ○

- اولین حلقه:

با شروع از خانه صفر، به ترتیب ارزش هر خانه را بروزرسانی می‌کنیم. لازم به ذکر است که به صورت کلی انتخاب خانه شروع و ترتیب بروزرسانی ارزش خانه‌ها اهمیتی نداشته و می‌تواند حتی به صورت تصادفی باشد. همچنین پس از بروزرسانی ارزش هر خانه، مقدار آن درجا اعمال شده و برای بروزرسانی خانه‌های مجاور از ارزش بروزرسانی شده استفاده خواهد شد.

$$V(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$$

$$\begin{aligned} V(0) &= \overbrace{\frac{1}{4} \times 1 \times (0 + 0 \times 0)}^{Left} + \overbrace{\frac{1}{4} \times 1 \times (0 + 0 \times 0)}^{Down} + \overbrace{\frac{1}{4} \times 1 \times (0 + 0 \times 0)}^{Right} + \overbrace{\frac{1}{4} \times 1 \times (0 + 0 \times 0)}^{Up} = 0 \\ V(1) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (-10 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) = -2.5 \\ V(2) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) = 0 \\ V(3) &= \frac{1}{4} \times 1 \times (-10 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) = -2.5 \\ V(4) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) = 0 \\ V(5) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times -2.5) = 0 \\ V(6) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (-10 + 0 \times 0) = -2.5 \\ V(7) &= \frac{1}{4} \times 1 \times (0 + 0 \times -2.5) + \frac{1}{4} \times 1 \times (10 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times -2.5) = 2.5 \\ V(8) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) = 0 \\ V(9) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) = 0 \\ V(10) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (10 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times -2.5) = 2.5 \\ V(11) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) = 0 \end{aligned}$$

جدول ۳- ارزش هر خانه پس از اولین حلقه‌ی تکرار اول

0	-2.5	Hell	-2.5
Obstacle	0	-2.5	2.5
0	0	2.5	Goal

- دومین حلقه:

در این حلقه مشاهده می‌شود که ارزش خانه‌ها ثابت مانده و بروزرسانی نمی‌شوند. این امر بدین دلیل است که ضریب discount را صفر در نظر گرفته‌ایم در نتیجه ارزش هر خانه را تنها با توجه به پاداشی که می‌گیریم بروزرسانی می‌کنیم و به خانه‌های مجاور آن کاری نداریم. بنابراین پس از اتمام این حلقه مقدار دلتا صفر شده و در نتیجه مرحله evaluation متوقف می‌شود.

$$\begin{aligned}
 V(0) &= \overbrace{\frac{1}{4} \times 1 \times (0 + 0 \times 0)}^{\text{Left}} + \overbrace{\frac{1}{4} \times 1 \times (0 + 0 \times 0)}^{\text{Down}} + \overbrace{\frac{1}{4} \times 1 \times (0 + 0 \times -2.5)}^{\text{Right}} + \overbrace{\frac{1}{4} \times 1 \times (0 + 0 \times 0)}^{\text{Up}} = 0 \\
 V(1) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (-10 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) = -2.5 \\
 V(2) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) = 0 \\
 V(3) &= \frac{1}{4} \times 1 \times (-10 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 2.5) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) = -2.5 \\
 V(4) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) = 0 \\
 V(5) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times -2.5) + \frac{1}{4} \times 1 \times (0 + 0 \times -2.5) = 0 \\
 V(6) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 2.5) + \frac{1}{4} \times 1 \times (0 + 0 \times 2.5) + \frac{1}{4} \times 1 \times (-10 + 0 \times 0) = -2.5 \\
 V(7) &= \frac{1}{4} \times 1 \times (0 + 0 \times -2.5) + \frac{1}{4} \times 1 \times (10 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times -2.5) = 2.5 \\
 V(8) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) = 0 \\
 V(9) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 2.5) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) = 0 \\
 V(10) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (10 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times -2.5) = 2.5 \\
 V(11) &= \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) + \frac{1}{4} \times 1 \times (0 + 0 \times 0) = 0
 \end{aligned}$$

جدول ۴- ارزش هر خانه پس از دومین حلقه‌ی تکرار اول

0	-2.5	Hell	-2.5
Obstacle	0	-2.5	2.5
0	0	2.5	Goal

Policy Improvement ○

در این مرحله با توجه به شبهه کد الگوریتم، q-value هر خانه را به ازای حرکات مختلف محاسبه می‌کنیم. سپس حرکتی که بیشترین مقدار q-value را داشته انتخاب کرده و سیاست را بروزرسانی می‌کنیم (در صورت برابر بودن q-value حرکات، اولین حرکت را انتخاب می‌کنیم). در انتها اگر سیاست بدست آمده نسبت به سیاست قبلی تغییری نداشته باشد، الگوریتم policy iteration به پایان می‌رسد، در غیر این صورت دوباره وارد مرحله evaluation می‌شویم. (در اینجا نیز بدلیل تغییر سیاست دوباره وارد مرحله evaluation می‌شویم)

$$Q(s|a) = \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$$

$$Q(0) = \left[\overbrace{1 \times (0 + 0 \times 0)}^{Left}, \overbrace{1 \times (0 + 0 \times 0)}^{Down}, \overbrace{1 \times (0 + 0 \times -2.5)}^{Right}, \overbrace{1 \times (0 + 0 \times 0)}^{Up} \right] = [0, 0, 0, 0]$$

$$Q(1) = [1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times -2.5), 1 \times (0 + 0 \times -2.5)] = [0, 0, 0, 0]$$

$$Q(3) = [1 \times (-10 + 0 \times 0), 1 \times (0 + 0 \times 2.5), 1 \times (0 + 0 \times -2.5), 1 \times (0 + 0 \times -2.5)] = [-10, 0, 0, 0]$$

$$Q(5) = [1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times -2.5), 1 \times (0 + 0 \times -2.5)] = [0, 0, 0, 0]$$

$$Q(6) = [1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 2.5), 1 \times (0 + 0 \times 2.5), 1 \times (-10 + 0 \times 0)] = [0, 0, 0, -10]$$

$$Q(7) = [1 \times (0 + 0 \times -2.5), 1 \times (10 + 0 \times 0), 1 \times (0 + 0 \times 2.5), 1 \times (0 + 0 \times -2.5)] = [0, 10, 0, 0]$$

$$Q(8) = [1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0)] = [0, 0, 0, 0]$$

$$Q(9) = [1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 2.5), 1 \times (0 + 0 \times -2.5)] = [0, 0, 0, 0]$$

$$Q(10) = [1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 2.5), 1 \times (10 + 0 \times 0), 1 \times (0 + 0 \times -2.5)] = [0, 0, 10, 0]$$

After argmax:

جدول ۵- سیاست بهینه بروزرسانی شده در تکرار اول

Action \ State	Left	Down	Right	Up
0	1	0	0	0
1	1	0	0	0
Hell	0	0	0	0
3	0	1	0	0
Obstacle	0	0	0	0
5	1	0	0	0
6	1	0	0	0
7	0	1	0	0
8	1	0	0	0
9	1	0	0	0
10	0	0	1	0
Goal	0	0	0	0

• تکرار دوم:

Policy Evaluation ○

مانند تکرار اول، ارزش هر خانه را بروزرسانی می‌کنیم. ولی این بار با استفاده از سیاست جدید

این عملیات را انجام خواهیم داد.

- اولین حلقه:

$$\begin{aligned}
 V(0) &= \overbrace{1 \times 1 \times (0 + 0 \times 0)}^{Left} + \overbrace{0 \times 1 \times (0 + 0 \times 0)}^{Down} + \overbrace{0 \times 1 \times (0 + 0 \times -2.5)}^{Right} + \overbrace{0 \times 1 \times (0 + 0 \times 0)}^{Up} = 0 \\
 V(1) &= 1 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (-10 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) = 0 \\
 V(2) &= 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) = 0 \\
 V(3) &= 0 \times 1 \times (-10 + 0 \times 0) + 1 \times 1 \times (0 + 0 \times 2.5) + 0 \times 1 \times (0 + 0 \times -2.5) + 0 \times 1 \times (0 + 0 \times -2.5) = 0 \\
 V(4) &= 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) = 0 \\
 V(5) &= 1 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times -2.5) + 0 \times 1 \times (0 + 0 \times -2.5) = 0 \\
 V(6) &= 1 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 2.5) + 0 \times 1 \times (0 + 0 \times 2.5) + 0 \times 1 \times (-10 + 0 \times 0) = 0 \\
 V(7) &= 0 \times 1 \times (0 + 0 \times -2.5) + 1 \times 1 \times (10 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times -2.5) = 10 \\
 V(8) &= 1 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) = 0 \\
 V(9) &= 1 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 2.5) + 0 \times 1 \times (0 + 0 \times 0) = 0 \\
 V(10) &= 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 1 \times 1 \times (10 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times -2.5) = 10 \\
 V(11) &= 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) = 0
 \end{aligned}$$

جدول ۶ - ارزش هر خانه پس از اولین حلقه‌ی تکرار دوم

0	0	Hell	0
Obstacle	0	0	10
0	0	10	Goal

- دومین حلقه:

در این حلقه نیز مجدداً مشاهده می‌شود که ارزش خانه‌ها ثابت مانده و بروزرسانی نمی‌شوند. بنابراین پس از اتمام این حلقه مقدار دلتا صفر شده و در نتیجه مرحله evaluation متوقف می‌شود.

$$\begin{aligned}
 V(0) &= \overbrace{1 \times 1 \times (0 + 0 \times 0)}^{Left} + \overbrace{0 \times 1 \times (0 + 0 \times 0)}^{Down} + \overbrace{0 \times 1 \times (0 + 0 \times 0)}^{Right} + \overbrace{0 \times 1 \times (0 + 0 \times 0)}^{Up} = 0 \\
 V(1) &= 1 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (-10 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) = 0 \\
 V(2) &= 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) = 0 \\
 V(3) &= 0 \times 1 \times (-10 + 0 \times 0) + 1 \times 1 \times (0 + 0 \times 10) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) = 0 \\
 V(4) &= 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) = 0 \\
 V(5) &= 1 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) = 0 \\
 V(6) &= 1 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 10) + 0 \times 1 \times (0 + 0 \times 10) + 0 \times 1 \times (-10 + 0 \times 0) = 0 \\
 V(7) &= 0 \times 1 \times (0 + 0 \times 0) + 1 \times 1 \times (10 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) = 10 \\
 V(8) &= 1 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) = 0 \\
 V(9) &= 1 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 10) + 0 \times 1 \times (0 + 0 \times 0) = 0 \\
 V(10) &= 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 1 \times 1 \times (10 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) = 10 \\
 V(11) &= 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) + 0 \times 1 \times (0 + 0 \times 0) = 0
 \end{aligned}$$

جدول ۷- ارزش هر خانه پس از دومین حلقه‌ی تکرار دوم

0	0	Hell	0
Obstacle	0	0	10
0	0	10	Goal

Policy Improvement ○

پس از انجام دوباره این مرحله، مشاهده می‌شود که سیاست بدست آمده با سیاست قبلی یکسان بوده و تغییری صورت نگرفته است. بنابراین مطابق شبه کد شکل یک، الگوریتم policy iteration پایان می‌یابد.

$$\begin{aligned}
 Q(0) &= \left[\overbrace{1 \times (0 + 0 \times 0)}^{Left}, \overbrace{1 \times (0 + 0 \times 0)}^{Down}, \overbrace{1 \times (0 + 0 \times 0)}^{Right}, \overbrace{1 \times (0 + 0 \times 0)}^{Up} \right] = [0, 0, 0, 0] \\
 Q(1) &= [1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0)] = [0, 0, 0, 0] \\
 Q(3) &= [1 \times (-10 + 0 \times 0), 1 \times (0 + 0 \times 10), 1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0)] = [-10, 0, 0, 0] \\
 Q(5) &= [1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0)] = [0, 0, 0, 0] \\
 Q(6) &= [1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 10), 1 \times (0 + 0 \times 10), 1 \times (-10 + 0 \times 0)] = [0, 0, 0, -10] \\
 Q(7) &= [1 \times (0 + 0 \times 0), 1 \times (10 + 0 \times 0), 1 \times (0 + 0 \times 10), 1 \times (0 + 0 \times 0)] = [0, 10, 0, 0] \\
 Q(8) &= [1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0)] = [0, 0, 0, 0] \\
 Q(9) &= [1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 10), 1 \times (0 + 0 \times 0)] = [0, 0, 0, 0] \\
 Q(10) &= [1 \times (0 + 0 \times 0), 1 \times (0 + 0 \times 10), 1 \times (10 + 0 \times 0), 1 \times (0 + 0 \times 0)] = [0, 0, 10, 0]
 \end{aligned}$$

After argmax:

جدول ۸ - سیاست بهینه برزسانی شده در تکرار دوم

Action \ State	Left	Down	Right	Up
0	1	0	0	0
1	1	0	0	0
Hell	0	0	0	0
3	0	1	0	0
Obstacle	0	0	0	0
5	1	0	0	0
6	1	0	0	0
7	0	1	0	0
8	1	0	0	0
9	1	0	0	0
10	0	0	1	0
Goal	0	0	0	0

نتیجه

همان طور که مشاهده کردیم با در نظر گرفتن ضریب discount برابر صفر، عامل به صورت myopic یا همان نزدیک‌بین عمل می‌کند، یعنی تنها به پاداشی که از طریق انتقال به خانه‌های مجاورش به آن می‌رسد نگاه می‌کند و کاری به خانه‌های دورتر ندارد.

همچنین راجب نحوه بروزرسانی خانه‌ها، دیدیم که در ابتدا خانه‌های اطراف hell مقدار 2.5- را اتخاذ کردند ولی در ادامه با پیشرفت الگوریتم، عامل یاد گرفت که دیگر حرکت منجر به پاداش منفی را انتخاب نکند، لذا در نهایت ارزش خانه‌های اطراف hell صفر شد.

در مورد سیاست نیز، چنانچه در شکل ۲ قابل مشاهده است، در تعداد زیادی از خانه‌ها عمل بهینه که منجر به رسیدن عامل به هدف می‌شود، انتخاب نشده است که این امر ناشی از همان myopic بودن عامل می‌باشد. لذا در ادامه انتظار می‌رود با در نظر گرفتن ضریب discount به سیاست بهینه دست یافت.

Values & Policies			
0.00 ←	0.00 ←	0.00	0.00 ↓
0.00	0.00 ←	0.00 ←	10.00 ↓
0.00 ←	0.00 ←	10.00 →	0.00

شکل ۲ - ارزش‌ها و سیاست نهایی

سوال ۲

توضیحات

- تکرار اول:

Policy Evaluation ○

- اولین حلقه:

در این قسمت با در نظر گرفتن ضریب $discount$ برابر با 0.9 عامل به ارزش خانه‌های دورتر از خود نیز توجه کرده و ارزش هر خانه به صورت زیر بروزرسانی می‌شود:

$$\begin{aligned}
 V(0) &= \overbrace{\frac{1}{4} \times 1 \times (0 + 0.9 \times 0)}^{Left} + \overbrace{\frac{1}{4} \times 1 \times (0 + 0.9 \times 0)}^{Down} + \overbrace{\frac{1}{4} \times 1 \times (0 + 0.9 \times 0)}^{Right} + \overbrace{\frac{1}{4} \times 1 \times (0 + 0.9 \times 0)}^{Up} = 0 \\
 V(1) &= \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (-10 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) = -2.5 \\
 V(2) &\xrightarrow{Terminal} 0 \\
 V(3) &= \frac{1}{4} \times 1 \times (-10 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) = -2.5 \\
 V(4) &\xrightarrow{Terminal} 0 \\
 V(5) &= \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times -2.5) = -0.56 \\
 V(6) &= \frac{1}{4} \times 1 \times (0 + 0.9 \times -0.56) + \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (-10 + 0.9 \times 0) = -2.63 \\
 V(7) &= \frac{1}{4} \times 1 \times (0 + 0.9 \times -2.62) + \frac{1}{4} \times 1 \times (10 + 0.9 \times 0) + \frac{1}{4} \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times -2.5) = 1.35 \\
 V(8) &= \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) = 0 \\
 V(9) &= \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times -0.56) = -0.13 \\
 V(10) &= \frac{1}{4} \times 1 \times (0 + 0.9 \times -0.13) + \frac{1}{4} \times 1 \times (0 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (10 + 0.9 \times 0) + \frac{1}{4} \times 1 \times (0 + 0.9 \times -2.63) = 1.88 \\
 V(11) &\xrightarrow{Terminal} 0
 \end{aligned}$$

جدول ۹ - ارزش هر خانه پس از اولین حلقه‌ی تکرار اول

0	-2.5	Hell	-2.5
Obstacle	-0.56	-2.63	1.35
0	-0.13	1.88	Goal

- دومین حلقه:

بدلیل دشواری و زمان‌بر بودن پیاده‌سازی دستی الگوریتم، ادامه آن را با زبان برنامه‌نویسی پایتون پیاده‌سازی کرده و تنها نتایج را در ادامه گزارش خواهیم کرد.
با در گرفتن تنها برابر با 0.01 مشاهده می‌شود که پس از ۱۹ بار تکرار حلقه evaluation مقدار دلتا کمتر از 0.01 شده و وارد فاز improvement می‌شویم.

○ Policy Improvement

پس از اجرای این فاز، سیاست بروزرسانی شده در شکل ۳ قابل مشاهده است. بدلیل تغییر سیاست در این مرحله (به علت مقداردهی اولیه صفر) وارد تکرار دوم الگوریتم خواهیم شد.

Values & Policies

-3.23 ←	-4.69 ↓	0.00 .	-3.92 ↓
0.00 .	-1.83 ↓	-1.95 ↓	1.52 ↓
0.23 →	0.34 →	2.76 →	0.00 .

شکل ۳ - ارزش‌ها و سیاست انتخاب شده پس از تکرار اول

• تکرار دوم:

در این مرحله نیز پس از ۳۴ بار تکرار حلقه evaluation وارد فاز improvement می‌شویم. ارزش‌ها و سیاست بروزرسانی شده در شکل ۴ قابل مشاهده است.

Values & Policies

-0.09 →	7.29 ↓	0.00 .	9.00 ↓
0.00 .	8.10 ↓	9.00 ↓	10.00 ↓
8.10 →	9.00 →	10.00 →	0.00 .

شکل ۴ - ارزش‌ها و سیاست انتخاب شده پس از تکرار دوم

- تکرار سوم:

در تکرار سوم پس از دو بار اجرای فاز *evaluation* مقدار دلتا کمتر از مقدار تتا شده و وارد فاز *improvement* می‌شویم. در ادامه به دلیل تغییر نکردن و ثابت ماند سیاست اتخاذ شده در مرحله قبل، الگوریتم *policy iteration* به پایان می‌رسد.

Values & Policies

6.56 →	7.29 ↓	0.00	9.00 ↓
0.00	8.10 ↓	9.00 ↓	10.00 ↓
8.10 →	9.00 →	10.00 →	0.00

شکل ۵ - ارزش‌ها و سیاست نهایی

نتیجه

چنان چه در شکل ۵ قابل مشاهده است، مطابق انتظار با در نظر گرفتن ضریب *discount* برابر 0.9 می‌بینیم که عامل به صورت *farsighted* عمل کرده و برای تمام خانه‌ها سیاست اتخاذ شده منجر به رسیدن به هدف خواهد شد.

در مورد اثر مقدار ضریب *discount* نیز، می‌دانیم هر چه این مقدار بیشتر باشد (نزدیک به یک) هر خانه، ارزش خانه‌های دورتری را خواهددید. در نتیجه خانه‌های ابتدایی نقشه و دورتر از هدف، سریع‌تر پاداش مربوط به خانه هدف را متوجه شده و انتظار می‌رود با تکرار کمتری به سیاست بهینه دست یابیم.

سوال ۳

توضیحات

همان‌طور که در کتاب نیز قابل مشاهده است، شبه کد الگوریتم value iteration به شکل ۶ می‌باشد. در این قسمت با شرط ضریب discount برابر با 0.9 به پیاده‌سازی دستی این الگوریتم می‌پردازیم. همانند قبل در ابتدا ارزش هر خانه را با عدد صفر مقداردهی اولیه می‌کنیم. سپس با توجه پاداشی که از محیط می‌گیریم و ارزش هر خانه، الگوریتم را پیاده می‌کنیم.

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation

Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$

Loop:

| $\Delta \leftarrow 0$

| Loop for each $s \in \mathcal{S}$:

| $v \leftarrow V(s)$

| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$

| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that

$$\pi(s) = \arg\max_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$$

شکل ۶ - شبه کد الگوریتم value iteration

• تکرار اول:

در این قسمت با در نظر گرفتن ضریب $discount$ برابر با 0.9 عامل به ارزش خانه‌های دورتر از خود نیز توجه کرده و ارزش هر خانه به صورت زیر بروزرسانی می‌شود:

$$V(s) = \max_a \sum_{s',r} p(s',r | s, a) [r + \gamma V(s')]$$

$$V(0) = \max_a \left[\overbrace{1 \times (0 + 0.9 \times 0)}^{Left}, \overbrace{1 \times (0 + 0.9 \times 0)}^{Down}, \overbrace{1 \times (0 + 0.9 \times 0)}^{Right}, \overbrace{1 \times (0 + 0.9 \times 0)}^{Up} \right] = 0$$

$$V(1) = \max_a [1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (-10 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0)] = 0$$

$$V(2) \xrightarrow{Terminal} 0$$

$$V(3) = \max_a [1 \times (-10 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0)] = 0$$

$$V(4) \xrightarrow{Terminal} 0$$

$$V(5) = \max_a [1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0)] = 0$$

$$V(6) = \max_a [1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (-10 + 0.9 \times 0)] = 0$$

$$V(7) = \max_a [1 \times (0 + 0.9 \times 0), 1 \times (10 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0)] = 10$$

$$V(8) = \max_a [1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0)] = 0$$

$$V(9) = \max_a [1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0)] = 0$$

$$V(10) = \max_a [1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (10 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0)] = 10$$

$$V(11) \xrightarrow{Terminal} 0$$

جدول ۱۰- ارزش هر خانه پس از تکرار اول

0	0	Hell	0
Obstacle	0	0	10
0	0	10	Goal

• تکرار دوم:

$$V(0) = \max_a \left[\overbrace{1 \times (0 + 0.9 \times 0)}^{Left}, \overbrace{1 \times (0 + 0.9 \times 0)}^{Down}, \overbrace{1 \times (0 + 0.9 \times 0)}^{Right}, \overbrace{1 \times (0 + 0.9 \times 0)}^{Up} \right] = 0$$

$$V(1) = \max_a [1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (-10 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0)] = 0$$

$$V(2) \xrightarrow{Terminal} 0$$

$$V(3) = \max_a [1 \times (-10 + 0.9 \times 0), 1 \times (0 + 0.9 \times 10), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0)] = 9$$

$$V(4) \xrightarrow{Terminal} 0$$

$$V(5) = \max_a [1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0)] = 0$$

$$V(6) = \max_a [1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 10), 1 \times (0 + 0.9 \times 10), 1 \times (-10 + 0.9 \times 0)] = 9$$

$$V(7) = \max_a [1 \times (0 + 0.9 \times 9), 1 \times (10 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 9)] = 10$$

$$V(8) = \max_a [1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0)] = 0$$

$$V(9) = \max_a [1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 0), 1 \times (0 + 0.9 \times 10), 1 \times (0 + 0.9 \times 0)] = 9$$

$$V(10) = \max_a [1 \times (0 + 0.9 \times 9), 1 \times (0 + 0.9 \times 0), 1 \times (10 + 0.9 \times 0), 1 \times (0 + 0.9 \times 9)] = 10$$

$$V(11) \xrightarrow{Terminal} 0$$

جدول ۱۱- ارزش هر خانه پس از تکرار دوم

0	0	Hell	9
Obstacle	0	9	10
0	9	10	Goal

ادامه الگوریتم را با استفاده از کد پایتون آن انجام داده و تنها نتایج را گزارش خواهیم داد.

• تکرار سوم:

جدول ۱۲- ارزش هر خانه پس از تکرار سوم

0	0	Hell	9
Obstacle	8.1	9	10
8.1	9	10	Goal

- تکرار چهارم:

جدول ۱۳- ارزش هر خانه پس از تکرار چهارم

0	7.29	Hell	9
Obstacle	8.1	9	10
8.1	9	10	Goal

- تکرار پنجم:

جدول ۱۴- ارزش هر خانه پس از تکرار پنجم

6.56	7.29	Hell	9
Obstacle	8.1	9	10
8.1	9	10	Goal

- تکرار ششم:

در این مرحله مشاهده می شود که مقدار دلتا یا همان بیشینه اختلاف ارزش خانه ها در دو حلقه متوالی کمتر از تتای تنظیم شده (0.01) شده است، در واقع مقدار ارزش خانه ها تغییری نخواهد کرد. لذا الگوریتم متوقف شده و وارد فاز بروزرسانی سیاست می شویم.

- بروزرسانی سیاست بهینه:

Values & Policies

6.56 →	7.29 ↓	0.00	9.00 ↓
0.00	8.10 ↓	9.00 ↓	10.00 ↓
8.10 →	9.00 →	10.00 →	0.00

شکل ۷- ارزش ها و سیاست نهایی

نتیجه

با توجه در شکل ۷ می‌بینیم که مقادیر نهایی بدست آمده برای ارزش هر خانه توسط الگوریتم value iteration کاملاً برابر با مقادیر متناظر در الگوریتم policy evaluation شده‌است.

○ آیا مفهوم خاصی برای ارزش بدست آمده در هر مرحله وجود دارد؟

با تعقیب روند بروزرسانی ارزش‌ها مشاهده می‌شود که این روند از خانه هدف شروع شده و به تدریج تا دورترین نقطه از آن انتشار می‌یابد. این روند بروزرسانی بدین صورت است که ارزش خانه‌ها به ازای هر خانه فاصله از هدف با ضریب 0.9 کاهش خواهد یافت. در واقع اگر روند بروزرسانی را مانند این سوال از دورترین خانه نسبت به هدف شروع کنیم، برای رسیدن به ارزش‌های نهایی، به تعداد خانه‌هایی که در این مسیر وجود دارد، تکرار خواهیم داشت. برای مثال در این سوال پس از $n=5$ بار تکرار، ارزش تمام خانه‌ها نهایی شده و ارزش دورترین خانه از هدف $V(0) = 10 * 0.9^{(n-1)} = 6.56$ خواهد شد.










جالب این است که با انتخاب درست ترتیب بروزرسانی خانه‌ها (شروع از خانه هدف)، حتی می‌توان در یک تکرار به مقادیر نهایی ارزش‌ها دست یافت. این امر در مقایسه با الگوریتم policy iteration مزیت بزرگی حساب شده و باعث شده است که الگوریتم value iteration بسیار سریع‌تر عمل کند.

نکته دیگر آن است که خانه hell که پاداش منفی برمی‌گرداند تاثیری در محاسبات ارزش خانه‌های دیگر نداشته و مانند خانه Obstacle عمل می‌کند، این به بدین دلیل است که عامل در هر خانه، عملی را انتخاب می‌کند که بیشترین پاداش را از محیط بگیرد، لذا با بررسی خانه‌های اطراف هیچ گاه خانه hell را انتخاب نخواهد کرد.

○ آیا لزوماً در هر مرحله الگوریتم، سیاستی متناظر با ارزش‌های بدست آمده وجود دارد؟

در هر مرحله از الگوریتم می‌توان با توجه به ارزش‌های بدست آمده، سیاست را استخراج کرد، اما احتمالاً این سیاست تنها در خانه‌های نزدیک به هدف برابر با سیاست بهینه خواهد بود. زیرا همانطور که گفته شد، در این الگوریتم، بروزرسانی از اطراف خانه هدف شروع و به دورترین خانه انتشار خواهد یافت. برای مثال سیاست متناظر با تکرار دوم همین سوال (جدول ۱۱) به شکل زیر خواهد بود:

جدول ۱۱- سیاست استخراج شده از تکرار دوم

		Hell	
Obstacle			
			Goal

سوال ۴

اگر مانند سوال یک از ضریب discount برابر با صفر استفاده کنیم، کلا به ارزش خانه‌های مجاور کاری نداشته و مقدار اولیه ارزش‌ها هیچ تفاوتی برای ما ایجاد نخواهد کرد. البته دیدیم که با این روش اصلا به ارزش نهایی همگرا نشده و به سیاست بهینه نرسیدیم.

اما اگر از ضریب discount بین صفر و یک استفاده کنیم، همانطور که در بخش قبل بحث شد، ارزش هر خانه به ازای فاصله از هدف با ضریب γ کاهش می‌یابد تا به مقدار نهایی همگرا شود. حال اگر ارزش خانه‌ها به مقدار نهایی آن نزدیک باشد انتظار می‌رود که در تعداد تکرار کمتری به مقدار نهایی همگرا شویم. در واقع مانند این است که برای مثال در سوال ۳ از تکرار سوم شروع به اجرای الگوریتم کنیم. در این صورت در تکرار کمتر و با سرعت بیشتری به مقدار نهایی ارزش‌ها همگرا خواهیم شد.

البته لازم به ذکر است که عکس این قضیه نیز برقرار است. یعنی اگر مقادیر اولیه فاصله زیادی با مقادیر نهایی داشته باشند، الگوریتم ما در تکرار بیشتری به پاسخ نهایی همگرا خواهد شد.

روند اجرای کد پیاده‌سازی

تمام کدها و پیاده‌سازی‌ها در فایل `< HW3_Q2.ipynb >` قرار دارد. تنها لازم است که تمام سلول‌ها به ترتیب از ابتدا اجرا شوند تا نتایج و نمودارها به دست آیند.

مسئله ۳ - شبیه‌سازی

هدف سوال

در این سوال نیز به مدلسازی و حل یک مسئله MDP (دریاچه یخ‌زده) با استفاده از الگوریتم‌های value iteration و policy iteration ولی این بار با استفاده از زبان پایتون خواهیم پرداخت تا گام به گام با نحوه عملکرد این الگوریتم‌ها در مسائل MDP با شرایط مختلف و همچنین اثر تغییر هایپرپارامترها آشنا شویم. مسئله مورد بحث در این قسمت پیچیده‌تر بوده و پارامترهای بیشتری در آن دخیل شده‌اند.

توضیحات پیاده‌سازی

توابع مورد استفاده در این سوال به قرار زیر می‌باشند:

○ تابع `make_map`

از این تابع برای ایجاد محیط بازی استفاده می‌شود. ورودی `shape` ابعاد محیط، `studentNum` شماره دانشجویی، `safe_break_prob` احتمال شکست در مسیر امن و `random_break` نیز احتمال شکست یخ در مسیر ناامن را مشخص می‌کند به این صورت که اگر `True` باشد احتمال شکست به صورت تصادفی خواهد بود و در غیر این صورت احتمال شکست برابر ۱ خواهد شد. در نهایت این تابع نقشه محیط و همچنین مسیر امن را به عنوان خروجی باز خواهد گرداند.

```
def make_map(shape, studentNum, safe_break_prob, random_break):

    shape = np.array(shape)
    minmoves = sum(shape)-2
    np.random.seed(studentNum)
    move = np.zeros(minmoves) # Minimum moves for start to the end point
    idx = np.random.choice(range(minmoves), size=minmoves//2, replace=False)
    move[idx] = 1

    point = [0,0]
    lowprobs = [tuple(point)]

    for m in move:
        if m:
            point[0] += 1
        else:
            point[1] += 1
        lowprobs.append(tuple(point))

    idx = np.array(lowprobs)
    map = np.ones(shape)
    if random_break:
        map *= np.random.rand(shape[0], shape[1])
    map[idx[:,0], idx[:,1]] = safe_break_prob
    map[0,0] = 0.0 # Start point
    map[shape[0]-1, shape[1]-1] = 0.0 # End point

    safe_path = np.zeros(shape)
    safe_path[idx[:,0], idx[:,1]] = 1
    safe_path[0,0] = 1
    safe_path[shape[0]-1, shape[1]-1] = 1

    return map, safe_path
```

○ کلاس FrozenLake

تمام توابع تعریف شده دیگر در این کلاس قرار می‌گیرند. در تابع `init` این کلاس به محاسبه ماتریس احتمال انتقال، ماتریس خانه‌های مجاور، ماتریس احتمالات لیز خوردن، ماتریس پاداش‌ها و همچنین ماتریس احتمالات شکستن یخ می‌پردازیم.

○ تابع Qvalues

از فرمول زیر برای محاسبه q-value هر عمل به ازای هر استیت استفاده می‌شود. در ادامه هر جا نیاز به محاسبه q-value باشد از این فرمول استفاده می‌کنیم:

$$Q(s) = \sum_{s'} P(s'|s, a) [P_{fail}(R(s) + Reward_{fail}) + (1 - P_{fail})(R(s) + \gamma V(s'))]$$

```
def Qvalues(self):
    Qs = np.zeros((self.numstates, self.numactions))
    for state in range(self.numstates):
        Qs[state] = np.sum(self.transitions[state] * \
                           ((1-self.pfail[state]) * (self.rewards[state] + self.Discount * np.tile(self.V, (self.numactions, 1))) + \
                            (np.tile(self.pfail[state], (self.numactions, 1)) * (self.rewards[state]+self.failReward)) ), axis=1)
    return pd.DataFrame(Qs, columns=['Left', 'Down', 'Right', 'Up'])
```

○ تابع Policy_Evaluation

مطابق شبه کدی که در کتاب ارائه شده فاز evaluation الگوریتم policy iteration را در این تابع پیاده‌سازی می‌کنیم.

```
def Policy_Evaluation(self):
    while True:
        self.counter = 0
        delta = 0
        for state in range(self.numstates):
            v = self.V[state].copy()
            q = np.sum(self.transitions[state] * \
                       ((1-self.pfail[state]) * (self.rewards[state] + self.Discount * np.tile(self.V, (self.numactions, 1))) + \
                        (np.tile(self.pfail[state], (self.numactions, 1)) * (self.rewards[state]+self.failReward)) ), axis=1)
            self.V[state] = np.sum(self.probability[state] * q)
            delta = max(delta, np.abs(v - self.V[state]))

        self.counter += 1
        if delta < self.theta:
            break
```

○ تابع Policy_Improvement

در اینجا نیز طبق شبه کد کتاب، فاز improvement را پیاده‌سازی می‌کنیم. مقادیر q-value هر عمل را نیز مانند قبل محاسبه می‌کنیم. در انتها عملی که بیشترین مقدار q-value داشته باشد به عنوان سیاست بهینه‌ی آن استیت برمی‌گردانیم.

```
def Policy_Improvement(self):
    policy_stable = True
    for state in range(self.numstates):
        if self.resapedMap[state] == 1 or state == self.numstates-1:
            continue

        old = self.probability[state].copy()
        q = np.sum(self.transitions[state] * \
                    ((1-self.pfail[state]) * (self.rewards[state] + self.Discount * np.tile(self.V, (self.numactions, 1))) + \
                     (np.tile(self.pfail[state], (self.numactions, 1)) * (self.rewards[state]+self.failReward))), axis=1)

        opt_act = np.argmax(q)
        self.probability[state] = np.eye(self.numactions)[opt_act]
        if (self.probability[state] != old).any():
            policy_stable = False

    return policy_stable
```

○ تابع Policy_Iteration

با قرار دادن دو تابع Policy_Evaluation و Policy_Improvement در کنار در یک حلقه، نهایتاً الگوریتم policy iteration ساخته خواهد شد.

```
def Policy_Iteration(self):
    self.reset()
    counter = 0
    start = time.time()
    while True:
        counter += 1
        self.Policy_Evaluation()
        policy_stable = self.Policy_Improvement()

        if policy_stable:
            end = time.time()
            print(f'Number of Iterations: {counter} \nTime Elapsed: {(end - start):.3f}s')
            break
```

○ تابع Value_Iteration

این الگوریتم شباهت زیادی به الگوریتم policy iteration دارد، لذا این تابع را نیز همانند توابع قبلی با توجه به شبه کد آن پیاده‌سازی می‌کنیم.

```
def Value_Iteration(self):
    self.reset()
    start = time.time()
    counter = 0
    while True:
        counter += 1
        delta = 0
        for state in range(self.numstates):

            v = self.V[state].copy()
            q = np.sum(self.transitions[state] * \
                ((1-self.pfail[state]) * (self.rewards[state] + self.Discount * np.tile(self.V, (self.numactions, 1))) + \
                (np.tile(self.pfail[state], (self.numactions, 1)) * (self.rewards[state]+self.failReward)) ), axis=1)

            self.V[state] = np.max(q)
            delta = max(delta, np.abs(v - self.V[state]))

        if delta < self.theta:
            break

    # policy improvement
    for state in range(self.numstates):
        if self.resapedMap[state] == 1 or state == self.numstates-1:
            continue

        q = np.sum(self.transitions[state] * \
            ((1-self.pfail[state]) * (self.rewards[state] + self.Discount * np.tile(self.V, (self.numactions, 1))) + \
            (np.tile(self.pfail[state], (self.numactions, 1)) * (self.rewards[state]+self.failReward)) ), axis=1)

        opt_act = np.argmax(q)
        self.probability[state] = np.eye(self.numactions)[opt_act]

    end = time.time()
    print(f'Number of Iterations: {counter} \nTime Elapsed: {(end - start):.3f}s')
```

○ تابع reset

در این تابع نیز مقادیر اولیه ارزش‌ها و سیاست را فراخوانی می‌کنیم.

```
def reset(self):
    self.V = np.zeros(self.numstates)
    self.probability = np.full((self.numstates, self.numactions), fill_value=0.25)
    for s in range(self.numstates):
        if self.resapedMap[s] == 1 or s == self.numstates-1:
            self.probability[s] = 0
```

نتایج

بخش اول

نقشه محیط با اعمال پارامترهای سوال همانند شکل ۸ بدست می‌آید.

در ادامه هر دو الگوریتم policy iteration و value iteration را در این محیط اجرا می‌کنیم. نتایج حاصل در شکل‌های ۹ و ۱۰ قابل مشاهده است. مطابق انتظار هر دو الگوریتم به ارزش‌های نهایی یکسانی همگرا شده و همچنین مسیر بهینه را نیز پیدا کرده‌اند.

تفاوتشان در این است که الگوریتم value iteration تنها در ۱۵ تکرار و ۰.۰۲ ثانیه به جواب رسیده، در حالی که الگوریتم policy iteration بعد از نزدیک به ۱۲۰ تکرار و در ۰.۱۴ ثانیه به جواب رسیده است. یعنی الگوریتم value iteration حدود ۷ برابر سریعتر عمل کرده است.

همچنین مشاهده می‌شود که به غیر از مسیر بهینه، ارزش بقیه استیت‌ها صفر شده‌است، زیرا این استیت‌ها را در واقع به عنوان ترمینال در نظر گرفته‌ایم یعنی وقتی در خانه‌ای می‌رویم و یخ می‌شکند، وارد یک خانه ترمینال غیر از هدف شده و ریوارد منفی ۱۰ می‌گیریم.

Map of Environment

0	1	1	1	1	1
0.0001	1	1	1	1	1
0.0001	1	1	1	1	1
0.0001	1	1	1	1	1
0.0001	0.0001	0.0001	0.0001	1	1
1	1	1	0.0001	0.0001	0

شکل ۸ - نقشه محیط مربوط به بخش اول

Iteration: 1 Number of Evaluation Iterations: 14 ----- Iteration: 2 Number of Evaluation Iterations: 39 ----- Iteration: 3 Number of Evaluation Iterations: 29 ----- Iteration: 4 Number of Evaluation Iterations: 29 ----- Iteration: 5 Number of Evaluation Iterations: 4 ----- Time Elapsed: 0.140s	<table><thead><tr><th></th><th>Left</th><th>Down</th><th>Right</th><th>Up</th></tr></thead><tbody><tr><td>0</td><td>17.233433</td><td>20.013659</td><td>-9.167616</td><td>17.233433</td></tr><tr><td>6</td><td>21.376364</td><td>24.522830</td><td>-9.048534</td><td>16.722636</td></tr><tr><td>12</td><td>25.716737</td><td>29.438069</td><td>-8.777984</td><td>20.723963</td></tr><tr><td>18</td><td>30.666671</td><td>35.048159</td><td>-8.473204</td><td>25.098160</td></tr><tr><td>24</td><td>36.320566</td><td>-8.125150</td><td>41.456179</td><td>30.090904</td></tr><tr><td>25</td><td>34.719173</td><td>-8.802190</td><td>48.775709</td><td>-8.802190</td></tr><tr><td>26</td><td>41.113578</td><td>-8.467751</td><td>58.434369</td><td>-8.467751</td></tr><tr><td>27</td><td>49.519911</td><td>69.696632</td><td>-8.057989</td><td>-8.057989</td></tr><tr><td>33</td><td>-5.906610</td><td>72.775787</td><td>82.804827</td><td>60.995511</td></tr><tr><td>34</td><td>72.593641</td><td>84.479329</td><td>96.039020</td><td>-5.160980</td></tr></tbody></table>		Left	Down	Right	Up	0	17.233433	20.013659	-9.167616	17.233433	6	21.376364	24.522830	-9.048534	16.722636	12	25.716737	29.438069	-8.777984	20.723963	18	30.666671	35.048159	-8.473204	25.098160	24	36.320566	-8.125150	41.456179	30.090904	25	34.719173	-8.802190	48.775709	-8.802190	26	41.113578	-8.467751	58.434369	-8.467751	27	49.519911	69.696632	-8.057989	-8.057989	33	-5.906610	72.775787	82.804827	60.995511	34	72.593641	84.479329	96.039020	-5.160980
	Left	Down	Right	Up																																																				
0	17.233433	20.013659	-9.167616	17.233433																																																				
6	21.376364	24.522830	-9.048534	16.722636																																																				
12	25.716737	29.438069	-8.777984	20.723963																																																				
18	30.666671	35.048159	-8.473204	25.098160																																																				
24	36.320566	-8.125150	41.456179	30.090904																																																				
25	34.719173	-8.802190	48.775709	-8.802190																																																				
26	41.113578	-8.467751	58.434369	-8.467751																																																				
27	49.519911	69.696632	-8.057989	-8.057989																																																				
33	-5.906610	72.775787	82.804827	60.995511																																																				
34	72.593641	84.479329	96.039020	-5.160980																																																				

Values & Policies

20.01 ↓	0.00 .	0.00 .	0.00 .	0.00 .	0.00 .
24.52 ↓	0.00 .	0.00 .	0.00 .	0.00 .	0.00 .
29.44 ↓	0.00 .	0.00 .	0.00 .	0.00 .	0.00 .
35.05 ↓	0.00 .	0.00 .	0.00 .	0.00 .	0.00 .
41.46 →	48.78 →	58.43 →	69.70 ↓	0.00 .	0.00 .
0.00 .	0.00 .	0.00 .	82.80 →	96.04 →	0.00 .

شکل ۹- نتایج الگوریتم **policy iteration** بخش اول

Number of Iterations: 15 Time Elapsed: 0.020s		Left	Down	Right	Up
	0	17.233465	20.013658	-9.167615	17.233465
	6	21.376363	24.522830	-9.048533	16.722668
	12	25.716737	29.438069	-8.777984	20.723962
	18	30.666671	35.048159	-8.473204	25.098159
	24	36.320566	-8.125150	41.456179	30.090904
	25	34.719173	-8.802190	48.775709	-8.802190
	26	41.113578	-8.467751	58.434369	-8.467751
	27	49.519911	69.696632	-8.057989	-8.057989
	33	-5.906610	72.775787	82.804827	60.995511
	34	72.593641	84.479329	96.039020	-5.160980

Values & Policies					
20.01 ↓	0.00 .	0.00 .	0.00 .	0.00 .	0.00 .
24.52 ↓	0.00 .	0.00 .	0.00 .	0.00 .	0.00 .
29.44 ↓	0.00 .	0.00 .	0.00 .	0.00 .	0.00 .
35.05 ↓	0.00 .	0.00 .	0.00 .	0.00 .	0.00 .
41.46 →	48.78 →	58.43 →	69.70 ↓	0.00 .	0.00 .
0.00 .	0.00 .	0.00 .	82.80 →	96.04 →	0.00 .

شكل ١٠- نتائج الـvalue iteration

بخش دوم

در این بخش نیز نقشه محیط با اعمال پارامترهای سوال همانند شکل ۱۱ بدست می‌آید. (احتمال شکست خانه‌های غیر امن به صورت تصادفی انتخاب شده و غیر یک است، احتمال شکست مسیر امن ۰.۰۰۱ است، با احتمال ۰.۷ در جهت عمل انتخاب شده حرکت کرده و در غیر این صورت به طور تصادفی به یکی از خانه‌های مجاور لیز می‌خورد)

در ادامه هر دو الگوریتم policy iteration و value iteration را در این محیط اجرا می‌کنیم. نتایج حاصل در شکل‌های ۱۲ و ۱۳ قابل مشاهده است. مطابق انتظار هر دو الگوریتم به ارزش‌های نهایی یکسانی همگرا شده و همچنین مسیر بهینه را نیز پیدا کرده‌اند.

تفاوتشان در این است که الگوریتم value تنها در ۲۱ تکرار و ۰.۰۶۲ ثانیه به جواب رسیده، در حالی که الگوریتم policy بعد از نزدیک به ۹۰ تکرار و در ۰.۲۴۸ ثانیه به جواب رسیده است. یعنی الگوریتم value iteration حدود ۴ برابر سریعتر عمل کرده است.

در مقایسه با بخش قبل مشاهده می‌شود که در این بخش، ارزش خانه‌های مسیر غیر امن برورسانی شده و برخلاف بخش قبل صفر نشده‌اند، زیرا احتمال شکست این خانه‌ها بر خلاف بخش قبلی کمتر از یک است. لذا با احتمال $1-P_{fail}$ با این خانه‌ها مشابه خانه امن رفتار می‌شود. تفاوت دیگر آن که در بخش دوم ارزش خانه‌های دورتر از هدف کمتر شده است زیرا احتمال لیز خوردن در اینجا بیشتر شده و احتمال بیشتری از مسیر امن خارج می‌شویم. همچنین زمان اجرای هر دو الگوریتم نسبت به بخش قبلی ۲ تا ۳ برابر شده است.

Map of Environment

0	0.052	0.16	0.91	0.11	0.12
0.001	0.55	0.64	0.86	0.19	0.42
0.001	0.28	0.013	0.36	0.44	0.88
0.001	0.57	0.46	0.15	0.054	0.95
0.001	0.001	0.001	0.001	0.42	0.52
0.65	0.68	0.55	0.001	0.001	0

شکل ۱۱ - نقشه محیط مربوط به بخش دوم

Iteration: 1 Number of Evaluation Iterations: 18 ----- Iteration: 2 Number of Evaluation Iterations: 28 ----- Iteration: 3 Number of Evaluation Iterations: 22 ----- Iteration: 4 Number of Evaluation Iterations: 11 ----- Iteration: 5 Number of Evaluation Iterations: 7 ----- Time Elapsed: 0.248s																																																																													
<div> <div>Values & Policies</div> <table> <tr><td>-0.06</td><td>-2.24</td><td>-4.66</td><td>-6.07</td><td>-5.75</td><td>-6.03</td></tr> <tr><td>↓</td><td>←</td><td>←</td><td>↑</td><td>↓</td><td>→</td></tr> <tr><td>1.85</td><td>-0.93</td><td>-1.24</td><td>0.13</td><td>-2.84</td><td>-5.68</td></tr> <tr><td>↓</td><td>←</td><td>↓</td><td>↓</td><td>↓</td><td>←</td></tr> <tr><td>5.06</td><td>1.76</td><td>2.90</td><td>14.18</td><td>9.60</td><td>-2.38</td></tr> <tr><td>↓</td><td>←</td><td>↓</td><td>↓</td><td>↓</td><td>←</td></tr> <tr><td>8.76</td><td>13.02</td><td>21.89</td><td>31.47</td><td>19.83</td><td>21.46</td></tr> <tr><td>↓</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td></tr> <tr><td>13.50</td><td>20.33</td><td>32.54</td><td>47.30</td><td>61.39</td><td>78.04</td></tr> <tr><td>→</td><td>→</td><td>→</td><td>↓</td><td>↓</td><td>↓</td></tr> <tr><td>8.35</td><td>14.10</td><td>46.27</td><td>64.14</td><td>85.31</td><td>0.00</td></tr> <tr><td>↑</td><td>↑</td><td>→</td><td>→</td><td>→</td><td>↓</td></tr> </table> </div>						-0.06	-2.24	-4.66	-6.07	-5.75	-6.03	↓	←	←	↑	↓	→	1.85	-0.93	-1.24	0.13	-2.84	-5.68	↓	←	↓	↓	↓	←	5.06	1.76	2.90	14.18	9.60	-2.38	↓	←	↓	↓	↓	←	8.76	13.02	21.89	31.47	19.83	21.46	↓	↓	↓	↓	↓	↓	13.50	20.33	32.54	47.30	61.39	78.04	→	→	→	↓	↓	↓	8.35	14.10	46.27	64.14	85.31	0.00	↑	↑	→	→	→	↓
-0.06	-2.24	-4.66	-6.07	-5.75	-6.03																																																																								
↓	←	←	↑	↓	→																																																																								
1.85	-0.93	-1.24	0.13	-2.84	-5.68																																																																								
↓	←	↓	↓	↓	←																																																																								
5.06	1.76	2.90	14.18	9.60	-2.38																																																																								
↓	←	↓	↓	↓	←																																																																								
8.76	13.02	21.89	31.47	19.83	21.46																																																																								
↓	↓	↓	↓	↓	↓																																																																								
13.50	20.33	32.54	47.30	61.39	78.04																																																																								
→	→	→	↓	↓	↓																																																																								
8.35	14.10	46.27	64.14	85.31	0.00																																																																								
↑	↑	→	→	→	↓																																																																								
	Left	Down	Right	Up																																																																									
0	-0.457867	-0.064810	-2.315360	-0.457867																																																																									
1	-2.244376	-5.726671	-5.300571	-2.821296																																																																									
2	-4.664813	-7.277585	-8.968898	-5.122424																																																																									
3	-6.480573	-8.521305	-6.821992	-6.069067																																																																									
4	-9.137203	-5.745040	-6.960520	-5.874012																																																																									
5	-6.742627	-7.532218	-6.031737	-6.031737																																																																									
6	0.729479	1.853180	-4.388606	-0.906311																																																																									
7	-0.929113	-2.917772	-5.997031	-3.384258																																																																									
8	-6.228508	-1.237228	-7.843140	-5.802408																																																																									
9	-6.657406	0.130753	-4.956555	-8.348719																																																																									
10	-8.231579	-2.838190	-7.393637	-6.532265																																																																									
11	-5.680882	-8.722755	-5.775197	-6.896362																																																																									
12	3.675033	5.061372	-0.653060	1.335599																																																																									
13	1.763682	-1.405986	0.513177	-4.478103																																																																									
14	-1.773596	2.904731	1.935304	-4.852855																																																																									
15	2.156929	14.177938	0.944406	-4.448982																																																																									
16	2.503084	9.598616	-5.626098	-2.584225																																																																									
17	-2.377381	-7.754831	-3.322544	-6.932828																																																																									
18	6.811493	8.759646	1.036022	4.205691																																																																									
19	6.781092	13.023398	5.744986	1.066659																																																																									
20	3.899550	21.890145	17.839722	5.818713																																																																									
21	9.636626	31.465981	15.762732	8.667200																																																																									
22	16.704408	19.826280	-1.906573	3.470876																																																																									
23	14.387885	21.464031	16.768984	-0.836829																																																																									
24	10.432928	0.216016	13.503060	7.260754																																																																									
25	10.064071	1.121545	20.331043	2.340447																																																																									
26	17.984060	14.998044	32.535002	10.705648																																																																									
27	30.254079	47.304206	29.325527	26.203656																																																																									
28	40.883577	61.388123	32.256474	25.180328																																																																									
29	34.657203	78.040423	60.782251	12.924351																																																																									
30	6.365964	6.365964	0.155724	8.353040																																																																									
31	0.812406	11.351370	11.113435	14.099451																																																																									
32	10.013156	37.264623	46.272780	29.222654																																																																									
33	26.099413	53.359739	64.140917	43.636371																																																																									
34	59.909056	71.982939	85.313597	41.930378																																																																									
35	0.000000	0.000000	0.000000	0.000000																																																																									

شکل ۱۲- نتایج الگوریتم policy iteration بخش دوم

Number of Iterations: 21
Time Elapsed: 0.062s

Values & Policies

-0.07 ↓	-2.25 ←	-4.67 ←	-6.08 ↑	-5.74 ↓	-6.03 →
1.85 ↓	-0.93 ←	-1.24 ↓	0.13 ↓	-2.84 ↓	-5.68 ←
5.06 ↓	1.76 ←	2.90 ↓	14.18 ↓	9.60 ↓	-2.38 ←
8.76 ↓	13.02 ↓	21.89 ↓	31.47 ↓	19.83 ↓	21.46 ↓
13.50 →	20.33 →	32.53 →	47.30 ↓	61.39 ↓	78.04 ↓
8.35 ↑	14.10 ↑	46.27 →	64.14 →	85.31 →	0.00 ↓

	Left	Down	Right	Up
0	-0.462414	-0.067440	-2.319635	-0.462414
1	-2.248845	-5.728474	-5.303798	-2.825622
2	-4.668468	-7.278539	-8.969865	-5.125496
3	-6.482916	-8.521806	-6.822440	-6.070755
4	-9.137250	-5.744978	-6.960089	-5.873899
5	-6.742456	-7.532069	-6.031221	-6.031221
6	0.727682	1.852074	-4.389827	-0.910199
7	-0.930782	-2.918617	-5.997793	-3.387722
8	-6.229311	-1.237640	-7.843526	-5.804635
9	-6.657526	0.130719	-4.956580	-8.348852
10	-8.231569	-2.838178	-7.393599	-6.532203
11	-5.680805	-8.722684	-5.775082	-6.895916
12	3.674469	5.061033	-0.653490	1.334345
13	1.763250	-1.406128	0.513021	-4.478650
14	-1.773812	2.904683	1.935257	-4.852988
15	2.156895	14.177932	0.944401	-4.448991
16	2.503081	9.598616	-5.626097	-2.584218
17	-2.377377	-7.754826	-3.322537	-6.932798
18	6.811342	8.759553	1.035938	4.205316
19	6.780967	13.023348	5.744939	1.066444
20	3.899528	21.890136	17.839714	5.818677
21	9.636621	31.465979	15.762731	8.667196
22	16.704407	19.826279	-1.906573	3.470876
23	14.387885	21.464031	16.768984	-0.836828
24	10.432886	0.215989	13.503035	7.260654
25	10.064042	1.121537	20.331035	2.340427
26	17.984052	14.998042	32.535000	10.705642
27	30.254077	47.304205	29.325527	26.203654
28	40.883577	61.388123	32.256474	25.180328
29	34.657203	78.040423	60.782251	12.924351
30	6.365932	6.365932	0.155711	8.353008
31	0.812396	11.351360	11.113432	14.099441
32	10.013153	37.264621	46.272780	29.222651
33	26.099413	53.359739	64.140917	43.636370
34	59.909056	71.982939	85.313597	41.930377
35	0.000000	0.000000	0.000000	0.000000

شکل ۱۳- نتایج الگوریتم value iteration بخش دوم

بخش سوم

با تغییر ابعاد نقشه به ۱۵ در ۱۵ و پارامترهای مشابه بخش دوم به نقشه شکل ۱۴ خواهیم رسید.

Map of Environment

0	0.001	0.73	0.41	0.16	0.67	0.17	0.77	0.86	0.57	0.38	0.91	0.85	0.17	0.95
0.93	0.001	0.46	0.48	0.33	0.24	0.67	0.46	0.97	0.66	0.81	0.84	0.33	0.87	0.35
0.46	0.001	0.001	0.68	0.26	0.69	0.43	0.3	0.16	0.25	0.8	0.16	0.35	0.003	0.35
0.26	0.41	0.001	0.001	0.001	0.001	0.39	0.11	0.77	0.34	0.86	0.59	0.0099	0.36	0.19
0.99	0.53	0.27	0.74	0.44	0.001	0.72	0.65	0.36	0.5	0.92	0.62	0.22	0.6	0.18
0.017	0.25	0.37	0.23	0.92	0.001	0.001	0.001	0.54	0.81	0.82	0.71	0.3	0.78	0.97
0.67	0.28	0.86	0.33	0.43	0.59	0.44	0.001	0.1	0.92	0.29	0.49	0.25	0.56	0.037
0.46	0.63	0.76	0.42	0.33	0.11	0.34	0.001	0.86	0.75	0.31	0.44	0.94	0.48	0.69
0.57	0.26	0.2	0.57	0.93	0.95	0.13	0.001	0.001	0.93	0.9	0.88	0.31	0.66	0.098
0.23	0.61	0.45	0.89	0.82	0.59	0.17	0.87	0.001	0.72	0.51	0.33	0.96	0.29	0.29
0.032	0.24	0.12	0.67	0.51	0.98	0.79	0.77	0.001	0.41	0.22	0.15	0.54	0.73	0.71
0.32	0.61	0.38	0.73	0.89	0.12	0.17	0.62	0.001	0.001	0.001	0.001	0.001	0.42	0.29
0.5	0.27	0.71	0.065	0.35	0.092	0.31	0.65	0.027	0.63	0.86	0.67	0.001	0.29	0.043
0.5	0.83	0.15	0.077	0.6	0.24	0.82	0.17	0.46	0.28	0.69	0.88	0.001	0.22	0.15
0.26	0.42	0.46	0.51	0.53	0.42	0.4	0.18	0.31	0.49	0.77	0.028	0.001	0.001	0

شکل ۱۴- نقشه محیط مربوط به بخش سوم

الف

چنان چه در شکل‌های ۱۵ تا ۱۷ قابل مشاهده است، با تغییر ترشولد در هیچ کدام از حالات به سیاست نهایی نرسیده‌ایم و تنها در بخشی از استیتهای سیاست بهینه یافت شده‌است. در واقع کوچک بودن پاداش خانه هدف، احتمال بالای لیز خوردن، تعداد زیاد خانه‌های با احتمال شکست و پاداش منفی این خانه‌ها و همچنین فاصله زیاد استیت شروع از هدف، باعث شده است که تاثیر مثبت استیت هدف به خوبی تا خانه شروع گسترش نیابد.

همچنین با کاهش ترشولد نه تنها به سیاست بهینه نرسیده‌ایم بلکه تعداد تکرارها نیز افزایش یافته‌است، به عبارت دیگر کاهش ترشولد در اینجا باعث افزایش بی دلیل حجم محاسبات شده است.

انتخاب مقدار ترشولد: این مقدار نباید خیلی کوچک باشد زیرا مانند این سوال بی دلیل فقط باعث افزایش حجم محاسبات خواهد شد، همچنین مقدار بزرگ آن باعث می‌شود که الگوریتم قبل از یافتن سیاست بهینه خاتمه یابد.

Values & Policies														
-7.00 ◀	-6.65 ▲	-6.82 ▲	-7.59 ▲	-7.63 ▲	-7.27 ▲	-8.23 ▲	-8.09 ▲	-8.40 ▲	-8.29 ▲	-8.46 ▲	-8.33 ▲	-8.13 ▲	-8.67 ▲	-7.36 ▶
-6.44 ◀	-7.79 ▲	-8.53 ◀	-9.38 ▲	-8.92 ▲	-9.88 ◀	-9.35 ▲	-10.47 ▲	-10.67 ▲	-10.19 ▲	-9.92 ▲	-10.41 ▼	-10.20 ▼	-9.24 ▼	-8.08 ▶
-7.49 ◀	-8.42 ▲	-8.96 ◀	-9.35 ◀	-9.88 ▲	-10.12 ▼	-10.53 ▲	-10.62 ▼	-10.75 ◀	-10.75 ▲	-10.45 ▶	-10.21 ▶	-9.40 ▶	-9.01 ▶	-6.98 ▶
-8.02 ◀	-8.87 ▲	-9.30 ▲	-9.66 ◀	-9.86 ◀	-10.05 ◀	-10.24 ◀	-10.60 ◀	-10.64 ◀	-10.80 ◀	-10.67 ▶	-10.00 ▶	-9.70 ▶	-8.50 ▶	-7.07 ▶
-7.36 ◀	-9.74 ▼	-9.70 ▲	-9.90 ▲	-10.09 ▲	-10.17 ▲	-10.25 ◀	-10.32 ▼	-10.72 ▼	-10.82 ◀	-10.79 ▶	-10.37 ▶	-9.97 ▲	-9.03 ▶	-7.81 ▶
-8.14 ◀	-8.94 ◀	-9.83 ◀	-10.42 ◀	-10.31 ▶	-10.29 ▲	-10.27 ▶	-10.26 ▼	-10.37 ◀	-10.76 ◀	-10.84 ▼	-10.61 ▶	-10.36 ▲	-10.32 ▼	-7.46 ▶
-7.46 ◀	-9.81 ▲	-10.29 ◀	-10.61 ▲	-10.71 ▼	-10.36 ▲	-10.23 ▶	-10.24 ▲	-10.40 ◀	-10.57 ◀	-10.80 ▶	-10.59 ▶	-10.38 ▶	-9.18 ▶	-8.33 ▶
-8.03 ◀	-9.77 ◀	-10.50 ▼	-10.71 ▶	-10.61 ▶	-10.55 ▶	-10.26 ▶	-10.27 ▼	-9.85 ▼	-10.84 ▶	-10.79 ▶	-10.62 ▼	-10.53 ▲	-10.06 ▶	-7.22 ▶
-7.39 ◀	-9.79 ◀	-10.28 ◀	-10.55 ◀	-10.76 ▲	-10.45 ▶	-10.31 ▶	-9.83 ▶	-8.73 ▼	-9.38 ◀	-8.69 ▼	-6.65 ▼	-10.37 ▶	-8.61 ▶	-7.43 ▶
-7.11 ◀	-8.69 ◀	-9.85 ▼	-10.51 ◀	-10.85 ▶	-10.66 ▶	-10.51 ▶	-8.64 ▶	-6.82 ▼	-6.76 ▼	-3.65 ▼	-0.83 ▼	-3.60 ▼	-8.18 ▶	-5.72 ▼
-6.75 ◀	-8.04 ◀	-9.35 ◀	-9.98 ◀	-10.75 ◀	-10.50 ▼	-10.21 ▼	-6.93 ▼	-4.26 ▼	-1.20 ▼	2.59 ▶	7.21 ▶	11.64 ▶	0.99 ▶	11.77 ▶
-7.10 ◀	-8.91 ◀	-9.89 ▲	-10.24 ▼	-10.45 ▶	-10.23 ▶	-9.53 ▶	4.90 ▶	-1.48 ▶	2.04 ▶	7.01 ▶	13.88 ▶	22.12 ▶	17.19 ▶	33.68 ▶
-7.46 ◀	-9.71 ◀	-9.92 ▼	-10.04 ▼	-10.22 ◀	-10.03 ▼	-9.84 ▶	-6.52 ▶	-4.25 ▲	-1.75 ▲	1.85 ▲	20.36 ▶	34.14 ▶	38.49 ▶	52.69 ▶
-7.62 ◀	-9.33 ▼	-9.59 ▼	-9.66 ▼	-9.71 ▼	-9.52 ▼	-9.18 ▼	-8.43 ▼	-6.11 ▲	-7.99 ▼	4.38 ▶	33.17 ▶	49.36 ▶	67.16 ▶	85.79 ▶
-7.42 ◀	-7.61 ▼	-7.59 ▼	-7.65 ▼	-7.78 ▼	-7.59 ▼	-7.40 ▼	-7.00 ▼	-6.60 ▼	-4.00 ▶	28.03 ▶	45.87 ▶	68.88 ▶	87.66 ▶	0.00 *

شکل ۱۵- ارزش و سیاست نهایی مربوط به بخش سوم ($\theta=1$)

Values & Policies														
-7.30 ◀	-6.98 ▲	-7.11 ▲	-7.77 ▲	-7.81 ▲	-7.47 ▲	-8.33 ▲	-8.19 ▲	-8.47 ▲	-8.37 ▲	-8.52 ▲	-8.40 ▲	-8.24 ▲	-8.72 ▲	-7.51 ▶
-6.79 ◀	-8.03 ▲	-8.71 ▲	-9.47 ▲	-9.02 ▲	-9.94 ▲	-9.41 ▲	-10.49 ▲	-10.68 ▲	-10.22 ▲	-9.94 ▲	-10.48 ▼	-10.29 ▼	-9.36 ▲	-8.21 ▶
-7.71 ◀	-8.60 ▲	-9.10 ▲	-9.46 ▲	-9.94 ▲	-10.18 ▼	-10.54 ▲	-10.63 ▼	-10.75 ▲	-10.75 ▲	-10.52 ▶	-10.30 ▶	-9.55 ▶	-9.17 ▶	-7.24 ▶
-8.16 ◀	-9.02 ▲	-9.41 ▲	-9.75 ▲	-9.93 ▲	-10.10 ▲	-10.27 ▲	-10.60 ▲	-10.64 ▲	-10.80 ▲	-10.71 ▶	-10.10 ▶	-9.82 ▶	-8.69 ▶	-7.30 ▶
-7.55 ◀	-9.81 ▼	-9.78 ▲	-9.96 ▲	-10.14 ▲	-10.20 ▲	-10.06 ▼	-9.97 ▼	-10.60 ▼	-10.76 ▲	-10.81 ▶	-10.43 ▶	-10.05 ▲	-9.14 ▶	-7.94 ▶
-8.25 ◀	-9.02 ▲	-9.87 ▲	-10.44 ▲	-10.25 ▶	-10.10 ▶	-9.88 ▶	-9.63 ▼	-9.92 ▲	-10.63 ▲	-10.84 ▼	-10.64 ▶	-10.40 ▲	-10.36 ▼	-7.61 ▶
-7.63 ◀	-9.86 ▲	-10.32 ▲	-10.62 ▲	-10.63 ▼	-10.18 ▲	-9.57 ▶	-9.25 ▼	-9.74 ▲	-10.19 ▲	-10.81 ▶	-10.61 ▶	-10.42 ▶	-9.25 ▶	-8.41 ▶
-8.15 ◀	-9.82 ▲	-10.52 ▼	-10.64 ▶	-10.40 ▶	-10.14 ▶	-9.24 ▶	-8.84 ▼	-8.27 ▼	-10.70 ▲	-10.76 ▶	-10.60 ▼	-10.55 ▲	-10.11 ▶	-7.36 ▶
-7.60 ◀	-9.86 ▲	-10.32 ▲	-10.57 ▲	-10.67 ▲	-9.69 ▶	-8.93 ▶	-8.18 ▼	-7.29 ▼	-8.45 ▲	-8.53 ▼	-6.56 ▼	-10.39 ▶	-8.67 ▶	-7.50 ▶
-7.37 ◀	-8.84 ▲	-9.94 ▼	-10.54 ▲	-10.75 ▶	-10.25 ▶	-9.70 ▲	-7.13 ▶	-5.73 ▼	-6.33 ▼	-3.45 ▼	-0.74 ▼	-3.58 ▼	-8.18 ▶	-5.72 ▼
-7.05 ◀	-8.24 ▲	-9.46 ▲	-10.04 ▲	-10.76 ▲	-10.25 ▼	-9.77 ▼	-5.93 ▶	-3.54 ▼	-0.84 ▼	2.74 ▼	7.27 ▼	11.66 ▼	0.99 ▼	11.77 ▼
-7.34 ◀	-9.03 ▲	-9.96 ▲	-10.30 ▼	-10.24 ▶	-9.88 ▶	-9.16 ▶	4.27 ▶	-1.07 ▶	2.22 ▶	7.08 ▶	13.90 ▶	22.13 ▼	17.19 ▼	33.68 ▼
-7.65 ◀	-9.78 ▲	-10.00 ▼	-10.11 ▼	-10.24 ▶	-10.03 ▼	-9.62 ▶	-6.09 ▶	-3.95 ▲	-1.61 ▲	1.89 ▲	20.37 ▶	34.14 ▼	38.49 ▶	52.69 ▼
-7.79 ◀	-9.43 ▼	-9.68 ▼	-9.74 ▼	-9.79 ▼	-9.60 ▼	-9.23 ▶	-8.47 ▼	-5.92 ▲	-7.97 ▼	4.38 ▼	33.17 ▶	49.36 ▼	67.16 ▼	85.79 ▼
-7.60 ◀	-7.77 ▼	-7.76 ▼	-7.80 ▼	-7.92 ▼	-7.75 ▼	-7.56 ▼	-7.09 ▼	-6.55 ▼	-3.99 ▶	28.03 ▶	45.87 ▶	68.88 ▶	87.66 ▶	0.00 •

شکل ۱۶- ارزش و سیاست نهایی مربوط به بخش سوم ($\theta=0.01$)

Values & Policies														
-7.31 ▲	-6.99 ▲	-7.12 ▲	-7.78 ▲	-7.81 ▲	-7.47 ▲	-8.33 ▲	-8.19 ▲	-8.47 ▲	-8.37 ▲	-8.52 ▲	-8.40 ▲	-8.24 ▲	-8.72 ▲	-7.52 ▲
-6.80 ▲	-8.04 ▲	-8.71 ▲	-9.47 ▲	-9.03 ▲	-9.94 ▲	-9.41 ▲	-10.49 ▲	-10.68 ▲	-10.22 ▲	-9.94 ▲	-10.48 ▼	-10.29 ▼	-9.36 ▲	-8.22 ▲
-7.71 ▲	-8.61 ▲	-9.11 ▲	-9.46 ▲	-9.94 ▲	-10.18 ▼	-10.54 ▲	-10.63 ▼	-10.75 ▲	-10.75 ▲	-10.52 ▲	-10.30 ▲	-9.56 ▲	-9.18 ▲	-7.24 ▲
-8.16 ▲	-9.02 ▲	-9.42 ▲	-9.75 ▲	-9.93 ▲	-10.10 ▲	-10.27 ▲	-10.60 ▲	-10.64 ▲	-10.80 ▲	-10.71 ▲	-10.11 ▲	-9.82 ▲	-8.69 ▲	-7.31 ▲
-7.56 ▲	-9.81 ▼	-9.78 ▲	-9.96 ▲	-10.14 ▲	-10.20 ▼	-10.06 ▼	-9.97 ▼	-10.59 ▼	-10.76 ▲	-10.81 ▲	-10.43 ▲	-10.06 ▲	-9.14 ▲	-7.95 ▲
-8.25 ▲	-9.02 ▲	-9.87 ▲	-10.44 ▲	-10.24 ▲	-10.10 ▲	-9.87 ▲	-9.63 ▲	-9.92 ▲	-10.63 ▲	-10.84 ▲	-10.64 ▲	-10.41 ▲	-10.36 ▲	-7.62 ▲
-7.63 ▲	-9.86 ▲	-10.32 ▲	-10.62 ▲	-10.63 ▲	-10.18 ▲	-9.57 ▲	-9.25 ▲	-9.74 ▲	-10.19 ▲	-10.81 ▲	-10.61 ▲	-10.42 ▲	-9.25 ▲	-8.41 ▲
-8.15 ▲	-9.83 ▲	-10.53 ▼	-10.64 ▲	-10.40 ▲	-10.14 ▲	-9.24 ▲	-8.84 ▲	-8.27 ▲	-10.70 ▲	-10.76 ▲	-10.60 ▲	-10.55 ▲	-10.11 ▲	-7.37 ▲
-7.61 ▲	-9.86 ▲	-10.32 ▲	-10.57 ▲	-10.67 ▲	-9.69 ▲	-8.93 ▲	-8.18 ▲	-7.29 ▲	-8.45 ▲	-8.53 ▲	-6.56 ▲	-10.39 ▲	-8.67 ▲	-7.50 ▲
-7.38 ▲	-8.85 ▲	-9.94 ▼	-10.54 ▲	-10.75 ▲	-10.25 ▲	-9.70 ▲	-7.13 ▲	-5.73 ▲	-6.33 ▲	-3.45 ▲	-0.74 ▲	-3.58 ▲	-8.18 ▲	-5.72 ▲
-7.06 ▲	-8.25 ▲	-9.47 ▲	-10.05 ▲	-10.76 ▲	-10.25 ▲	-9.77 ▲	-5.93 ▲	-3.54 ▲	-0.84 ▲	2.74 ▲	7.27 ▲	11.66 ▲	0.99 ▲	11.77 ▲
-7.35 ▲	-9.04 ▲	-9.97 ▲	-10.30 ▲	-10.24 ▲	-9.88 ▲	-9.16 ▲	4.27 ▲	-1.07 ▲	2.22 ▲	7.08 ▲	13.90 ▲	22.13 ▲	17.19 ▲	33.68 ▲
-7.65 ▲	-9.78 ▲	-10.00 ▲	-10.11 ▲	-10.24 ▲	-10.04 ▲	-9.62 ▲	-6.09 ▲	-3.95 ▲	-1.61 ▲	1.89 ▲	20.37 ▲	34.14 ▲	38.49 ▲	52.69 ▲
-7.80 ▲	-9.43 ▲	-9.68 ▲	-9.75 ▲	-9.79 ▲	-9.60 ▲	-9.23 ▲	-8.48 ▲	-5.92 ▲	-7.97 ▲	4.38 ▲	33.17 ▲	49.36 ▲	67.16 ▲	85.79 ▲
-7.60 ▲	-7.77 ▲	-7.76 ▲	-7.80 ▲	-7.92 ▲	-7.75 ▲	-7.56 ▲	-7.10 ▲	-6.55 ▲	-3.99 ▲	28.03 ▲	45.87 ▲	68.88 ▲	87.66 ▲	0.00 ▲

شکل ۱۷- ارزش و سیاست نهایی مربوط به بخش سوم ($\theta=0.000001$)

ب

همانطور که در قسمت قبل گفته شد، پاداش خانه هدف به اندازه کافی بزرگ نبود تا بتواند تاثیر مثبت آن تا خانه شروع گسترش یابد، لذا فقط برای بخشی از انتهای مسیر، سیاست بهینه یافت شد.

در این قسمت می‌خواهیم با افزایش مقدار پاداش هدف کاری کنیم تا تاثیر آن تا نقطه شروع گسترش یافته و سیاست بهینه را پیدا کنیم. با آزمودن مقادیر مختلف دیدیم که با پاداش هدف ۲۰۰۰۰ بالاخره این امر محقق شده و سیاست بهینه یافت می‌شود. با توجه به شکل ۱۵ می‌بینیم که اثرات این افزایش پاداش به وضوح در مقادیر ارزش‌های نهایی بدست آمده قابل مشاهده است.

Values & Policies														
-6.05 ▶	-4.66 ▼	-5.33 ▲	-6.29 ▲	-4.34 ▼	-5.42 ▲	-7.68 ▲	-7.05 ▲	-8.35 ▲	-8.21 ▲	-8.50 ▲	-8.39 ▲	-8.22 ▲	-8.72 ▲	-7.51 ▶
4.38 ▶	-3.07 ▼	-0.95 ▼	4.16 ▶	2.04 ▼	-3.40 ▼	4.18 ▼	-2.86 ▼	-5.84 ▼	-8.35 ▼	-9.84 ▲	-10.35 ▼	-10.14 ▼	-9.35 ▲	-8.21 ▶
-2.88 ▶	-0.28 ▶	2.97 ▼	8.25 ▼	13.84 ▼	21.52 ▼	4.64 ▼	6.76 ▼	-2.39 ◀	-5.85 ◀	-8.36 ◀	-10.09 ▶	-9.20 ▼	-9.13 ▶	-7.21 ▶
-4.78 ▶	2.64 ▶	7.86 ▶	14.30 ▶	24.75 ▶	35.79 ▼	23.84 ◀	17.20 ▼	8.53 ▼	-4.77 ▼	-8.22 ◀	-9.32 ▶	-8.74 ▼	-8.51 ▶	-7.19 ▶
-6.22 ◀	-3.92 ▶	2.10 ▲	8.63 ▲	33.88 ▶	53.61 ▼	79.60 ▼	102.51 ▼	29.20 ▼	6.28 ◀	-5.32 ◀	-8.73 ▶	-7.18 ▼	-8.38 ◀	-7.54 ▶
-7.95 ◀	-7.48 ▶	4.08 ▲	-3.97 ▼	49.26 ▶	77.65 ▶	114.93 ▶	158.39 ▼	111.59 ◀	26.09 ◀	-2.80 ▼	-2.29 ▼	-3.30 ▼	2.30 ▼	34.37 ▼
-7.40 ◀	-9.02 ▲	-4.51 ▶	2.44 ▶	20.41 ▶	70.50 ▶	167.24 ▶	226.34 ▼	149.55 ◀	82.38 ◀	5.12 ▼	13.83 ▼	3.45 ▶	34.43 ▶	55.90 ▼
-8.03 ◀	-8.89 ▶	-0.13 ▶	17.58 ▶	53.30 ▶	97.73 ▶	230.85 ▶	306.85 ▼	411.41 ▶	31.76 ◀	21.61 ▼	52.33 ▼	31.01 ▼	64.02 ▼	286.77 ▼
-7.40 ◀	-8.87 ▶	-7.50 ▶	-0.06 ▲	17.83 ▶	169.46 ▶	294.61 ▶	428.60 ▼	592.50 ▼	393.74 ◀	411.04 ▼	759.46 ▼	67.54 ▶	288.83 ▶	449.05 ▼
-7.33 ◀	-8.69 ◀	-8.86 ▲	-7.34 ▶	14.65 ▶	85.49 ▶	171.19 ▶	615.92 ▶	872.26 ▼	787.73 ▼	1305.89 ▼	1797.48 ▼	1299.96 ▼	457.27 ▶	884.30 ▼
-7.00 ◀	-8.11 ◀	-8.66 ▼	-6.91 ▼	-3.55 ▼	89.85 ▶	178.99 ▶	840.48 ▶	1263.02 ▼	1753.98 ▼	2407.41 ▼	3232.30 ▼	4037.70 ▼	2131.68 ▼	4047.77 ▼
-7.18 ◀	-8.24 ▼	-6.26 ▼	9.90 ▶	87.56 ▶	158.18 ▶	292.12 ▶	1138.20 ▶	1709.19 ▶	2310.61 ▶	3195.15 ▶	4436.00 ▶	5935.34 ▼	5059.15 ▼	8029.22 ▼
-7.02 ◀	-6.64 ▶	10.24 ▶	22.06 ▶	61.92 ▶	106.99 ▶	206.55 ▶	804.60 ▶	1193.23 ▲	1618.00 ▲	2262.56 ▲	5619.93 ▶	8119.12 ▼	8920.28 ▶	11506.34 ▼
-7.37 ◀	-2.38 ▶	3.63 ▶	12.02 ▶	33.24 ▶	65.33 ▶	190.62 ▶	322.09 ▶	815.67 ▶	501.80 ▶	1168.99 ▶	7946.40 ▶	10888.80 ▼	14150.47 ▶	17560.54 ▼
-7.26 ◀	-6.54 ▼	-1.09 ▲	5.99 ▲	22.33 ▶	64.77 ▶	152.14 ▶	261.45 ▶	478.32 ▶	1193.96 ▶	6982.57 ▶	10237.24 ▶	14437.26 ▶	17897.49 ▶	0.00 .

شکل ۱۸ - ارزش و سیاست نهایی مربوط به بخش سوم (theta=0.01 , goal reward=20000)

روند اجرای کد پیاده‌سازی

تمام توابع تعریف شده در فایل `< env.py >` قرار داشته و اجرای آنها و نمایش نمودارها در فایل `< HW3_Q3.ipynb >` قابل مشاهده است. تنها لازم است که تمام سلول‌ها در فایل دوم به ترتیب از ابتدا اجرا شوند تا نتایج و نمودارها به دست آیند.

