گزارش تمرین شماره ۳

نام و نام خانوادگی	علی عدالت
شماره دانشجویی	۸۱۰۱۹۹۳۴۸

سوال 1 – بازار بورس

در این جا ما یک مسئله‌ی MDP داریم که state ها و اکشن‌ها برای ما به عنوان عامل مشخص است ولی $P_{ss\acute{s}}^{a}$ و $R_{ss\acute{s}}^{a}$ مشخص نیستند. به همین دلیل ما با مسئله یادگیری مواجه هستیم. State در این مسئله برابر مجموعه‌ی میزان سرمایه اولیه‌ی ما و قیمت هر سهم هر شرکت است. به صورت زیر یک state را می‌توان تعریف کرد.

$$state : \{cash, B\_price, C\_price, D\_price\}$$

Cash برابر موجودی ما در حساب بانکی و سرمایه اولیه ما است. B_price ارزش یک سهم شرکت B است. C_price ارزش یک سهم شرکت C است. D_price ارزش یک سهم شرکت D است. با توجه به این موضوع حالت‌های ممکن برای سرمایه اولیه‌ی ما از ۰ تا ۱۰۰ دلار و حالت‌های ممکن برای قیمت هر سهم هر شرکت از ۵ تا ۵۰ دلار سازنده‌ی state های مسئله است. ضرایب ۵ از ۰ تا خود ۱۰۰ حالت‌های ممکن برای سرمایه اولیه و ضرایب ۵ از ۵ تا خود ۵۰ حالت‌های ممکن برای ارزش هر سهم برای هر شرکت است. در کل ما ۲۱۰۰۰ حالت برای state داریم. در این جا state اولیه برابر (4,15,15,10) است که اولین عدد از چپ نشان دهنده‌ی سرمایه‌ی ما یعنی ۲۰ دلار تفسیم بر ۵ است. اعداد دیگر قیمت سهم شرکت‌ها به ترتیبی است که در تعریف state در بالا آمده است. state نهایی هدف یا ترمینال هدف برابر تمام حالاتی است که سرمایه‌ی ما ۱۰۰ دلار است. زمانی که ما وارد ترمینال هدف می‌شویم، هیچ وقت از آن خارج نخواهیم شد. حالت‌هایی که سرمایه‌ی ما صفر است، برابر state چاه است که زمانی که به این stateها رفتیم دیگر نمی‌توانیم از آن خارج شویم. به همین دلیل این stateها نیز یک ترمینال محسوب می‌شود که هدف ما وارد نشدن به آن است. در این مسئله هدف ما رسیدن هر چه سریع تر به ۱۰۰ دلار و state نهایی هدف است به این صورت که در مسیر رسیدن به آن هرگز وارد چاه نشویم. با توجه به این موضوع هر بار تصمیم گیری در یک state و ماندن در آن، برای ما هزینه دارد. اگر در state ای قرار داشته باشیم و با انجام اکشن به ترمینال هدف برسیم، علاوه بر جزای مربوط به تصمیم گیری و جا به جایی، مقدار ۲۰۰۰۰۰۰۰ واحد پاداش دریافت می‌کنیم و پاداش کل ما ۱۹۹۹۹۹۹ واحد است. همچنین اگر در state ای قرار داشته باشیم و با انجام اکشن به چاه برسیم، علاوه بر جزای مربوط به تصمیم گیری، مقدار ۲۰۰۰۰۰۰۰ واحد جزای بیشتر دریافت می‌کنیم و جزای کل برابر ۲۰۰۰۰۰۰۱ واحد است. در هر state می‌توانیم یکی از این

اکشن‌های زیر را انجام دهیم. در هر state بعد از تصمیم گیری، با توجه به تغییرات قیمت سهام در آن روز، سرمایه state ما به روز می‌شود و ما روز بعد در state جدید دوباره به تصمیم گیری خواهیم پرداخت.

$$A = \{a_0, a_B, a_C, a_D, a_{BC}, a_{BD}, a_{CD}\}$$

در این جا a_X به معنی خرید یک سهم از شرکت X است و a_{XY} به معنی خرید دو سهم یکی از شرکت X و دیگری شرکت Y است. همچنین a_0 به معنی خرید نکردن سهام در آن روز است. با توجه به هدف رسیدن سریع به ۱۰۰ دلار در صورت نخریدن سهم در یک روز ۱۰۰۰۰۰۰ واحد جزا دریافت می‌کنیم. پاداش‌ها برای حالت‌هایی که تصمیم ما باعث رسیدن ما به ترمینال نمی‌شود، در ادامه آمده است. اگر در یک state عملی انجام دهیم و به state با سرمایه‌ی بیشتر برویم، به اندازه‌ی حاصل ضرب ۵۰۰۰۰ در تعداد واحدی که سرمایه ما زیاد شده است، پاداش دریافت می‌کنیم. هر واحد افزایش سرمایه به اندازه‌ی ۵ دلار ارزش دارد. اگر در state اکشنی انجام دهیم و سرمایه‌ی ما تغییر نکند، به اندازه‌ی ۵۰۰۰۰ برابر تعداد سهم‌های خریداری شده، جزا دریافت می‌کنیم. اگر در تصمیم گیری در یک state ضرر کنیم، به اندازه‌ی حاصل ضرب ۵۰۰۰۰ در تعداد واحدی که سرمایه ما کم شده است، جزا دریافت می‌کنیم. در این جا اگر در یک state ای باشیم که توانایی انجام اکشن انتخابی را نداشته باشیم، بی نهایت واحد جزا دریافت می‌کنیم. دلیل این موضوع این است که ما در هر state اکشنی که با سرمایه‌ی ما ممکن است را باید انجام دهیم. در خرید سهام قیمت سهم‌های مورد نظر برای خرید در ابتدای روز باید از سرمایه اولیه ما در آن زمان کمتر یا مساوی باشد وگرنه با مجازاتی که گفته شد مواجه می‌شویم. پاداش‌های تعیین شده با توجه به utility کاربر در مواجه با شرایط تعیین شده است. در این جا فرض کردیم که agent توانایی مشاهده‌ی قیمت سهم‌های شرکت‌ها را دارد و agent در state از میان اعمال متناسب با سرمایه‌ی خود، یکی را انتخاب می‌کند. به همین دلیل جزای بی‌نهایت ممکن نیست. هزینه‌ها و پاداش‌هایی که در این جا گفته شد برای بیان نحوه‌ی مدل سازی است و عامل از آن‌ها با خبر نیست. همچنین قیمت سهام شرکت‌ها به صورت احتمالاتی تغییر می‌کنند که این قیمت‌ها تعیین کننده‌ی پاداش دریافتی و state عامل در هر زمان هستند. به همین دلیل state عامل و پاداش دریافتی به صورت احتمالاتی تعیین می‌شوند. یعنی وقوع transition ها و پاداش هر transition از یک توزیع احتمال می‌آید که عامل از آن اطلاع ندارد. هدف ما رسیدن به state ترمینال هدف یا سرمایه‌ی ۱۰۰ دلار با کمترین هزینه است. در این جا این فرض را داریم که در انتهای روز درآمد سهام یا سهم‌های خریداری شده را داریم که با توجه به آن سرمایه اولیه‌ی ما یا state ما برای فردا به روز می‌شود. بعد از گذشت روز، سهام به پول تبدیل می‌شوند و تغییرات آتی قیمت سهام

خریداری شده در روزهای بعد در سرمایه‌ی ما تاثیری ندارد. با قرار گیری با تعداد زیاد در یک state، عامل یاد می‌گیرد که با توجه به سرمایه اولیه متناظر با state، بهترین عمل چه می‌تواند باشد. این عمل بهینه با توجه به تغییرات قیمت سهام و میزان سرمایه انتخاب شده است. در جدول زیر قیمت سهام شرکت‌ها در اولین روز بر اساس شماره دانشجویی من آمده است.

شرکت	قیمت اولیه‌ی سهم
B	15
C	15
D	10

الف) در این جا ما به دنبال یک سیاست greedy بهینه هستیم. نکته این است که با این نوع سیاست نمی‌توان در دنیا زندگی کرد. ما در این جا باید با زندگی کردن، به یادگیری بپردازیم. به همین دلیل برای حل این موضوع از یک الگوریتم off-policy برای یادگیری استفاده می‌کنیم. با یک سیاست epsilon greedy در دنیا زندگی می‌کنیم و سیاست greedy اولیه خود را به تدریج با policy iteration ارتقا می‌دهیم. برای حل سوال و برای داشتن عملکرد مناسب بر اساس تعداد episode های دیده شده، در این جا از الگوریتم qlearning استفاده کردیم که یک الگوریتم GPI است. دلیل استفاده از این الگوریتم، توانایی به روز کردن مقدار q در هر state بدون نیاز به دیدن کل episode و امکان شروع از هر state است. این موارد باعث می‌شوند با تعداد کمتر episode و در زمان کمتری بتوانیم به جواب برسیم. با توجه به تعداد زیاد state در این مسئله، این الگوریتم انتخاب شده است. در این مسئله هدف پیدا کردن سیاستی است که ما را سریع تر به ۱۰۰ دلار برساند و هم چنین با توجه به این که باید با محیط تعامل کنیم، باید هزینه‌ها را برای رسیدن به سیاست بهینه کمینه کنیم. پس باید هر چه سریع تر بتوانیم در الگوریتم به سیاست بهینه برسیم. ساختار کلی الگوریتم عامل برای یادگیری به شکل زیر است. در این الگوریتم از هر state می‌توانیم شروع کنیم و طبق سیاست egreedy که شرط پوشش را بر آورده می‌کند اکشن انتخاب کنیم و با محیط تعامل داشته باشیم. در این الگوریتم یک پارامتر گام برای تعیین میزان تاثیر داده جدید در یادگیری داریم که lr می نامیم. در ابتدای یادگیری ما باور اولیه‌ای از ارزش عمل در state یا $q(s,a)$ نداریم. پس در ابتدا lr باید زیاد باشد و در طول یادگیری با افزایش دانش نسبت به سیاست بهینه، به تدریج باید آن را کم کنیم تا به سیاست بهینه میل کنیم. که این کار به صورت $lr_t = lr_{t-1} \times 0.9$ انجام

شده است که lr در ابتدا یک است. پارامتر دیگر در این الگوریتم، epsilon در سیاست رفتاری است. مقدار بالای این پارامتر یعنی یک به معنی انتخاب به صورت رندم است. این رفتار رندم در ابتدای یادگیری نیاز است که دلیل آن نداشتن اطلاعات کافی در ابتدا است.

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
    $S \leftarrow S'$
  until S is terminal

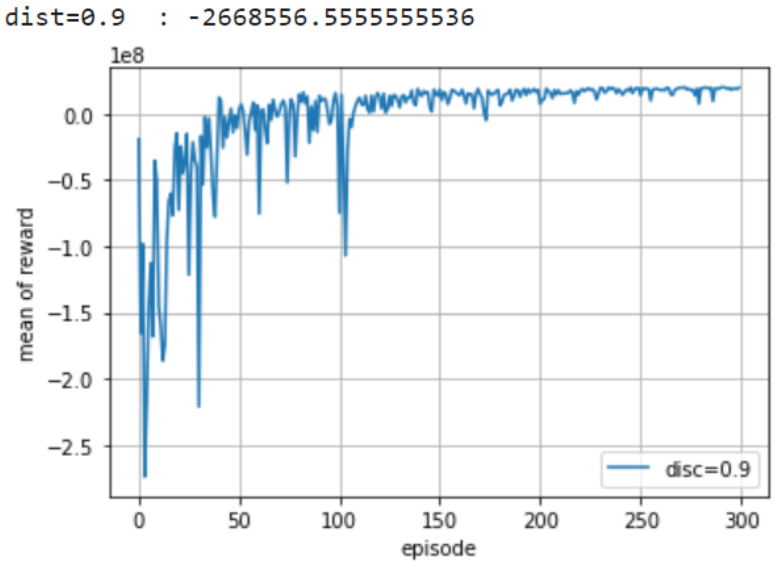
در طول زمان و با بدست آوردن اطلاعات بیشتر و نزدیک تر شدن به سیاست بهینه باید کمتر رندم عمل کنیم. این کار باعث می‌شود که طی زمان رسیدن به سیاست بهینه مجموع هزینه‌های ما کمینه بشود. در همین راستا به صورت $eps_t = eps_{t-1} \times 0.9$ پارامتر را تغییر می‌دهیم. در این الگوریتم یک سیاست رفتاری به شکل egreedy داریم. این سیاست باید پوشش دهنده‌ی سیاست بهینه greedy باشد. در سیاست بهینه در هر state احتمال انتخاب اکشن با $q(s,a)$ بیشینه برابر یک است. به همین دلیل در سیاست رفتاری در هر state احتمال انتخاب اکشن با $q(s,a)$ بیشینه را $1 - eps$ بیشتر از دیگر اکشن‌ها قرار می‌دهیم و بعد از بررسی هر episode سیاست رفتاری را بر حسب $q(s,a)$ ها به روز می‌کنیم. شرط اتمام الگوریتم زمانی است که بیشینه اختلاف q value ها بعد از یک episode با episode قبل کمتر از θ نزدیک به صفر باشد. در این مسئله با توجه به θ مورد نظر تعداد episode ها را برابر ۱۰۰ قرار دادیم. بعد از هر episode محیط به حالت قبل باز می‌گردد. در این الگوریتم نیازی نیست که در یک episode حتما به state نهایی برسیم، به همین دلیل برای کاهش زمان یادگیری، تعداد بیشینه گام در هر episode را برابر ۵۰۰۰۰۰ قرار دادیم. در برخی موارد ممکن است در یک episode ترتیب اکشن‌های انتخابی در state ها به گونه باشد که تعداد گام زیادی را انجام دهیم ولی state و اکشن‌های جدیدی را نبینیم. در این حالت ادامه‌ی یادگیری در episode بی تاثیر است و زمان را هدر می‌دهد. با توجه به این موضوع، سه بار از state هایی با سرمایه ۲۰، ۵۰، ۸۰ مانند بالا شروع به یادگیری می‌کنیم. ابتدا با ۲۰ دلار شروع می‌کنیم و ۱۰۰ episode را می‌بینیم سپس epsilon را دوباره یک می کنیم و این بار از ۵۰ دلار شروع می‌کنیم. برای

یادگیری با سرمایه‌ی جدید نباید ارزش‌های q گذشته را دور بریزیم. به همین دلیل lr را برای سرمایه‌ی جدید از 0.3 با نرخ 0.9 کم می‌کنیم. همین روش را برای سرمایه‌های بعدی انجام می‌دهیم تا q های نهایی بدست آیند. این کار باعث می‌شود که state و اکشن‌های بیشتری را ببینیم و q ها واقعی‌تر شوند. در زیر سیاست بهینه برای الگوریتم با $discount = 0.9$ آمده است.

```
[0, 15, 10, 10] [1, 0, 0, 0, 0, 0, 0]
[1, 15, 10, 10] [1, 0, 0, 0, 0, 0, 0]
[2, 15, 10, 10] [0, 0, 1, 0, 0, 0, 0]
[3, 15, 10, 10] [0, 1, 0, 0, 0, 0, 0]
[4, 15, 10, 10] [0, 1, 0, 0, 0, 0, 0]
[14, 20, 10, 10] [0, 0, 0, 0, 1, 0, 0]
[15, 20, 10, 10] [0, 0, 0, 0, 1, 0, 0]
[16, 20, 10, 10] [0, 0, 0, 0, 1, 0, 0]
[17, 20, 10, 10] [0, 0, 0, 0, 1, 0, 0]
[18, 20, 10, 10] [0, 0, 0, 0, 0, 1, 0]
[19, 20, 10, 10] [0, 0, 0, 0, 1, 0, 0]
```

تعداد state ها در مسیله زیاد است. تعدادی از این stateها در بالا آمده است. همان طور که دیده می‌شود، عامل یادگرفته است که در state های صفر و ۵ دلار بهترین کار صبر کردن است. در ۵ دلار برای کاهش قیمت سهام صبر می‌کنیم چرا که خرید بی‌جا می‌تواند باعث شود سرمایه‌ی ما به مقداری برسد که دیگر نتوانیم به خرید و فروش سهام ادامه دهیم. به معنایی عامل در این state ها مسیرهای سرمایه گذاری تا ترمینال را برای تصمیم گیری مد نظر قرار داده است. همچنین در چند state از ۱۵ تا ۲۰ دلار عامل سهم B را انتخاب کرده است که با توجه به احتمال بیشتر افزایش قیمت آن، انتخابی منطقی است. در ۱۰ دلار نیز عامل C را انتخاب کرده که با توجه به سرمایه در لحظه و قیمت سهم شرکت‌ها است. سهم C احتمال ضرر کمتری از D دارد و به همین دلیل انتخاب شده است. باز در این جا پاداش تا انتها در نظر گرفته شده است. از ۷۰ دلار تا ۹۵ دلار نیز دو سهم برای خرید در نظر گرفته شده است که با توجه به سرمایه‌ی اولیه بیشتر ما است. در همه به جز یک مورد BC انتخاب شده است که یعنی سهم B و C خریداری شود. این به این خاطر است که احتمال افزایش سرمایه در آن بیشتر است و احتمال بالایی در آن است که ۱۰ دلار پولمان زیاد شود و دو گام به هدف نزدیک‌تر شود. احتمال افزایش یرمایه در آن نزدیک به B است. در یک مورد نیز BD انتخاب شده است که احتمال افزایش درآمد در آن نزدیک به سهم D است و بیشترین احتمال در آن برای ثابت ماندن سرمایه است. این انتخاب ازB بدتر است ولی از بقیه موترد بهتر می‌باشد. همچنین در آن احتمالی برای دو گام نزدیک شدن به هدف وجود دارد. در این موارد نیز مسیر تا هدف برای انتخاب مسیر بهینه در نظر گرفته شده است.

نمودار متوسط پاداش به ازای هر episode و مقدار متوسط پاداش به ازای یک episode در زیر آمده است. همان طور که دیده می‌شود به طور کلی میزان متوسط پاداش به صورت صعودی افزایش پیدا کرده است و در انتها در یک مقدار ثابت شده است. این ثابت شدن نشان دهنده‌ی رسیدن الگوریتم به سیاست بهینه است.



در نمودار بالا در episode هایی دیده می‌شود که به طور ناگهانی پاداش کم شده است و بعد از آن دوباره با سرعت به مقداری بالاتر از مقدار اولیه رسیده است. این موارد نشان دهنده‌ی تغییر state شروع و ادامه‌ی یادگیری از آن است. برای رسم نمودار سه بار یادگیری انجام شده و مقدار پاداش در هر episode متوسط گرفته شده است.

ب) در ادامه عملکرد با توجه به مقادیر discount مختلف آمده است. در ابتدا مقدار discount آمده و بعد از آن، عملکرد عامل در stateهایی که در قسمت قبل بررسی کردیم آمده است.

```
0
[0, 15, 10, 10] [1, 0, 0, 0, 0, 0, 0]
[1, 15, 10, 10] [1, 0, 0, 0, 0, 0, 0]
[2, 15, 10, 10] [0, 0, 0, 1, 0, 0, 0]
[3, 15, 10, 10] [0, 0, 0, 1, 0, 0, 0]
[4, 15, 10, 10] [0, 1, 0, 0, 0, 0, 0]
[14, 20, 10, 10] [0, 1, 0, 0, 0, 0, 0]
[15, 20, 10, 10] [0, 0, 0, 0, 1, 0, 0]
[16, 20, 10, 10] [0, 1, 0, 0, 0, 0, 0]
[17, 20, 10, 10] [0, 0, 0, 1, 0, 0, 0]
[18, 20, 10, 10] [0, 1, 0, 0, 0, 0, 0]
[19, 20, 10, 10] [0, 0, 0, 1, 0, 0, 0]
```

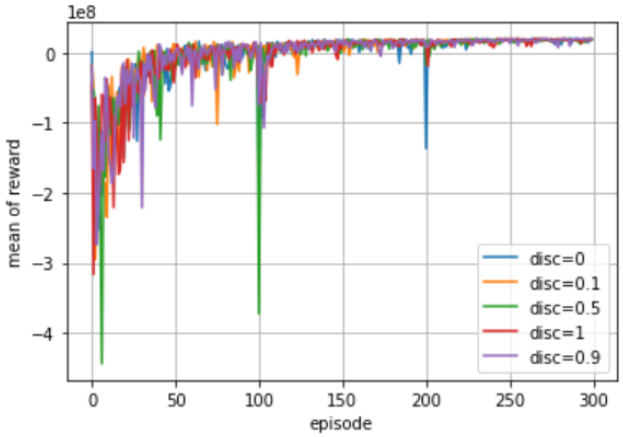
```
0.1
[0, 15, 10, 10] [1, 0, 0, 0, 0, 0, 0]
[1, 15, 10, 10] [1, 0, 0, 0, 0, 0, 0]
[2, 15, 10, 10] [0, 0, 0, 1, 0, 0, 0]
[3, 15, 10, 10] [0, 0, 0, 1, 0, 0, 0]
[4, 15, 10, 10] [0, 0, 1, 0, 0, 0, 0]
[14, 20, 10, 10] [0, 1, 0, 0, 0, 0, 0]
[15, 20, 10, 10] [0, 1, 0, 0, 0, 0, 0]
[16, 20, 10, 10] [0, 0, 0, 0, 1, 0, 0]
[17, 20, 10, 10] [0, 0, 0, 0, 1, 0, 0]
[18, 20, 10, 10] [0, 0, 0, 0, 0, 0, 1]
[19, 20, 10, 10] [0, 0, 0, 0, 0, 0, 1]
```

```
0.5
[0, 15, 10, 10] [1, 0, 0, 0, 0, 0, 0]
[1, 15, 10, 10] [1, 0, 0, 0, 0, 0, 0]
[2, 15, 10, 10] [0, 0, 1, 0, 0, 0, 0]
[3, 15, 10, 10] [0, 0, 0, 1, 0, 0, 0]
[4, 15, 10, 10] [0, 0, 0, 1, 0, 0, 0]
[14, 20, 10, 10] [0, 0, 0, 0, 0, 1, 0]
[15, 20, 10, 10] [0, 0, 0, 0, 1, 0, 0]
[16, 20, 10, 10] [0, 1, 0, 0, 0, 0, 0]
[17, 20, 10, 10] [0, 0, 1, 0, 0, 0, 0]
[18, 20, 10, 10] [0, 0, 0, 0, 0, 0, 1]
[19, 20, 10, 10] [0, 0, 0, 0, 1, 0, 0]
```

```
1
[0, 15, 10, 10] [1, 0, 0, 0, 0, 0, 0]
[1, 15, 10, 10] [1, 0, 0, 0, 0, 0, 0]
[2, 15, 10, 10] [0, 0, 1, 0, 0, 0, 0]
[3, 15, 10, 10] [0, 1, 0, 0, 0, 0, 0]
[4, 15, 10, 10] [0, 1, 0, 0, 0, 0, 0]
[14, 20, 10, 10] [0, 0, 0, 0, 1, 0, 0]
[15, 20, 10, 10] [0, 0, 0, 0, 1, 0, 0]
[16, 20, 10, 10] [0, 0, 0, 0, 1, 0, 0]
[17, 20, 10, 10] [0, 0, 0, 0, 1, 0, 0]
[18, 20, 10, 10] [0, 0, 0, 0, 0, 1, 0]
[19, 20, 10, 10] [0, 0, 0, 0, 1, 0, 0]
```

همان طور که دیده می‌شود، برای discount نزدیک به یک، عامل تمام پاداش‌های مسیر تا ترمینال هدف را برای تصمیم گیری انتخاب می‌کند. در discount نزدیک به صفر، عامل فقط پاداش هر اکشن در هر state را در نظر می گیرد. پاداش instant در این جا تصمیم را مشخص می‌کند. به همین دلیل می‌بینیم که ۹۵ دلار در discount=0 سهم D انتخاب شده که با احتمال زیادی ضرر می‌دهد و زمان رسیدن ما به هدف را بیشتر می‌کند. در ۸۰ دلار نیز دیده می‌شود که B انتخاب شده و احتمال گرفتن ۱۰ دلار در نظر گرفته نشده است. احتمال گرفتن ۱۰ دلار با بررسی مسیر تا انتها بدست می‌آید چرا که کوتاه شدن مسیر باعث می‌شود مجموع پاداش دریافتی تا انتها بیشتر شود. در discount=0.1 نیز در ۷۰ دلار B انتخاب شده است که باز احتمال گرفتن ۱۰ دلار در نظر گرفته نشده است. دلیل آن نزدیک بینی است. مقدار discount پایین است به همین دلیل برای پاداش‌های با فاصله‌ی دور از ۷۰ دلار تا ۱۰۰ دلار، ما پاداش نزدیک به صفر در نظر می‌گیریم. پس مسیر تا انتها در تصمیم گیری در نظر گرفته نمی‌شود. در discount های بالا، ۰.۹ و ۱ ، تمام مسیر تا انتها برای تصمیم مورد نظر است و انتخاب‌ها منطقی تر هستند و پاداش‌های delayed در تصمیم گیری موثر هستند. در discount=0.5 نیز در ۸۰ دلار انتخاب بهینه نیست. دلیل آن این است که تمام مسیر تا انتها در نظر گرفته نمی‌شود. مقدار discount از ۰.۱ بیشتر است و ما طول بیشتری از مسیرها را برای تصمیم در نظر می‌گیریم به همین دلیل در موارد بیشتری تصمیم ما منطقی است. در ادامه نمودار متوسط پاداش به ازای هر episode مانند قسمت قبل برای حالت‌های مختلف discount آمده است. همان طور که دیده می‌شود، نمودارها به مقدار خوبی بر هم منطبق هستند.

```
dist=0   : -3830001.0
dist=0.1   : -1689612.1111111117
dist=0.5   : -3902945.4444444464
dist=1   : -3104278.7777777785
```



در episode های اول برای شروع از هر state، متوسط پاداش الگوریتم با discount نزدیک به یک، به مقدار نهایی نزدیک تر است. در حالی که برای مقادیر discount نزدیک به صفر، فاصله تا متوسط نهایی بیشتر است که این مورد نشان دهنده‌ی انتخاب‌های غیر بهینه تر در state ها است. این اختلاف تا حدی در مسیر رسیدن به مقدار نهایی نیز وجود دارد. به همین دلیل است که متوسط کل پاداش‌های مسیر یادگیری در discount=1 از discount=0 کمتر است. در این جا باید توجه شود که پاداش‌ها منفی هستند. در discount های پایین تر در مسیر یادگیری جهش‌های بیشتری را مشاهده می‌کنیم که می‌تواند به دلیل در نظر نگرفتن پاداش‌های کل در episode باشد. فاصله‌ی مقدار نهایی متوسط پاداش‌ها برای مقادیر مختلف discount ناچیز است و همچنین نمودارها یک روند دارند. این وجود نزدیکی و یک روندی نشان می‌دهد که discount تاثیر کمی در بهبود الگوریتم دارد.

سوال ۲ – کارخانه تولید مواد غذایی

در این جا فرض می‌کنیم که قیمت هر واحد مواد اولیه و مواد غذایی ثابت است. هم چنین این فرض را داریم که کارخانه تعطیلی ندارد و در تمام ساعات کار می‌کند. همچنین تولید هر ماده‌ی غذایی یک فرمول مشخص بر حسب مواد اولیه دارد که در زمان تغییر نمی‌کند. در این سوال به دنبال یک مدل MDP هستیم. پس باید مجموعه state ها و مجموعه اکشن‌ها را تعریف کنیم. در این جا هر state به صورت زیر تعریف می‌شود.

$$state: \{\text{cash}, T, Raw\ material, foods, orders, demands\}$$

در این جا cash سرمایه و پول نقد کارخانه است که در حساب موجود است. هم چنین این فرض را داریم که برای خرید مواد اولیه، ما فقط با پرداخت پول از حساب کارخانه می‌توانیم خرید انجام دهیم. در این جا با توجه به قیمت هر واحد مواد غذایی و اولیه، ما تعداد محدودی حالت برای cash داریم. Raw material شامل حجم موجود از هر ماده‌ی اولیه و سن مانده تا زمان انقضا هر ماده‌ی اولیه است. مثلا اگر ما یک ماده‌ی اولیه به اسم آرد داشته باشیم، ما در state یک سه تایی مثل $(Flour, 100kg, 3\ month)$ داریم. این سه تایی نشان می‌دهد که ما در کارخانه ۱۰۰ کیلوگرم آرد داریم که ۳ ماه دیگر منقضی می‌شود و باید دور ریخته شود. همچنین اگر ما یک ماه بعد ۱۰۰ کیلوگرم آرد جدید بخریم که ۳ ماه ماندگاری داشته باشد، یک سه‌تایی دیگر علاوه بر قبلی به صورت $(Flour, 100kg, 3\ month)$ خواهیم داشت. این نگاه باعث می‌شود که بتوانیم برای مواد در حال انقضا تصمیم گیری کنیم. همچنین این کار باعث می‌شود که میزان خرید مواد اولیه را طوری تنظیم کنیم که اسراف نشود. مثلا اگر ۲ کیلو ماده‌ی اولیه داشته باشیم که در حال انقضا است، آن را در صورت امکان به غذا تبدیل کنیم تا بتوانیم از آن درآمد کسب کنیم. foods شامل تعداد موجود از ماده‌ی غذایی و سن مانده‌ی آن‌ها تا انقضا است. به ازای هر غذایی مثل A سه تایی مانند گذشته به شکل $(A, 10, 2\ day)$ وجود دارد. خواص سه تایی در این جا مانند مواد اولیه است. این نگاه باعث می‌شود که میزان تولید مواد غذایی را طوری تنظیم کنیم که اسراف نشود. orders شامل تعداد درخواست در کل برای هر ماده‌ی غذایی، تعداد درخواست با پاسخ مثبت برای هر ماده‌ی غذایی و درخواست‌های این لحظه برای هر غدا است. با توجه به عملی که در state کنونی انجام می‌شود، این مقادیر به روز می‌شوند. فرض کنید که در state کنونی ۲ در خواست برای یک غذا داشته باشیم و فقط بتوانیم یک سفارش را انجام دهیم. اگر یک واحد غذای مورد نظر را تولید کنیم، در state بعدی تعداد سفارشات

مثبت برابر یک است. تعداد سفارشات کل برابر ۲ به علاوه‌ی سفارشات این لحظه است. هر transition و تصمیم گیری در لحظه انجام می‌شود پس اطلاعاتی از دست نمی‌رود و به صورتی که گفته شد می‌توان اطلاعات را در state به روز کرد. به اندازه‌ی مواد اولیه‌مورد نیاز برای سفارش با پاسخ منفی پارامترهای demands در state بعدی به روز می‌شوند. این نوع نگه‌داری اطلاعات، میزان رضایت را به ما نشان می‌دهد و ما می‌توانیم در تصمیم‌گیری از آن استفاده کنیم. orders در عمل باوری است که ما از رضایت مشتریان و سفارشات داریم که هر لحظه به روز می‌شود. در orders و demands اطلاعات برای تمام زمان‌ها در روزهای مختلف ماه‌های سال به روز می‌شوند. در ابتدا برای توزیع سفارش و تقاضا در هر زمان یک باور اولیه داریم که با دیدن سفارشات و تقاضا هر لحظه آن‌ها را با bayesian belief revision به روز می‌کنیم. دلیل این موضوع این است که تقاضا از یک توزیع متغیر با زمان می‌آید. T در این جا نشان می‌دهد که زمان برابر ابتدای ماه است یا نه. چون هر transition در لحظه است، پس این پارامتر در state بعدی یک لحظه از مقدار آن در state قبل جلوتر است و زمانی که به اول ماه می‌رسیم را می‌توانیم مشخص کنیم. عملا یک شمارنده‌ی لحظه است، در ابتدای ماه صفر می‌شود و بعد از هر transition یکی زیاد تر می‌شود. Demands شامل حجم‌های مورد نیاز هر ماده اولیه است که برای پاسخ به سفارشات با پاسخ منفی نیاز داشته ایم. این مقادیر برای state جدید از مقادیر متناظر در state گذشته و با توجه به سفارشات حین transition به state جدید، بدست می‌آیند. در این سوال ثبت سفارش تا رسیدن سفارش به مشتری در لحظه صورت می‌گیرد و هم چنین تولید غذا از مواد اولیه در لحظه صورت می‌گیرد. همچنین در این جا این فرض را داریم که خرید در اول ماه در یک لحظه صورت می‌گیرد. با توجه به این موارد، پارامترهای هر state در حین transition از اطلاعات state قبل و با توجه به اکشن انجام شده و سفارشات در حین transition بدست می‌آیند. این بدان معنی است که اطلاعات لازم هر state از stateهای قبل مستقل است. پس در هر state فقط با توجه به پارامترهای آن می‌توان تصمیم گیری کرد. یعنی عبارت زیر برقرار است.

$$p(\acute{s}|s,a) = p(\acute{s}|s_{t-1} = s, a_{t-1} = a, s_{t-2} = \ddot{s}, a_{t-2} = \ddot{a}, \dots)$$

به همین دلیل مدل یک MDP است. در ادامه درباره‌ی اکشن‌ها و $P_{s\acute{s}}^{a}$ و $R_{s\acute{s}}^{a}$ صحبت می‌کنیم. زمان لازم برای تصمیم گیری و انجام اکشن و به روز رسانی سفارشات، زمان کمی مثل δ است. با توجه به این موضوع یک ماه تعداد محدود و قابل شمارشی از δ است که برای transition و برای گرفتن سفارشات در نظر گرفتیم. به طور کلی state مشخصی به عنوان هدف نهایی برای ما وجود ندارد و فرآیند همیشه ادامه دارد. اما می‌توان state با پارامتر های خاصی را به عنوان هدف نهایی تعیین کرد. در این جا اگر cash برابر

صفر شود و کالایی برای فروش نداشته باشیم، ورشکست شده ایم. تمام حالاتی که در آن دیگر نمی‌توانیم درآمدی بدست آوریم، چاه نام دارد. مثلا در حسابمان پولی نباشد و مواد غذایی و مواد اولیه نداشته باشیم. در state چاه همیشه می‌مانیم و همیشه جزا دریافت می‌کنیم. همچنین می‌توانیم state چاه را یک پایانه در نظر بگیریم که در صورت ورود یک جزا علاوه بر پاداش جابه جایی دریافت می‌کنیم و بعد ار آن فرآیند تمام می‌شود.

اکشن‌های مدل به صورت زیر است. در این جا a_0 به معنی تولید نکردن و خرید نکردن است. a_{bj} به معنی خرید حجم مشخص شده در bj برای هر ماده‌ی اولیه است.

$$A = \{a_{pi}, a_{bj} \,|1 \le i \le P, 1 \le j \le B\} \cup \{a_0\}$$

یک حالت برای bj می‌تواند به صورت $(1,0,0,\dots,0)$ باشد که به معنی خرید یک واحد از ماده‌ی اولیه شماره یک و خرید نکردن بقیه‌ی مواد اولیه است. زیر یک مقدار حجم برای خرید مواد اولیه معنا ندارد که ما در این جا این مقدار را برابر ۱۰۰ کیلوگرم در نظر گرفته‌ایم. یک واحد برابر این مقدار ۱۰۰ کیلوگرم است. حداکثر تعداد واحد برای یک ماده‌ی اولیه برابر فضای انبار تقسیم بر ۱۰۰ کیلوگرم است که ما آن را برابر N قرار داده‌ایم. با توجه به این موضوع مجموع المان‌های bj برابر N است. a_{pi} به معنی تولید تعداد مشخص شده در pi برای هر ماده‌ی غذایی است. یک حالت برای pi می‌تواند به صورت $(1,0,0,\dots,0)$ باشد که به معنی تولید یک واحد از ماده‌ی غذایی شماره یک و تولید نکردن بقیه‌ی مواد غذایی است. حداکثر تعداد واحد برای یک ماده‌ی غذایی برابر فضای انبار تقسیم بر حجم یک واحد معیار است که ما آن را برابر N_i قرار داده‌ایم. با توجه به این موضوع مجموع حجم المان‌های pi حداکثر برابر N یا حجم موجود انبار مواد غذایی است. در نظر گرفتن اکشن تولید باعث می‌شود که در زمانی که نیازی به ماده‌ی غذایی نداریم نیز در صورت سودمند بودن ماده‌ی غذایی تولید کنیم تا در آینده بتوانیم سفارشات بیشتری را انجام دهیم. این موضوع می‌تواند سود ما را بیشتر کند. با توجه به این توضیحات تعداد اکشن‌های ممکن در هر state نیز محدود و قابل شمارش است. در $P^a_{s\acute{s}}$ احتمال وقوع هر transition از s به $\acute{s}$ با انجام a مشخص می‌شود. در هر state با توجه به پارامترها تعدادی از اکشن‌ها قابل انجام هستند که این اکشن‌ها برای state ها از قبل به صورت $A(s_t)$ مشخص است. در هر state فقط می‌توان اکشن های ممکن را انجام داد. یعنی عبارت زیر برقرار است.

$$p(\acute{s}|s,a) = 0 \ \forall_{s,\acute{s},a} \ a \notin A(s)$$

فرض کنید که در اول ماه در s باشیم و بخواهیم یک اکشن خرید از حالت‌های ممکن را مانند a انجام دهیم. اگر خرید انجام شود پارامترها به گونه‌ای است که به s_1 می‌رویم. خرید موفق یعنی دادن پول و در یافت مواد اولیه. اگر خرید موفق نباشد ما پولمان را از دست داده ایم و مواد اولیه دریافت نکرده ایم. در این شرایط به s_2 می‌رویم. این فرض را داریم که در خرید حالت دیگری ممکن نیست. در این صورت احتمال جابه‌جایی و پاداش آن به صورت زیر است. در این جا a_i به معنای خرید کردن حجم مشخص شده‌ی کالای i در a به تنهایی است. $p(s_1|s,a_i)$ به معنی احتمال خرید موفق a_i است و $p(s_2|s,a_i)$ به معنی احتمال خرید ناموفق a_i است. برای تمام کالاهای غیر تحریمی $p(s_1|s,a_i) = 0.7$ و $p(s_2|s,a_i) = 0.3$ است. برای تمام کالاهای تحریمی $p(s_1|s,a_i) = 0.4$ و $p(s_2|s,a_i) = 0.6$ است. البته این احتمال‌ها می توانند از یک توزیع خاص نیز بیایند.

$$p(s_1|s,a) = \prod_i p(s_1|s,a_i)$$
$$p(s_2|s,a) = \prod_i p(s_2|s,a_i)$$

در این جا $r(s_1|s,a)$ برابر $r(s_2|s,a)$ و برابر هزینه خرید مواد اولیه به علاوه درآمد حاصل از فروش غذا، هزینه‌ی شفارشات پاسخ داده نشده و هزینه‌ی مواد منقضی شده در لحظه انجام جابه‌جایی است. هزینه‌ی شفارشات پاسخ داده نشده برابر قیمت آن‌ها است. هزینه‌ی مواد منقضی شده برابر قیمت آن‌ها به علاوه‌ی هزینه‌ی انبار داری آن‌ها است. هزینه‌ی انبار داری هر کالا را ما در این جا روزانه ثابت در نظر می‌گیریم. مثلا اگر در s باشیم و بخواهیم a را انجام دهیم و در این زمان دو سفارش داشته باشیم که بتوانیم با غذای موجود یا مواد اولیه به یکی پاسخ دهیم، پاداش ما برابر هزینه حاصل از جمع قیمت مواد اولیه خریداری شده، در آمد یک سفارشی است که پاسخ دادیم، هزینه‌ی مواد منقضی شده و هزینه‌ای برابر قیمت سفارشی که پاسخ ندادیم است. در هر transition یک لحظه مواد به انقضا نزدیک می‌شوند و سن جدید آنها در state مقصد در نظر گرفته می‌شود.

فرض کنید در وضعیت s باشیم و عمل تولید غذای a را بخواهیم انجام دهیم. اگر s_1 همان state بعد از انجام موفق عمل باشد داریم:

$$p(s_1|s,a) = 1$$

یعنی قطعا اکشن موفق انجام می‌شود و پارامتر های state جدید بر اساس state قبلی و اکشن انجام شده به روز می‌شوند. در این جا $r(s_1|s,a)$ برابر هزینه‌ی ناشی از استحلاک کارخانه به علاوه‌ی درآمد حاصل از فروش در آن لحظه، هزینه‌ی سفارشات پاسخ داده نشده و هزینه‌ی مواد منقضی شده در این جابه‌جایی است. هزینه‌ی استحلاک کارخانه هر سال به روز می‌شود و در طول سال برای هر لحظه ثابت است. پاداش‌ها و احتمالات بیان شده می‌توانند با موارد گفته شده متفاوت باشند ولی المان‌های بیان شده برای هر کدام، مهمترین مواردی بودند که می‌توان در نظر گرفت.

اگر در s باشیم و با انجام a وارد چاه شویم علاوه بر پاداش و جزای اکشن، به اندازه‌ی قیمت کارخانه جزای بیشتر دریافت می‌کنیم. بعد از این پاداش و جزا، فرآیند تمام می‌شود. اگر چاه را ترمینال در نظر نگریم همیشه در آن می‌مانیم و هر لحظه به اندازه‌ی سفارشاتی که پاسخ نمی‌دهیم جزا دریافت می‌کنیم.