ارائه‌ی روش تسریع شده‌ی مبتنی بر یادگیری تقویتی توزیع شده، در سیستمهای چند عامله

سارا اسفندیاری[۱]، بهروز معصومی[۲] و محمدرضا میبدی[۳]

چکیده

در این مقاله، الگوریتم جدیدی مبتنی بر یادگیری تقویتی توزیع شده، برای افزایش سرعت یادگیری، در الگوریتم‌های یادگیری تقویتی، ارائه شده است. در این روش، برای هر حالت از محیط، یک عامل مجازی وجود دارد که مسئول یادگیری در آن حالت است. عامل مجازی، برای یادگیری از الگوریتم Q-Learning استفاده می‌کند. در روش پیشنهادی، یک تابع بهبود یافته‌ی جدید برای انتخاب عمل در هر حالت، پیشنهاد شده است که موجب تسریع در الگوریتم‌های مبتنی بر Q-Learning می‌شود. برتری تابع پیشنهادی با انجام آزمایشات متعدد و تحلیل‌های ریاضی به اثبات رسیده است. از الگوریتم مذکور، برای حل بازی‌های مارکوف استفاده شده است. نتایج بدست آمده در آزمایش‌ها، نشان می‌دهد که الگوریتم پیشنهادی از لحاظ تعداد حرکات لازم برای رسیدن به پاسخ بهینه و میانگین پاداش‌های بدست آمده، دارای کارایی بسیار بالایی نسبت به روش‌های قبلی است.

کلمات کلیدی

الگوریتم Q-Learning ، بازی مارکوف، سیستم‌های چندعامله، توزیع بولتزمن، یادگیری تقویتی، یادگیری ماشین.

Accelerated Method Based on Distributed Reinforcement Learning in Multi-Agent Systems

Sara Esfandiari ; Behrooz Masoumi ; Mohammad reza Meybodi

ABSTRACT

In this paper, has been proposed a new method using distributed Reinforcement Learning to increase rate of learning in Reinforcement Learning Algorithms. In this recommended method, for every state of the environment, a virtual agent that is responsible for learning in this state. Virtual agents use Q-Learning Algorithms for learning best policy. In this method, used a new optimized function has been proposed to select the action, which has led to an increase in algorithms based on Q-Learning. The superiority of proposed function, with variety experimental and mathematical analysis has been proved. The mentioned algorithm, has been used for solving the problem of cooperative Markov Games. The results gained during the experiments show that in comparison with the previous methods, the recommended algorithm has a very high efficiency from the perspective of the number of movements needed for achieving the optimized answer and the average of rewards gained.

KEYWORDS

Q-Learning Algorithms, Markov Games, Multi agent Systems, Boltzmann function, Reinforcement Learning, Machine Learning

[۱] دانشجوی کارشناسی ارشد دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه آزاد اسلامی واحد قزوین، Sara.esfandiari@gmail.com
[۲] عضو هیات علمی دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه آزاد اسلامی واحد قزوین، Masoumi@Qiau.ac.ir
[۳] عضو هیات علمی دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیر کبیر، تهران ،MMeybodi@aut.ac.ir

۱. مقدمه

یک سیستم چندعامله[۱]، شامل مجموعه‌ای از عامل‌های هوشمند و خودمختار است که هر یک به طور جداگانه با محیطی اشتراکی در ارتباط هستند، تا بتوانند به هدف مشخصی برسند[۱]. عامل‌ها، موجودیت‌های محاسباتی هستند که به وسیله سنسورهایشان قادر به مشاهده‌ی محیط پیرامون خود هستند. این عامل‌ها بایستی قادر باشند که در هر لحظه عمل مناسبی را بر اساس مشاهدات خود انجام دهند. در حال حاضر از سیستم‌های چندعامله در زمینه‌ی مدیریت منابع، سیستم‌های کنترل توزیع شده، سیستم‌های نظامی و مخابراتی و تیم‌های روباتیکی و داده‌کاوی، استفاده فراوانی می‌شود[۲]. با توجه به اینکه عامل‌ها در سیستم‌های چندعامله، با مسئله‌ی کمبود یا فقدان اطلاعات درباره‌ی محیط مواجه هستند و شناخت کاملی از محیط وجود ندارد، بنابراین بدون وجود روشی برای ایجاد هماهنگی میان عامل‌ها، ممکن است سیستم دچار هرج و مرج شده و از رسیدن به هدف نهایی بازبماند[۳]. تاکنون روش‌های مختلفی برای حل مسئله‌ی هماهنگی عامل‌ها در سیستم‌های چندعامله، به وجود آمده است. در این میان، الگوریتم‌های یادگیری تقویتی[۲] به عنوان ابزاری برای دستیابی به رفتار هماهنگ، به خاطر سادگی و استحکامی[۳] که دارند، مورد توجه قرار گرفته است. . الگوریتم‌های یادگیری تقویتی در هر مرحله‌ی زمانی، به عامل اجازه می‌دهند که بر اساس مشاهدات خود از محیط، یک عمل را انجام دهد و به حالت جدیدی وارد شود، سپس یک سیگنال پاداش که نشان‌دهنده‌ی کیفیت عمل انتخاب شده است، از محیط دریافت کند. روشهای مبتنی بر یادگیری تقویتی به دو دسته تقسیم-بندی می‌شوند: دسته‌ی اول آنهایی هستند که هیچ‌گونه فرضی در مورد ارتباط بین عامل‌ها در نظر نمی‌گیرند و به نام یادگیرنده‌های مستقل[۴] نامیده می‌شوند و دسته‌ی دوم، آنهایی هستند که فرض می‌کنند، عامل‌ها می‌توانند اَعمال عامل‌های دیگر را مشاهده نموده و از این مشاهدات برای بهبود هماهنگی استفاده می‌کنند، این روش را روش یادگیری با اعمال مشترک[۵] می‌نامند[۴] .

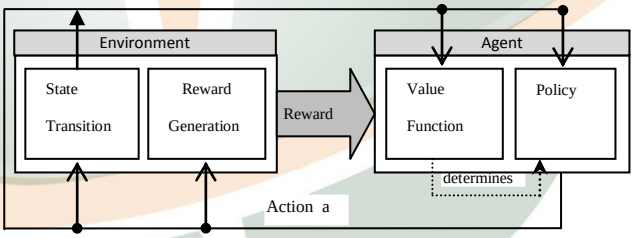
تاکنون برای مدل‌سازی سیستم‌های چندعامله، مدلهای مختلفی پیشنهاد شده است؛ مدل بازی‌های مارکوف[۶] (MG) ، یکی از این مدل‌هاست[۵, ۶]. این مدل، توسعه‌ای از فرآیندهای تصادفی مارکوف[۷](MDP) ، به حالت چندعامله است. بازی‌های مارکوف بر اساس پاداش عامل‌ها به دو دسته‌ی همکارانه[۸] و غیرهمکارانه[۹] تقسیم‌بندی می‌شوند. در بازی‌های همکارانه، عامل‌ها هدف مشترکی را دنبال می‌کنند و نیز می‌توانند از اَعمال عامل‌های دیگر مطلع شوند و تصمیم‌گیری‌های خود را بر اساس مشاهدات خود از محیط و اعمال عامل‌های دیگر اتخاذ نمایند[۷]. اگر همه‌ی عامل‌ها پاداش یکسانی را دریافت کنند، آنها را بازی‌های کاملا همکارانه(MMDP) می‌نامند[۸]. در بازی‌های غیرهمکارانه، عامل‌ها هدف مشترکی ندارند و هر عامل سعی می-کند، هدف خود را ماکزیمم نماید. این بازی‌ها به دو گروه بازی‌های با مجموع صفر[۱۰](رقابتی) و بازی‌های با مجموع کلی[۱۱](غیررقابتی) تقسیم‌بندی می-شوند. در بازی‌های با مجموع صفر، دو عامل در بازی وجود دارد که پاداش اولی قرینه‌ی پاداش دومی است. در مقابل، در بازی‌های مجموع کلی(غیررقابتی)، به عامل‌ها اجازه داده می‌شود که هر پاداش دلخواهی را دریافت کنند. در بازی‌های کاملا همکارانه، با توجه به اینکه همه‌ی عامل‌ها پاداش یکسانی دریافت می‌کنند، عامل‌ها بایستی یاد بگیرند، تا در مورد سیاست بهینه، توافق نمایند. در مقابل، در بازی‌های مارکوفی مجموع کلی، به دلیل وجود پاداش‌های متفاوت، پیدا کردن راه‌حل بهینه مشکل بوده و لذا نقاط تعادل بازی[۱۲](سیاست تعادل نش[۱۳]) مورد جستجو قرار می‌گیرد. تعادل نش، وضعیتی است که هیچ عاملی به تنهایی نمی‌تواند برای بهبود پاداشش، سیاستش را تغییر دهد تا زمانی‌که تمام عامل‌های دیگر سیاستشان را ثابت نگه‌دارند. برای پیدا کرن راه‌حل بهینه در بازی‌های مارکوفی، اعم از رقابتی و غیررقابتی، الگوریتم‌های یادگیری تقویتی مختلفی، به کار رفته‌اند. در سال ۱۹۹۴، الگوریتم minimax برای بازی‌های رقابتی پیشنهاد شد[۹]. در سال ۲۰۰۳، این الگوریتم برای بازی‌های با مجموع کلی، بازنگری شدو آن را Nash-Q نامیدند که تحت شرایط خاص به سیاست تعادل نش همگرا می‌شود[۵]. در سال ۲۰۰۰، الگوریتم Distributed Q-Learning برای عامل-های مستقل معرفی شد، ثابت شده است که این الگوریتم به سمت تعادل نش همگرا می‌شود، با اینحال در این الگوریتم از هیچ‌گونه مکانیزم هماهنگی استفاده نشده است[۱۱]. در سال ۲۰۰۴ ، الگوریتمی به نام FMQ معرفی شده است که مقادیر Q هر عمل در استراتژی بولتزمن را توسط تابع جدیدی، تغییر می‌دهد و بدین ترتیب باعث همگرایی زودتر به سمت سیاست بهینه می‌شود[۱۲]. در سال ۲۰۰۷، الگوریتمی به نام Hysteretic Q-Learning ، معرفی شده است که با اضافه کردن پارامتر جدیدی به روش FMQ ، باعث بهبود عملکرد الگوریتم شده است[۱۵].

در الگوریتم‌های یادگیری تقویتی برای حل مسئله‌ی همکاری در سیستم‌های چندعامله، برای هر یک از عامل‌ها، جدولی موسوم به Q-table در نظر گرفته می‌شود. با افزایش فضای حالات محیط و اَعمال عامل‌ها، کارکرد این الگوریتم‌ها بدلیل بزرگی حجم اطلاعات جداول Q-table با مشکل مواجه می‌شود. در این مقاله، برای حل این مشکل، از الگوریتم جدید توزیع شده، که در آن، برای هر یک از عامل‌ها در هر حالت از محیط، یک جدول در نظر گرفته شده است، استفاده شده است. یکی از راه‌های افزایش سرعت الگوریتم‌های یادگیری تقویتی، بهبود تابع انتخاب عمل است. در [۱۲, ۳] نمونه-هایی از این توابع ارائه شده‌اند. این توابع برای انتخاب عمل در هر حالت به کار برده شده‌اند. در این مقاله، به منظور افزایش سرعت یادگیری سیاست بهینه برای بازی‌های مارکوف در حالت عامل‌های مستقل، الگوریتم جدیدی مبتنی بر یادگیری تقویتی توزیع شده به نام Dist-BSAQL پیشنهاد می-شود؛ که در آن، از یک تابع بهبودیافته‌ی جدید، برای انتخاب عمل در هر حالت استفاده می‌شود. نتایج شبیه‌سازی‌های انجام شده بر روی محیط Grid World نشان می‌دهد که الگوریتم پیشنهادی، نسبت به روش‌های موجود، کارایی بهتری از نظر سرعت دارد. در ادامه‌ی مقاله، ابتدا در بخش ۲ مفاهیم اولیه، و در بخش ۳ الگوریتم پیشنهادی شرح داده می‌شود و بخش ۵ به بررسی رفتار الگوریتم پیشنهادی و تحلیل آن، اختصاص یافته است و بخش ۶ نتیجه‌گیری است.

۲. مفاهیم اولیه

۲-۱- یادگیری تقویتی

یک عامل یادگیر تقویتی، رفتارش را از طریق تعامل با یک محیط ناشناخته و مشاهده‌ی نتایج اعمالش، تعیین می‌کند[۱۶]. ایده‌ی یادگیری تقویتی در شکل ۱ آمده است. در شکل ۱، ابتدا عامل، حالت فعلی سیستم (S) را دریافت می‌کند. با استفاده از یک تابع تصمیم‌گیرنده(Policy) عمل a مشخص می‌شود و عامل پس از اِعمال عمل a بر روی محیط، پاداش r را دریافت می‌کند. سپس با استفاده از مقادیر a وS و r مقدار تابع یادگیر تقویتی توسط تابع ارزیابی[۱۴]، بِروزرسانی می‌شود. الگوریتم‌های یادگیری تقویتی، سعی می‌کنند که سیاست‌هایی را برای نگاشت حالت‌ها به عمل‌هایی که هر عامل باید در آن حالت انجام دهد، پیدا کنند.



شکل۱- مدل یادگیری تقویتی

در الگوریتم‌های یادگیری تقویتی، معمولاً محیط به صورت یک فرآیند تصادفی مارکوف[۱۵] (MDP) مدل می‌شود. یک MDP یک چهارتایی به صورت $< S, A, T, r >$ است که در آن S مجموعه متناهی از حالات و A مجموعه‌ای از عملیات قابل دسترس برای عامل و$T : S \times A \times S \to [0,1]$ تابع انتقال از حالت فعلی به حالت بعدی و $r : S \times A \to R$ تابع پاداش است. هدف، پیدا کردن سیاستی به صورت $\pi : S \to A$ است به گونه‌ای که میانگین پاداش دریافتی در طول زمان بیشینه گردد. برای هر خط مشی نظیر π که عامل می‌تواند دنبال کند، بر روی وضعیت‌ها تابعی به نام تابع ارزیابی تعریف می‌شود. الگوریتم Distributed Q-Learning یکی از تکنیک‌هایی است که برای تابع ارزیابی، استفاده می‌شود، که در سال ۲۰۰۰ در [۱۷] معررفی شده است. شبه‌کد الگوریتم Distributed Q-Learning در شکل ۲ آمده است. در این الگوریتم، برای هر عمل a در هر حالت S مقدار ارزش آن عمل ((Q(S,a)) مطابق رابطه‌ی ۱ تعیین می‌شود. در رابطه‌ی ۱، α نرخ یادگیری و $\gamma \in [۰,۱]$ فاکتور کاهش است. ایده‌ی اصلی در این الگوریتم، این است که سیاست کنونی عامل فقط زمانی بِروزرسانی می‌شود که پیشرفتی در ارزیابی تابع ارزش $Q_{i,max}$ رخ دهد. الگوریتم زمانی به پایان می‌رسد که سیاست بهینه برای مدت زمان معینی تغییر نکند.

$$q \leftarrow r + \gamma_{u \in A_i} \max Q_i(S', u)$$

$$Q_i(S,a) = f(x) = \begin{cases} (1-\alpha)Q_i(S,a), & if \;\; Q_i(S,a) \leq q \\ Q_i(S,a), & else \end{cases} \qquad (1)$$

Algorithm Distributed Q-Learning

1. Initialize;
2. **for all** $a \in A_i$ and $s \in S$ **do**
3. $Q_{i,max}(S,a) \leftarrow 0$; $\pi_i(S,a)$ $arbitrarily$
4. **end for.**
5. $s \leftarrow initial\ Sate$
6. **while** *s is not absorbing state* **do**
7. from *s* **select** *a* according to **EQ(2)** Selection Method based on π_i
8. **Apply** *a* and **observe** reward *r* and *next state S'*
9. $q \leftarrow r + \gamma \max_{u \epsilon A} Q_i(S', u)$
10. **if** $q \geq Q_i(S', a)$ **then**
11. $Q_{i,max}(S,a) \leftarrow (1-\alpha)Q_i(S', a) + \alpha q$
12. **end if.**
13. **if**$(Q_{i,max}(S, \arg\max_{u \in A} \pi_i(S,u) \neq \max_{u \in A} Q_{i,max}(S,u))$ **then**
14. **select** *a* random action $a_{max} \in \arg\max_{u \in A_i} Q_{i,max}(S,u)$
15. $\forall b \in A_i \;\; \pi_i(S,b) \leftarrow f(x) = \begin{cases} 1, & if\ b = a_{max} \\ 0, & else \end{cases}$
16. **end if.**
17. $S \leftarrow S'$
18. **end.**

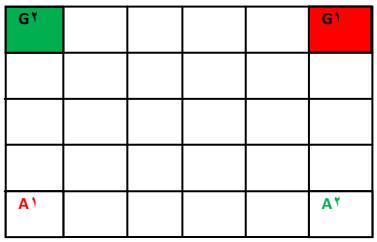
شکل۲- شبه‌کد الگوریتم Distributed Q-Learning

معمولاً برای انتخاب عمل در هر حالت (قسمت Policy)، از روش توزیع بولتزمن(رابطه‌ی ۲) استفاده می‌شود. در رابطه‌ی ۲، m تعداد اعمال مجاز برای حالت S و τ یک ثابت است و Q(S,a) مقدار تابع ارزیابی حالت S را هنگامی که عمل a انجام می‌گیرد، نشان می‌دهد.

$$\pi_1(S) = \arg\max \left(\frac{e^{\frac{Q(S,t)}{\tau}}}{\sum_{j=1}^{m} e^{\frac{Q_j(S,t)}{\tau}}} \right) \qquad (2)$$

۲-۲- بازی‌های مارکوف

بازی مارکوف، تعمیم یافته‌ی مسئله‌ی تصمیم‌گیری مارکوف (MDP) به حالت چندعامله است. یک بازی مارکوفی، یک چندتایی $< n, S, A_1, \dots, A_n, T, r_1, \dots r_n >$ است که n تعداد عامل‌های موجود در محیط و S مجموعه‌ی متناهی از حالات محیط و A_i مجموعه‌ی متناهی از اعمال قابل دسترس برای عامل i ، $(A_1, \dots, A_n)$ مجموعه‌ی اعمال قابل دسترس برای برای هر کدام ازعامل‌های موجود در محیط است) و $T_i : S \times A_i \times S \rightarrow [0,1]$ تابع انتقال و R_i تابع پاداش i امین عامل $R_i : S \times A_i \rightarrow R$ است.($R_1, R_2, \dots, R_n$ تابع پاداش بدست آمده برای هر یک از عامل‌های موجود در محیط است) . یکی از بازی‌های استفاده شده برای بازی‌های مارکوف چندعامله، بازی Grid World است که در[۵] معرفی شده است. در این بازی، دو عامل مستقل از مراحل ردیف پایین شروع به بازی می‌کنند و سعی می‌کنند که سلول هدفشان را در مراحل بالایی، پیدا کنند. یک عامل فقط می‌تواند در یک لحظه، به داخل یک سلول، حرکت کند. چهار عمل ممکن برای هر عامل وجود دارد: بالا، پایین، چپ، راست. اگر دو عامل سعی کنند که به خانه‌های یکسانی وارد شوند، بجز حالت هدف، به سلول قبلی خود بازگشت خورده و هر دو عامل، ۱- واحد جریمه می‌شوند. به محض اینکه یکی از عامل‌ها به حالت هدف رسید، بازی به پایان می‌رسد و عاملی که به حالت هدف رسیده است ۱۰۰+ واحد پاداش دریافت می‌کند. هدف یک عامل این است که با مینیمم تعداد حرکات بتواند به حالت هدف برسد. یک سیاست (استراتژی)، دنباله‌ی اعمال انجام گرفته از حالت شروع به حالت پایان است. کوتاه‌ترین مسیری که اجازه دهد تا یک عامل بتواند هر چه زودتر به هدف برسد، یک استراتژی بهینه است. شکل ۳ نمونه‌ای از این بازی است. استراتژی بهینه در شکل ۳ شامل ۹ حرکت است.



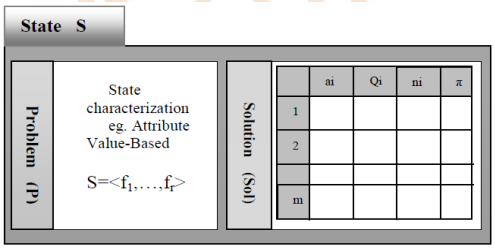
شکل ۳- نمونه‌ای از بازی Grid World

۳. الگوریتم پیشنهادی

در این بخش، یک الگوریتم جدید مبتنی بر یادگیری تقویتی توزیع شده برای عامل‌های مستقل، به منظور افزایش سرعت همگرایی در بازی‌های مارکوف، موسوم به Dist BSAQL ارائه شده است. یکی از چالش‌های مهم در الگوریتم‌های یادگیری تقویتی، سیاست انتخاب عمل[۱۶] است. برای این منظور، معمولا از توزیع بولتزمن (رابطه‌ی ۲) استفاده می‌شود. در رابطه‌ی ۲، پارامتر τ، پارامتر دما بوده و میزان تصادفی بودن جستجو را کنترل می‌کند و معمولا به دو صورت به‌کار برده می‌شود. اول اینکه، به صورت یک ثابت که در طول فرآیند یادگیری، مقدار آن همواره ثابت است، دوم اینکه، به صورت تابعی از زمان، که در طول فرآیند یادگیری، مقدار τ کاهش می‌یابد. در هر دو صورت، امکان تعیین مقدار دقیق و بهینه‌ی τ برای مسائل واقعی مشکل بوده و در صورتی که مقدار τ به درستی تعیین نشود، ممکن است الگوریتم هرگز به سمت سیاست بهینه همگرا نشود. در این مقاله، به منظور حل این مشکل، تابع جدیدی را پیشنهاد می‌کنیم که در آن نیازی به سِت کردن هیچ‌گونه پارامتر اضافی نبوده و مکانیزم انتخاب عمل، توسط مقادیر داخلی الگوریتم انجام شود. تابع پیشنهادی در رابطه‌ی ۳ آمده است. در رابطه‌ی ۳، m تعداد اَعمال مجاز برای حالت S و $n(S,a)$ تعداد دفعاتی است که تاکنون عمل a انتخاب شده است و $Q(S,a)$ مقدار تابع ارزیابی حالت S را هنگامی‌که عمل a انجام می‌گیرد، نشان می‌دهد.

$$\pi_{۲}(S) = \frac{e^{n(S,a)\times Q(S,a)}}{\sum_{i=۱}^{m} e^{n_i(S,a)\times Q_i(S,a)}} \qquad (۳)$$

در روش پیشنهادی، از رابطه‌ی ۳ برای انتخاب عمل عامل‌ها در هر حالت استفاده شده است. در الگوریتم پیشنهادی، به ازای هر یک از حالات محیط و به ازای هر عامل اصلی، از ساختار داده‌ی شکل ۴ استفاده می‌شود.



شکل ۴- ساختار داده یک عامل مجازی در الگوریتم پیشنهادی Dist BSAQL

ساختار داده شکل ۴، از دو قسمت به صورت $State(S) =< Prob, Q_{table} >$ ، تشکیل شده است، که Prob توصیف کننده مسئله و Q_{table} حاوی مقادیر تابع ارزیابی برای هر یک از اعمال حالات S است. در الگوریتم پیشنهادی، توصیف کننده مسئله (Prob) به صورت Prob(S)={m, <Up,

{Down, Right, Left>, index} تعریف می شود، که در آن، m تعداد اعمال هر حالت و مجموعه‌ی <Up ,Down ,Right ,Left> اعمال مجاز برای هر حالت و index، اندیس هر حالت است. جدول تابع ارزیابی برای هر مسئله $Q_{table}(S) =< \vec{E} >$ است که در آن بردار $\vec{E}$ به صورت $\vec{E} = (\vec{E}[1], \dots, \vec{E}[m])$ لیستی از تجربیات جمع آوری شده از محیط توسط عامل برای حالت S است. (m تعداد اعمال مجاز برای حالت S است). هر بردار $\vec{E}$ شامل یک چهارتایی $e =< a_i, Q_i, n_i, \pi_i >$ است که a_i عمل مجاز برای حالت S و n_i تعداد دفعاتی که عمل a_i بروز رسانی شده است و Q_i مقدار تخمین زده شده برای تابع ارزیابی $Q(S,a)$ است. π'_i احتمال انتخاب عمل a_i است که توسط تابع پیشنهادی (رابطه‌ی۳) محاسبه می شود. درالگوریتم Dist BSAQL، برای هر حالت و به ازای هر عامل اصلی، یک عامل مجازی، وجود دارد. عامل مجازی در هر حالت، مسئول یادگیری عمل بهینه در آن حالت است. در الگوریتم Dist BSAQL ، فرآیند یادگیری، براساس الگوریتم یادگیری تقویتی Distributed Q-Learning می‌باشد. هر بار که عامل اصلی $Aent_i^s$ وارد حالت جدید S می‌شود ، یک درخواست help به عامل مجازی $Aent'^s_i$ آن حالت، ارسال می‌کند. عامل مجازی با استفاده از مکانیزم جدید $X_i = Select\_index()$، که در آن از تابع پیشنهادی استفاده می‌کند، اقدام به انتخاب عمل بهینه با اندیس X_i می‌نماید. برای این منظور، به ازای هریک از لیست‌های $e =< a_i, Q_i, n_i, \pi_i >$ مقدار π'_i را با استفاده از رابطه‌ی ۳ محاسبه می‌کند. در واقع $\pi'_i(a_i)$ احتمال انتخاب عمل a_i را در حالت کنونی، نشان می‌دهد. بعد از محاسبه مقادیر احتمال هر یک از اعمال، برای انتخاب عمل نهایی به عنوان عمل بهینه عامل در حالت کنونی از روش max که اکثر محققین از آن استفاده کرده‌اند، استفاده می‌کند. یعنی عملی که دارای بزرگترین مقدار $\pi'_i(a_i)$ است به عنوان عمل بهینه انتخاب می‌شود و آن را به عنوان عمل بهینه به عامل اصلی $Aent_i^s$ برمی‌گرداند. سپس عامل اصلی $Aent_i^s$ ، عمل X_i را بر محیط اِعمال نموده و پاداش r_t را دریافت می‌کند و مقدار تابع ارزیابی $Q(S,x_i)$ را بِروزرسانی می‌کند و لیست Q_{table} را بر حسب مقدار تابع ارزیابی Q به صورت نزولی مرتب می‌کند و مقدار n_i را یک واحد افزایش می دهد. شبه کد الگوریتم Dist BSAQL در شکل ۵ آمده است.

Proposed Algorithm Dist BSAQL

1. **Let** *t* be the *global time* , γ the *discount factor* , *n* the number of mobile agents , s=s'$\in S$ to the initial State.
2. Repeat
3. S=S'
4. **for all** *agents* $i \in [1..n]$ do
5. $x_i = Send(help, agent'(S))$
6. **Select** action $a_i = S.SOL[x_i].a_i$
7. **Observe** *Successor state* $s' \in S$
8. **Set** *Learning Rate* $\alpha_i = \frac{1}{1+S.Q_{table}[x_i].n}$
9. **Set** $S.Q_{table}[x_i].Q_i$ according to **EQ(۱)**
10. **Set** $S.Q_{table}[x_i].n_i = S.Q_{table}[x_i].n_i + 1$
11. **Resort** *decrementally* the experience list *Q* in *S.Sol*
12. **end for.**
13. **Until** *Stop_Criterion()* becomes *true*
14. //----------
15. **Send (help,** $Agent'^{s'}_i$)
16. **begin:**
17. **Set** *index* x_i according to **EQ (3)**
18. **Return** $max(S.Q_{table}.\pi_i)$
19. **end.**

شکل۵- شبه‌کد الگوریتم پیشنهادی Dist BSAQL

۴. شبیه‌سازی‌های انجام شده در محیط Grid Wolrd و تحلیل آن‌ها

برای ارزیابی کارایی الگوریتم Dist BSAQL از یک بازی Grid World مطابق شکل۳ با ۳۰ حالت و دو عامل مستقل A_1 وA_2که دارای هدف‌های جداگانه‌ای هستند، استفاده شده است. هدف ما این است که نشان دهیم آیا الگوریتم Dist BSAQL که در آن از تابع بهبودیافته‌ی جدیدی برای انتخاب عمل استفاده شده است، عملکرد بهتری نسبت به زمانی که برای انتخاب عمل از تابع بولتزمن استفاده می‌شود، دارد؟ به همین منظور، الگوریتم

Dist BSAQL را با الگوریتم Distributed Q-Learning که شبه‌کد آن در شکل ۲ آمده است و در آن از توزیع بولتزمن(رابطه‌ی ۲) برای انتخاب عمل استفاده شده است، مقایسه کرده‌ایم.

آزمایش۱: تعیین مقدار پارامتر γ - یکی از فاکتورهای مهم تعیین‌کننده در نتایج الگوریتم‌های مبتنی بر یادگیریQ، مقدار نرخ کاهش$\leq$ ۰) ($۱ \leq \gamma$ است که تعیین مقدار دقیق آن، باعث کارایی بهتر الگوریتم‌ها، می شود. اگر$۱ \rightarrow \gamma$ باشد، اهمیت پاداش‌های تجمعی بلندمدت در فرآیند یادگیری، بسیار بالاتر از پاداش های بلافصل کوتاه‌مدت، در نظر گرفته می‌شود. و در مقابل، اگر$۰ \rightarrow \gamma$ باشد، اهمیت پاداش‌های بلافصل کوتاه‌مدت، بیشتر خواهد بود. برای این منظور، الگوریتم پیشنهادی و الگوریتم Distributed Q-Learning را به تعداد ۵۰۰ بار به ازای مقادیر مختلف γ اجرا کرده-ایم و نتیجه‌ی بدست آمده در جدول ۱ و ۲ آمده است. طبق نتایج بدست آمده در جدول ۲ مشاهده می‌شود که الگوریتم Dist BSAQL با هر سه مقدارγ صددرصد مواقع به سمت پاسخ بهینه همگرا می‌شود و میانگین پاداش‌های بدست آمده به ازای هر سه مقدار γتقریبا باهم برابر هستند ولی میانگین تعداد حرکات انجام شده، هنگامی که $\gamma = ۰.۹$ است، کمتر است. بنابراین، در همه آزمایشات انجام شده برای این الگوریتم، مقدار$\gamma = ۰.۹$ قرار داده می‌شود. در جدول ۱ مشاهده می‌شود که الگوریتم Distributed Q-Learning با هر سه مقدار γ صددرصد مواقع به سمت پاسخ بهینه همگرا می شود و پاداش‌های بدست آمده نیزتقریبا برابر است ولی تعداد حرکات لازم برای همگرایی هنگامی که $\gamma = ۰.۷$ است، کمتر می‌باشد. بنابراین، در همه آزمایشات انجام شده برای الگوریتمDistributed Q-Learning ، مقدار$\gamma = ۰.۷$ قرار داده می‌شود.

جدول۱- نتایج۵۰۰ بار اجرای الگوریتم Dist BSAQL بر روی محیط Grid World

episode	تعداد حرکات لازم برای همگرایی			میانگین تعداد حرکات انجام شده			پاداش			درصد همگرایی به سمت سیاست بهینه		
	0.5	0.7	0.9	0.5	0.7	0.9	0.5	0.7	0.9	0.5	0.7	0.9
200	13.56	11.27	13.24	30	14.4	17.2	100.15	100.15	100	100	100	100
400	10.09	13.01	8.22	28	84.3	14.4	100	100.25	100.02	100	100	100
600	8.15	8.21	7.09	29.5	21.8	15.3	100	109.65	100	100	100	100
800	7.50	7.3	7.61	25.4	16.5	17.9	100.92	100	100.02	100	100	100
1000	7.27	7.3.	7.30	31	24	17.8	100	100.04	100	100	100	100
1200	7.41	7.73	7.54	23.9	22.9	15.3	100	100.01	100	100	100	100
1400	7.74	7.17	8.09	27.5	40.5	17.7	100.02	127.41	100	100	100	100
1600	6.69	7.18	6.6	24	49.3	13.7	100	109.03	100	100	100	100
1800	6.63	7.20	6.69	24.9	165.8	15.9	100	118.98	109.95	100	100	100
2000	6.56	7.19	6.75	24.3	25.8	17.5	100	100.01	100	100	100	100
2200	6.88	7.44	6.71	24.5	50.7	11.7	100	137.76	100	100	100	100
2400	6.75	7.12	6.43	24.9	38.5	14.7	100	128.8	109.95	100	100	100
2600	6.93	6.88	6.30	33.9	48.5	29.2	100	102.8	100	100	100	100
2800	6.29	6.42	6.68	22	29.8	20.3	100	109.47	100.01	100	100	100
3000	6.27	6.41	6.85	22	638.8	13.2	100	100.07	100	100	100	100
میانگین	**7.64**	**7.89**	***7.47***	**26.38**	**84.77**	***16.78***	**100.07**	**109.62**	***101.33***	**100**	**100**	***100***

جدول۳- نتایج۵۰۰ بار اجرای الگوریتم Distributed Q-Learning بر روی محیط Grid World

episode	تعداد حرکات لازم برای همگرایی			میانگین تعداد حرکات انجام شده			پاداش			درصد همگرایی به سمت سیاست بهینه		
	0.5	*0.7*	0.9	0.5	*0.7*	0.9	0.5	*0.7*	0.9	0.5	*0.7*	0.9
200	۱۰,۷٤	۱۹,٥٤	۳۹,۲۱	۲۳,۷	۲۱,۷	۱۸,۹	۱۰۰,۱	۱۰۰,۰٥	۱۰۰,۰٥	۱۰۰	۱۰۰	۱۰۰
400	۱٦,٥۹	۱٥,۹۲	۲۸,٦۲	۲٦,۱	۱۹,۷	۱٦,۲	۱۰۹,٥	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰
600	۲٤,۰٦	۱٤,٥٦	۸,۸۱	۲٤,۷	۱۸,۲	۱٦,۸	۱۰۰,۰٥	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰

800	۱۱,٤۹	۸,۵۲	۱۰,۸۸	۲٤,۲	۱۷,۷	۱۹,۲	۱۰۰	۱۰۰,۰۳	۱۰۰	۱۰۰	۱۰۰	۱۰۰
1000	۹,۲۳	۱۲,۲۸	۸,٤۲	۲۵,۱	۱۹,۷	۱۷,٦	۱۰۰,۰۲	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰
1200	۸,٤۲	۹,۲۳	۱۰,۰۲	۲۵	۱۸,۵	۱۷	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰
1400	۱۰,٦۲	۷,۹٦	۷,۳٦	۲۳,٤	۲۰,۳	۱۵,۱	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰
1600	۸	۹,۱۰	۹,۵۰	۲٤,۲	۱۸,۸	۱۸,۵	۱۰۰	۱۰۰,۰۱	۱۰۰	۱۰۰	۱۰۰	۱۰۰
1800	۱۰۲۸	۸,۷۵	۸,۲٦	۲٦,۱	۱۹,۸	۱۸,٤	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰
2000	۷,٤٤	۱۰,۸۱	۸,۲۵	۲٦,٦	۱۹,٤	۱٤,۸	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰
2200	٦,۹	۷,۳۵	۸,۵۷	۲۸,۵	۱۹	۱٦,۷	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰
2400	۷,۷۸	۷,۷٦	۸,٤۳	۲۷,۷۵	۲۰,۳	۱٦,۸	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰
2600	٦,۸۷	٦,۸۲	۷,۰۲	۲٦,٤	۲۰,٤	۱٦,۸	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰
2800	۹,۰۹	٦,۵٦	۷,۱۹	۲۸,۷	۲۰,۲	۱٦,٤	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰
3000	۷,۹۹	٦,۷٤	٦,۵۲	۲۲,۸	۱٦,۲	۱٦,۵	۱۰۰	۱۰۰	۱۰۰,۰۱	۱۰۰	۱۰۰	۱۰۰
میانگین	۷۸,۲۱	**۱۰,۱۲**	۱۱,۸۰	۲۵,۵۵	**۱۹,۳۲**	۱۷,۰٤	۱۰۰,٦٤	**۱۰۰,۰۰٦**	۱۰۰,۰۰٤	۱۰۰	**۱۰۰**	۱۰۰

آزمایش ۲: تعیین مقدار دقیق τ برای الگوریتم Distributed Q-Learning – برای این منظور الگوریتم Distributed Q-Learning را به تعداد ۵۰۰ بار به ازای مقادیر مختلف τ اجرا کرده‌ایم. نتایج آزمایشات در جدول ۳ آمده است. در جدول ۳، می‌بینیم که زمانیکه $\tau = ۰.۰۵$ است، الگوریتم نتایج بهتری را ارائه می‌دهد. در همه‌ی آزمایشات انجام شده مقدار پارامتر τ برابر با ۰,۰۵ قرار می‌گیرد.

جدول ۳- تعیین مقدار τ برای الگوریتم Distributed Q-Learning

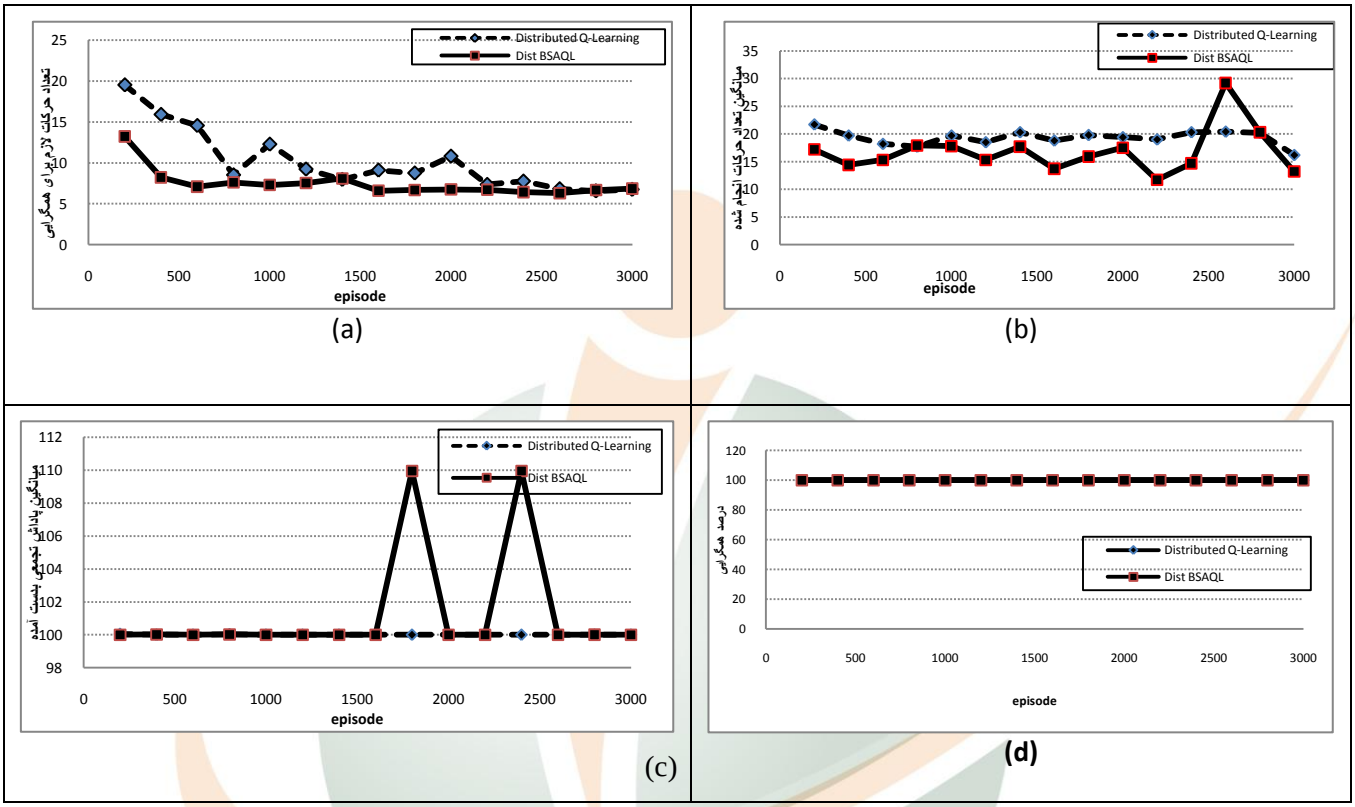
	$\tau = ۵$	$\tau = ٤$	$\tau = ۳$	$\tau = ۲$	$\tau = ۱$	$\tau = ۰.۵$	$\tau = ۰.۲$	$\tau = ۰.۱$	$\tau = ۰.۰۵$
درصد همگرایی	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰
تعداد حرکات لازم برای همگرایی	۱۱,۳۸	۱٦,٤۱	۱٤,۱۱	۱۳,۷۷	۱۵,۲۹	۱۱,۵۵	۱۳,۳۲	۱۹,٦۱	۱۰,۵٦

آزمایش۳: مقایسه کارایی الگوریتم پیشنهادی با الگوریتم Distributed Q-Learning از نظر تعداد حرکات لازم برای رسیدن به هـدف. برای این منظور، تعداد حرکات لازم برای همگرایی الگوریتم در مقایسه با الگوریتم دیگر مورد آزمایش قرار می گیرد. نمودار شکل ۶-a دو الگـوریتم را از لحاظ تعداد حرکات لازم برای همگرایی به سمت سیاست بهینه، مقایسه می‌کند. با توجه به نمودار می بینیم که تعداد حرکات لازم برای همگرایـی بـه سمت سیاست بهینه در الگوریتم پیشنهادی کمتر است. هر چه تعداد حرکات لازم برای رسیدن به سیاست بهینه، کمتر باشد، الگوریتم کاراتری خواهیم داشت.

آزمایش۴: مقایسه کارایی الگوریتم پیشنهادی با الگوریتم Distributed Q-Learning از نظر میانگین تعداد حرکات انجام شده، در ۵۰۰ بار اجرای الگوریتم‌ها مورد ارزیابی قرار گرفته است. شکل ۶- b نتایج را نشان می دهد. با توجه به شکل دیده می شود که عامل برای یادگیری سیاست بهینه در الگوریتم پیشنهادی، به تعداد حرکات کمتری نیاز دارد.

آزمایش۵: مقایسه کارایی الگوریتم پیشنهادی با الگوریتم Distributed Q-Learning از نظر میانگین پاداش تجمعی بدست آمده در هر اپیزود. نمودار شکل ۶-c الگوریتم‌ها را از لحاظ میانگین پاداش تجمعی بدست آمده در هر اپیزود، مقایسه می‌کند. هر چه میانگین پاداش تجمعی بدست آمده برای الگوریتمی بیشتر باشد، الگوریتم کاراتری خواهیم داشت. بهترین حالت زمانی است که هر دو عامل، همزمان به حالت هدف رسیده و ۲۰۰+ پاداش دریافت کنند. نتایج بدست آمده، حاکی از این است که الگوریتم Dist BSAQL با تعداد حرکات کمتر و بدست آوردن پاداش بیشتر، به سمت یک راه‌حل بهینه همگرا می‌شود و برتری الگوریتم پیشنهادی را نسبت به الگوریتم دیگر نشان می‌دهد.

آزمایش۶ : مقایسه کارایی الگوریتم پیشنهادی با الگوریتم Distributed Q-Learning از نظر درصد همگرایی به سمت پاسخ بهینه در هر اپیزود، در ۵۰۰ بار اجرای الگوریتم‌ها مورد ارزیابی قرار گرفته است. شکل ۶- d نتایج را نشان می دهد. با توجه به شکل دیده می شود که هر دو الگوریتم صد در صد مواقع به سمت پاسخ بهینه همگرا می‌شوند.



شکل ۶- مقایسه دو الگوریتم Dist BSAQL و Distributed Q-Learning در ۵۰۰ بار اجرای دو الگوریتم. (a) مقایسه دو الگوریتم از لحاظ تعداد حرکات لازم برای رسیدن به هدف در ۵۰۰ بار اجرای الگوریتم‌ها. (b) مقایسه دو الگوریتم از لحاظ میانگین تعداد حرکات انجام شده در ۵۰۰ بار اجرای الگوریتم‌ها (c) مقایسه دو الگوریتم از لحاظ میانگین پاداش‌های بدست آمده در ۵۰۰ بار اجرای الگوریتم‌ها . (d) مقایسه دو الگوریتم از لحاظ درصد همگرایی به سمت پاسخ بهینه در ۵۰۰ بار اجرای الگوریتم‌ها.

۵. بررسی رفتار الگوریتم و تحلیل آن

در این بخش تحلیلی برای نحوه‌ی عملکرد الگوریتم پیشنهادی ارائه می شود. که در آن برتری تابع $\pi_2(S)$ (رابطه‌ی ۳) نسبت به تابع $\pi_1(S)$ (رابطه-ی ۲) به دو طریق ۱- آزمایش و ۲-تحلیل ریاضی ارائه گردیده است. می خواهیم نشان دهیم که در روش پیشنهادی، تابع $\pi_2(S) = \frac{e^{nQ}}{\sum_i e^{n_i Q_i}}$ نسبت به تابع $\pi_1(S) = \frac{e^{\frac{Q}{\tau}}}{\sum_i e^{\frac{Q_i}{\tau}}}$ با سرعت بیشتری به سمت پاسخ بهینه همگرا می‌شود. به عبارت دیگر، آهنگ تغییر $\pi_2(S)$ نسبت به Q بیشتر از آهنگ تغییر $\pi_1(S)$ نسبت به Q است.

۵-۱- بررسی با استفاده از آزمایش‌های مختلف

برای نشان دادن برتری رفتار تابع انتخاب عمل پیشنهادی، الگوریتم Dist BSAQL برای حالت S. و عمل a_1 با توجه به مقادیر مختلف n مورد بررسی قرار گرفته ونتایج زیر بدست آمده است.

نتیجه‌ی ۱: با توجه به نمودار شکل های ۷ و ۸ و۹ می بینیم که با افزایش n رشد$\pi_2(S)$ خیلی سریعتر از رشد $\pi_1(S)$ است.

نتیجه‌ی ۲: نمودار شکل ۸ و۹ و ۱۰ نشان می‌دهد که با افزایشn ، مقدار تابع Q(S,a) در رابطه‌ی ۱، افزایش می‌یابد.

نتیجه‌ی ۳: نمودار شکل ۱۱ نشان می‌دهد که با افزایش مقدار Q(S,a)، مقدار تابع $\pi_1(S)$ افزایش می‌یابد.

از آنجاییکه همواره $lim_{t\to\infty} Q_t(S,a) = \infty$ است، طبق نتیجه‌ی ۲، مقدار $Q_t(S,a)$ افزایش می‌یابد و طبق نتیجه‌ی ۱، با افزایش n، تابع $\pi_2(S)$ سریعتر از $\pi_1(S)$ رشد می‌کند. با توجه به مطالب قبلی، نتیجه می‌شود که بایستی با افزایش مقدار $Q_t(S,a)$ تابع $\pi_2(S)$ سریعتر از $\pi_1(S)$ رشد کند. نمودار شکل ۱۲ نتیجه‌ی بدست آمده را تایید می‌کند.

۵-۲- تحلیل ریاضی

برای سادگی محاسبات توابع $\pi_1(S)$ و $\pi_2(S)$ را به ترتیب به صورت رابطه‌ی ۴ و ۵ بازنویسی می‌کنیم.

$$\pi_1(S) = e^{\frac{Q}{\tau}} \qquad (۴)$$

$$\pi_2(S) = e^{nQ} \qquad (۵)$$

آهنگ تغییر $\pi_1(S)$ نسبت به Q با پارامتر $\tau = ۰.۰۵$ در رابطه‌ی ۶ آمده است.

$$\frac{\Delta\pi_1(S)}{\Delta Q} = \frac{1}{\tau}e^{\frac{1}{\tau}} = \frac{1}{0.05}e^{\frac{1}{0.05}} = 20e^{20Q} \qquad (۶)$$

آهنگ تغییر $\pi_2(S)$ نسبت به Q در رابطه‌ی ۷ آمده است.

$$\frac{\Delta\pi_2(S)}{\Delta Q} = \frac{d\pi_2(S)}{dn} \times \frac{dn}{dQ}$$

$$\frac{\Delta\pi_2(S)}{\Delta Q} = Qe^{nQ} \times \frac{dn}{dQ} \qquad \frac{\Delta\pi_2(S)}{\Delta Q} \qquad (۷)$$

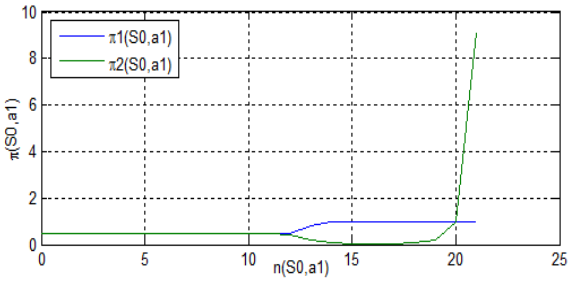
با مقایسه‌ی رابطه‌ی ۶ و۷ نتایج زیر بدست می‌آید.

۱. تابع $\frac{\Delta\pi_1(S)}{\Delta Q}$ همواره مثبت است. در رابطه‌ی ۶، بدلیل مثبت بودن Q مقدار تابع $\frac{\Delta\pi_1(S)}{\Delta Q}$همواره مثبت است.

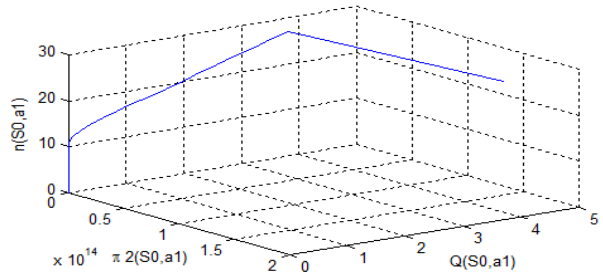
۲. تابع $\frac{\Delta\pi_2(S)}{\Delta Q}$ همواره مثبت است. طبق نمودار شکل ۱۰ با افزایش n مقدار Q(S, a) همواره افزایش می‌یابد. بنابراین $\frac{dn}{dQ} > 0$ و از طرفی $n > 0$ و $Q > 0$ است. در نتیجه، $\frac{\Delta\pi_2(S)}{\Delta Q}$ همواره مثبت است.

۳. آهنگ رشد $\frac{\Delta\pi_1(S)}{\Delta Q}$ نسبت به $\frac{\Delta\pi_2(S)}{\Delta Q}$ کمتر است.

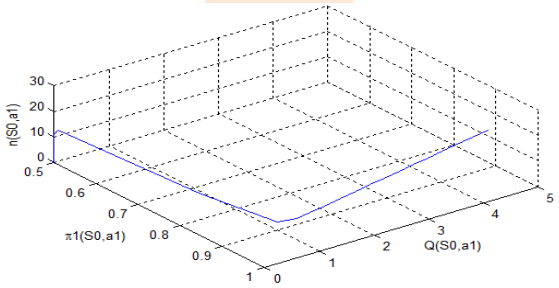
در رابطه‌ی ۶، e^{20Q} در مقدار ثابت ۲۰ ضرب شده است، این در حالی است که در رابطه‌ی ۷، e^{nQ} در متغیر Q ضرب شده است. با افزایش n، مقدار Q افزایش می‌یابد. بنابراین، آهنگ رشد $\frac{\Delta\pi_1(S)}{\Delta Q}$ نسبت به $\frac{\Delta\pi_2(S)}{\Delta Q}$ کمتر است. نمودار شکل ۱۲ نیز نتیجه‌ی بدست آمده را تایید می‌کند.



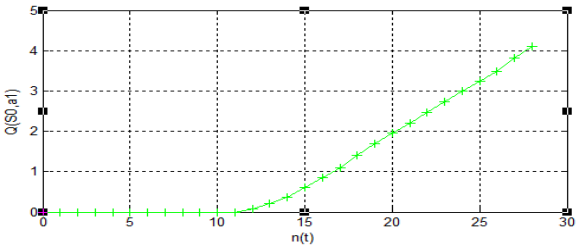
شکل ۷- بررسی رشد $\pi_1(S)$ و $\pi_2(S)$ بر حسب مقادیر مختلف n



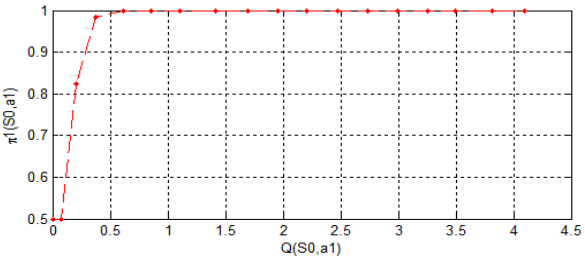
شکل ۸- بررسی رشد $\pi_2(S)$ بر حسب مقادیر مختلف n



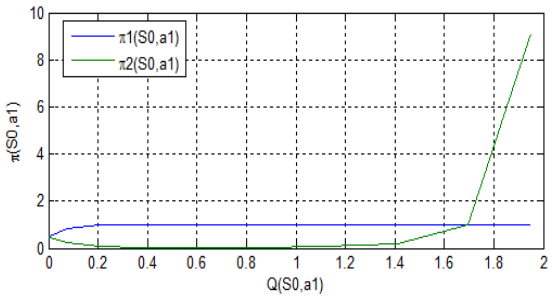
شکل ۹- بررسی رشد $\pi_1(S)$ بر حسب مقادیر مختلف n



شکل ۱۰- بررسی رشد *Q(S,a)* بر حسب مقادیر مختلف n



شکل ۱۱- بررسی رشد $\pi_1(S)$ بر حسب مقادیر مختلف *Q(S,a)*



شکل ۱۲- بررسی رشد $\pi_1(S)$ و $\pi_2(S)$ بر حسب مقادیر مختلف *Q(S,a)*

۶. نتیجه‌گیری

در این مقاله، الگوریتم جدید Dist BSAQL برای حل بازی‌های مارکوفی بر اساس یادگیری تقویتی توزیع‌شده، که در آن از تابع جدیدی برای انتخاب عمل در هر حالت استفاده شده است، ارائه گردید. نتایج بدست آمده، با الگوریتم‌های موجود مقایسه شد. بر طبق نتایج بدست آمده، الگوریتم Dist BSAQL در مقایسه با الگوریتم Distributed Q-Learning ، هم از لحاظ سرعت همگرایی به پاسخ بهینه و هم از لحاظ میانگین پاداش تجمعی بدست آمده و نیز تعداد حرکات لازم برای همگرایی به سمت سیاست بهینه، از کارایی بسیار خوبی برخوردار است. بنابراین استفاده از تابع پیشنهادی، برای انتخاب عمل به جای توزیع بولتزمن پیشنهاد می‌شود.

۷. مراجع

[۱] N. Vlassia; " *A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence*", Amesterdam:Morgan Claypool Publishe,۲۰۰۷.

[۲] L. Busoniue; R. Babuska and B. D. Schutter , *"A Comperhensive Survey of Multi agent Reinforce ment Learning "* , IEEE Transactions on Systems , MAN , and Cybernatics part C: Appilications and Reviews, Vol. ۳۸ ,No. ۲ ,March ۲۰۰۸.

[۳] Y. Shoham; *"Multi-agent Systems: Algorithmic Game Theoretic and Logical Foundations*", Cambridge University Press, ۲۰۰۹.

[٤] C. Boutilier; "*Sequential optimality and coordination in multi-agent systems*", In Proceedings of the ١٦th International Joint Conference on Artificial intelligence , Volume ١, Morgan Kaufmann Publishers Inc. , Stockholm, Sweden, ١٩٩٩.

[٥] J. Hu and M. P. Wellman, "*Nash Q-Learning for General-Sum Stochastic Games*", Journal of Machine Learning Research, vol. ٤, pp. ١٠٣٩-١٠٦٩, ٢٠٠٣.

[٦] M. Littman; "*Markov Games as a Framework for Multi-agent Reinforcement Learning*", in: ١١th Int. Conf. on Machine Learning, San Francisco, Morgan Kaufmann, pp. ١٥٧-١٦٣،١٩٩٤.

[٧] K. Tuyls; A. Nowe; T. Lenaerts; and B. Manderick . "*An Evolutionary Game Theoretic perspective on Learning in Multi-Agent Systems". Synthese, section Knowledge, Rationality and Action*, ١٣٩, issue ٢:٢٩٧–٣٣٠, ٢٠٠٤.

[٨] L. Matignon; G. Laurent; N. Le Fort-Piat , "*Coordination of independent learners in cooperative Markov games.*",Thechnical Report , Institute FEMTO-ST, May ٢٠٠٩.

[٩] M. Littman , "*Markov Games as a Framework for Multi agent Reinforcement Learning*" , In ١١th International Conference on Machine Learning , San Fransisco , Morgan Kaufmann , pp. ١٥٧-١٦٣ ,١٩٩٤.

[١٠] M. Song ; G. Gu and G. Zhang , " *Pareto_Q Learning Algorithm for Cooperative Agents in General Sum Games*", In Multiagent Systems and Applications , Vol.٣٦٩٠ : Springer, Berlin/Heidelberg , pp.٥٧٦-٥٧٨ , ٢٠٠٥.

[١١] M. Laur ; M. Reidmiller ; "*An Algorithm for Distributed Reinforcement Learning in Cooperative Multi-agent Systems*" , In proc. ١٧th ICML ,Morgan Kaufmann , San Francisco ,CA , pp.٥٣٥-٥٤٢،٢٠٠٠.

[١٢] S. Kapetanakis and D.Kudenko , " *Reinforcement Learning of Coordination in Hetergenous Cooperative Multi-agent Systems*" , In proc. Of AAMAS '٠٤ ,PP.١٢٥٨-١٢٥٩ , ٢٠٠٤.

[١٣] M. Lauer and M. Riedmiller, "*Reinforcement Learning for Stochastic cooperative Multi-agent Systems*", In Proceeding of AAMAS ٢٠٠٤, New York , NY, ACM Press , pp. ١٥١٤-١٥١٥, July ٢٠٠٤.

[١٤] Richard S. Sutton ; Andrew G. Barto, *" Reinforcement Learning:An Introduction*" , The MIT Press, London , ١٩٨٩.

[١٥] L. Matignon; G. J. Lauent and N. L.Front-piat , " *Hysteretic Q-Learning : An Algorithm for Decentralized Reinforcement Learning in Cooperative Multi-agent Teams* " , In Poceedings of IEEE/RSJ International Conference on Intelligence Robots and Systems IROS ,San diego , CA , USA, PP.٦٤-٦٩,Nov.٢٠٠٧.

[١٦] T. Gabel and M. Riedmiller, "*CBR for state value function Approximation in Reinforcement Learning*", Proceeding of the Inter. Conference on Case Based Learning ٢٠٠٥ (ICCBR ٢٠٠٥) , Springer , Chicago, USA, Aug ٢٠٠٥.

[۱۷] K. Tumer;A. K. Agogino,and D. H. Wolpert, "*Learning sequences of Actions in Collectives of Autonomous Agents*", Proceedings of the International Conference on Autonomous Agents, No. ۲, pp. ۳۷۸–۳۸۵, ۲۰۰۲.

---

۱ Multi agent Systems
۲ Reinforcement Learning
۳ Robustness
٤ Independent Learner(IL)
٥ Joint Action Learner(JAL)
٦ Markov games
۷ Markov Decision Process
۸ Cooperative
۹ Non-cooperative
۱۰ Zero-sum Games
۱۱ General-sum Games
۱۲ Equilibrium points
۱۳ Nash Equilibrium
۱٤ Value Function
۱٥ Markov Decision Process
۱٦ Exploration Policy