حل مساله مجموعه مستقل ماکزیمال توسط اتوماتاهای یادگیر توزیع شده

محمد علیپور محمدرضا میبدی

آزمایشگاه سیستمهای نرم افزاری
دانشکده مهندسی کامپیوتر و فناوری اطلاعات
دانشگاه صنعتی امیرکبیر
تهران ایران

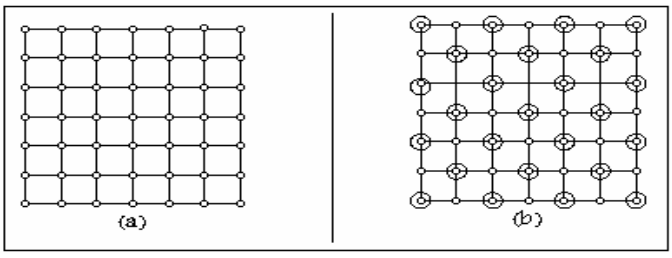
چکیده

مجموعه مستقل ماکزیمال در یک گراف، مجموعه‌ای از رئوس می باشد که هیچ دو رأسی در آن با یکدیگر همسایه نبوده و همچنین زیر مجموعهٔ هیچ مجموعهٔ مستقل بزرگتری نمی باشد. این مساله، از نوع مسائل NP-Complete بوده و دارای هزینه اجرایی از مرتبه نمایی است و بهمین دلیل الگوریتمهای تقریبی متعددی برای حل آن گزارش شده است که الگوریتم های شبکه های عصبی، الگوریتم های ژنتیکی، منجمد سازی فلزات از ان جمله میباشند. آتوماتای یادگیر یک ابزار جستجوی عمومی می باشد و برای حل تعدادی از مسائل NP-Complete بکار رفته است. در این مقاله با استفاده از آتوماتای یادگیر توزیع شده، سه الگوریتم جدید برای حل مجموعه مستقل ماکزیمال براساس اتوماتای یادگیر توزیع شده پیشنهاد و کارایی آنها بر روی نمونه هایی از مسائل مجموعه مستقل ماکزیمال آزمایش گردیده است.

کلمات کلیدی: مجموعه مستقل ماکزیمال، اتوماتای یادگیر، اتوماتای یادگیر توزیع شده

۱- مقدمه

یک مجموعه مستقل در یک گراف، مجموعه‌ای از رئوس می باشد که هیچ دو رأسی در آن با یکدیگر همسایه نیستند. مجموعه مستقل با بیشترین کاردینالیتی را مجموعه مستقل ماکزیمم[1] و مجموعه مستقل ماکزیمال[2]، مجموعه مستقلی می باشد که زیر مجموعهٔ هیچ مجموعهٔ مستقل دیگری نمی باشد. گره‌هایی از گراف داده شده در شکل 1(a) که تشکیل مجموعه مستقل ماکزیمال می دهند، در شکل 1(b) مشخص شده‌اند[1].



شکل ۱: (a) گراف ورودی، (b) گره‌هایی که تشکیل مجموعه مستقل ماکزیمال می دهند.

[1] Maximum Independent Set
[2] Maximal Independent Set

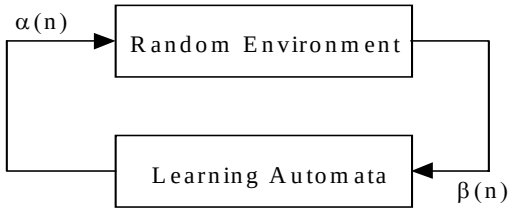
در [3] گزارش شده است که مجوعه مستقل ماکزیمال یک درخت در زمان خطی قابل محاسبه می باشد. با توجه به اینکه مساله مجموعه مستقل ماکزیمال، از نوع مسائل NP-Complete بوده الگوریتمهای تقریبی متعددی برای حل آن گزارش شده است که الگوریتم های CMPF,[3] ARH[4] و PCAOP[5] از آن جمله هستند. یک الگوریتم ترتیبی حریصانه‌ای با حد پایین $\Omega\left(\frac{\log n}{\log\log n}\right)$ نیز توسط اندرو[6] و گولدبرگ[7] [۷] پیشنهاد شده است. الگوریتمهای موازی نیز برای مساله مجموعه مستقل ماکزیمال طراحی شده‌اند که از آن جمله میتوان به الگوریتم معین PRAM با مرتبه زمانی $O(\log^4 n)$ با استفاده از $O\left(\left(\frac{n}{\log n}\right)^3\right)$ پردازنده[۸] و الگوریتم تصادفی PRAM با مرتبه زمانی $O(Logn)$ با استفاده از O(m) پردازنده [۶] [۹] اشاره کرد. الگوریتم‌های توزیع شده نیز برای این مسئله طراحی شده‌اند [5]. مساله مجموعه مستقل ماکزیمال عمدتاً بعنوان ابزاری برای حل مسائل شبکه مانند انتخاب(election)، حل بن بست (deadlock)، طراحی شبکه ارتباطی بصورت گسترده‌ای استفاده می‌شود.

در این مقاله یک الگوریتم جدید مبتنی بر اتوماتای یادگیر توزیع شده که بر اولین بار در [2] معرفی گردیده است برای حل مساله مجموعه مستقل ماکزیمال پیشنهاد گردیده است. این الگوریتم قادراست جوابهای بهینه ویا نزدیک به بهینه را برای هر دو نوع متقارن و غیر متقارن مجموعه مستقل ماکزیمال تولید نماید. الگوریتم پیشنهادی با الگوریتمهای گزارش شده ARH،CMPF وPCAOP مقایسه گردیده است. نتایج این مقایسه سرعت الگوریتم پیشنهادی و کیفیت جوابهای بدست آمده را تائید می گردد

ادامه مقاله بصورت زیر سازماندهی شده است. در بخش ۲ مقاله اتوماتای یادگیر بطور خلاصه معرفی می شود. سپس در بخش ۳ اتوماتای یادگیر توزیع شده شرح داده میشود. الگوریتم پیشنهادی در بخش ۴ و نتایج شبیه سازیها در بخش ۵ گزارش شده است. بخش پایانی نتیجه گیری میباشد.

۲- اتوماتای یادگیر[8]

اتوماتای یادگیر یک مدل انتزاعی است که تعداد محدودی عمل را می تواند انجام دهد. هر عمل انتخاب شده توسط محیطی احتمالی ارزیابی شده و پاسخی به اتوماتای یادگیر داده می شود. اتوماتای یادگیر از این پاسخ استفاده نموده وعمل خود را برای مرحله بعد انتخاب می کند. شکل ۱ ارتباط بین اتوماتای یادگیر و محیط را نشان می دهد.



شکل ۱: ارتباط بین اتوماتای یادگیر و محیط

محیط[9]: محیط را می توان توسط سه تایی $E \equiv \{\alpha, \beta, c\}$ نشان داد که در آن $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه ورودیها، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_m\}$ مجموعه خروجیها و $c \equiv \{c_1, c_2, ..., c_r\}$ مجموعه احتمالهای جریمه می باشد. هر گاه β مجموعه دو

[3] Continuous Multivariable Polynomial Formulations
[4] Annealed Replication Heuristic(
[5] Probabilistic Constructive Approach To Optimization Problems
[6] Andrew
[7] Goldberg
[8] Learning Automata
[9] Environment

عضوی باشد ، محیط از نوع P می باشد. در چنین محیطی $\beta_1 = 1$ به عنوان جریمه و $\beta_2 = 0$ به عنوان پاداش در نظر گرفته می شود. در محیط از نوع Q ، $\beta(n)$ می تواند به طور گسسته یک مقدار از مقادیر محدود در فاصله [0,1] و در محیط از نوع S، $\beta(n)$ متغیر تصادفی در فاصله [0,1] است. c_i احتمال اینکه عمل α_i نتیجه نامطلوب داشته باشد می باشد. در محیط ایستا[10] مقادیر c_i بدون تغییر می مانند، حال آنکه در محیط غیر ایستا[11] این مقادیر در طی زمان تغییر می کنند. اتوماتاهای یادگیربه دو گروه با ساختار ثابت و با ساختار متغیر تقسیم بندی میگردند. در ادامه به شرح مختصری درباره اتوماتای یادگیر با ساختار متغییر که در این مقاله از آنها استفاده شده است می پردازیم.

اتوماتای یادگیر با ساختار متغیر[12]: اتوماتای یادگیر با ساختار متغیر توسط ۴ تائی $\{\alpha, \beta, p, T\}$ نشان داده می شود که در آن $\alpha \equiv \{\alpha_1, \alpha_2, \cdots, \alpha_r\}$ مجموعه عملهای اتوماتا، $\beta \equiv \{\beta_1, \beta_2, \cdots, \beta_m\}$ مجموعه ورودیهای اتوماتا، $p \equiv \{p_1, p_2, \cdots, p_r\}$ بردار احتمال انتخاب هر یک از عملها، و $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ الگوریتم یادگیری می باشد. در این نوع از اتوماتاها، اگر عمل α_i در مرحله n ام انتخاب شود و پاسخ مطلوب از محیط دریافت نماید، احتمال $p_i(n)$ افزایش یافته و سایر احتمالها کاهش می یابند. و برای پاسخ نامطلوب احتمال $p_i(n)$ کاهش یافته و سایر احتمالها افزایش می یابند. در هر حال، تغییرات به گونه ای صورت می گیرد تا حاصل جمع $p_i(n)$ ها همواره ثابت و مساوی یک باقی بماند. الگوریتم زیر یک نمونه از الگوریتمهای یادگیری خطی برای اتوماتای یادگیر با ساختار ثابت است.

الف- پاسخ مطلوب

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)]$$
$$p_j(n+1) = (1-a)\, p_j(n) \qquad j \neq i \quad \forall j$$

ب- پاسخ نامطلوب

$$p_i(n+1) = (1-b)\, p_i(n)$$
$$p_j(n+1) = \frac{b}{r-1} + (1-b)\, p_j(n) \qquad j \neq i \quad \forall j$$

در روابط فوق، پارامتر پاداش و a پارامتر پاداش و b پارامتر جریمه می باشد. با توجه به مقادیر a و b سه حالت را می توان در نظر گرفت. زمانیکه a و b با هم برابر باشند، الگوریتم ر، L_{RP}[13] زمانیکه b از a خیلی کوچکتر باشد، الگوریتم را $L_{R\varepsilon P}$[14] و زمانیکه b مساوی صفر باشد، الگوریتم را L_{RI}[15] می نامیم. برای مطالعه بیشتر در باره اتوماتاهای یادگیر می توان به [4],[5],[6],[7] مراجعه کرد.

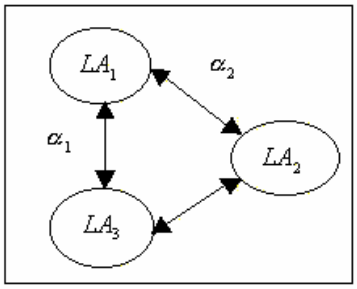
۳- اتوماتای یادگیر توزیع شده[16]

اتوماتای یادگیر توزیع شده (DLA) ، شبکه‌ای از اتوماتای یادگیر است که برای حل مساله خاصی با یکدیگر همکاری می نمایند [2]. تعداد اقدامهای یک اتوماتا در DLA برابر تعداد اتوماتاهای متصل به اتوماتا یادگیر فوق می باشد. انتخاب یک اقدام توسط یک اتوماتا در شبکه، اتوماتای متناظر با این اقدام را فعال می سازد. بعنوان مثال در شکل ۲ هر اتوماتای یادگیر دارای دو اقدام می باشد. انتخاب اقدام α_2 توسط اتوماتای یادگیر LA_1، اتوماتای یادگیر LA_3 را فعال خواهد کرد. اتوماتای یادگیر فعال شده (LA_3) بنوبه‌ی خود یکی از اقدامهای خود را انتخاب می کند که در نتیجه آن یکی از اتوماتاهای یادگیر متصل به اتوماتای یادگیر که متناظر با اقدام انتخاب شده می باشد فعال می شود. در هر زمان فقط یک اتوماتای یادگیر در شبکه فعال میباشد. بطور رسمی DLA را میتوان توسط گراف $DLA = (V, E)$ که $V = \{LA_1, LA_2, ..., LA_n\}$ مجموعه اتوماتاهای یادگیر و n تعداد اتوماتاهای یادگیر در DLA و $E \subset V \times V$ مجموعه لبه‌های گراف می باشد، تعریف کرد. لبه (i, j) اقدام j اتوماتای یادگر LA_i را نشان می دهد. LA_j زمانی

---

[10] Stationary
[11] Non-Stationary
[12] Variable Learning Automata
[13] Linear Reward Pealty
[14] Linear Reward Epsilon Penalty
[15] Linear Reward Inaction
[16] Distributed Learning Automata

فعال خواهد شد که اقدام j اتاماتوی یادگیر LA_i انتخاب شود. تعداد اقدامهای اتوماتای یادگیر LA_k ($k = 1,2,...,n$) گره k برابر درجه‌ی خروجی آن گره می باشد. برای اطلاعات بیشتر در باره اتوماتای یادگیر توزیع شده میتوان به مراجع [17] و [2] مراجعه کرد.



شکل ۲: اتوماتای یادگیر توزیع شده (DLA) با ۳ اتوماتا یادگیر

۴-۱- الگوریتم پیشنهادی اول

ابتدا شبکه‌ای از اتوماتاهای یادگیر متناظر[17] با گراف ورودی نمونه مساله MIS ایجاد می شود. در این شبکه به هر گره گراف یک اتوماتای یادگیر با ساختار متغیر تخصیص داده میشود .مجموعه اقدامهای یک اتوماتای یادگیر که به یک گره منصوب شده است برابر است با مجموعه گره های گراف به جز گره ای که آن اتوماتای یادگیر به آن تخصیص داده شده است و گره های همسایه آن گره. توجه کنید که در بعضی مواقع کلمات گره و اتوماتای یادگر به جای همدیگر استفاده میشوند. در حین اجرای الگوریتم یک اتوماتای یادگیر (گره) میتواند در یکی از دو حالت فعال و یا غیر فعال قرار گیرد. در شروع الگوریتم کلیه اتوماتاهای یادگیر در DLA فعال هستند.

اکنون به توصیف الگوریتم پیشنهادی می پردازیم. در گام نخست یکی از گره های گراف بصورت تصادفی انتخاب و در مجموعه MIS درج میگردد و سپس اتوماتای یادگیر متناظر با این گره که اتوماتای جاری نامیده میشود و همچنین کلیه اتوماتاهای یادگیر همسایهٔ اتوماتای یادگیر جاری غیر فعال می شوند. همچنین اقدامهای متناظر با اتوماتاهای یادگیر غیر فعال شده در تمام اتوماتاهای یادگیر فعال غیر فعال میگردند ولی از لیست اقدامهای آنها حذف نمیگردند. اتوماتای یادگیر جاری بنوبهٔ خود یکی از اقدامهایش را برطبق بردار احتمال اقدامها انتخاب کرده و گره متناظر با اقدام انتخاب شده در MIS درج میگردد و سپس اتوماتای یادگیر متناظر با این اقدام وکلیه اتوماتاهای همسایه این اتوماتا غیر فعال میگردد. این اقدام و اقدامهای متناظر با کلیه اتوماتاهای یادگیر همسایهٔ آن در همه اتوماتاهای یادگیر موجود در DLA غیر فعال می شوند ولی از لیست اقدامهای آنها حذف نمیگردد. فرآیند انتخاب اقدام و درج آن در MIS و غیر فعال سازی اتوماتای یادگیر متناظر با آن اقدام و تمامی همسایگانش در DLA، تا غیر فعال شدن همه گره‌های (اتوماتاهای یادگیر) موجود در گراف(DLA) ادامه مییابد. پس از غیر فعال شدن همه اتوماتاهای یادگیر DLA، کاردینالیتی مجموعه مستقل ماکزیمال(MIS)، تولید شده محاسبه میشود و با کاردینالیتی بهترین MIS که تابحال ایجاد شده است مقایسه میگردد. بر طبق نتیجه این مقایسه، بردار احتمال اقدام اتوماتاهای یادگیر DLA بروزرسانی خواهد شد. سپس همه اتوماتاهای یادگیر در DLA و اقدامهای آنها که قبلا غیر فعال شده بودند مجددا فعال میشوند و فرایند تشکیل یک مجموعه مستقل دیگر آغاز میگردد.

نحوه‌ی بروز رسانی بردار احتمال اقدام برطبق الگوریتم یادگیری می باشد. اگر الگوریتم یادگیری $L_{R\varepsilon P}$ باشد در این صورت اگر کاردینالیتی MIS ایجاد شده در تکرار t بزرگتر و یا مساوی کاردینالیتی بهترین MIS ایجاد شده تاکنون باشد و در این تکرار اتوماتای یادگیر j از مجموعه اقدامهای مجاز خود اقدام i را انتخاب کرده باشد، احتمال انتخاب اقدام i طبق رابطه‌ی زیر افزایش خواهد یافت:

$$p_i(t+1) = p_i(t) + a[1 - p_i(t)]$$

احتمال انتخاب سایر اقدامهای اتوماتای یادگیر j بصورت زیر کاهش خواهد یافت:

$$p_k(t+1) = (1-a)\,p_k(t) \qquad k \neq i \quad k = 1,2,...,r$$

و اگر کاردینالیتی MIS ایجاد شده در تکرار t کوچکتر از کاردینالیتی بهترین MIS ایجاد شده تاکنون باشد و از الگوریتم یادگیری $L_{R\varepsilon P}$ استفاده کرده باشیم (الگوریتم یادگیری L_{R-I} در این وضعیت عکس العملی نشان نمی دهد)، احتمال انتخاب اقدام i طبق رابطه‌ی زیر کاهش خواهد یافت:

---

[17] Isomorphic

$$p_i(t+1) = P_i(t) \times (1-b)$$

و احتمال انتخاب سایر اقدامهای اتوماتا ی یادگیر j بصورت زیر افزایش خواهد یافت:

$$p_k(t+1) = \frac{b}{1-b} + p_k(t) \times (1-b) \qquad k \neq i \quad k = 1,2,...,r$$

در رابطه‌ی بالا پارامترهای $0 < a < 1$ و $0 < b \leq a$ نرخ یادگیری[۱۸] و r تعداد اقدامهای اتوماتای یادگیر j می باشد. فرآیند ایجاد MIS تا رسیدن به شرط پایانی ادامه میبابد. آخرین MIS ایجاد شده توسط الگوریتم بعنوان بهترین MIS ، تولید شده توسط الگوریتم میباشد.ر نسخه L_{R-I} الگوریتم پیشنهادی اول در شکل ۳ دیده می شود. . این الگوریتم بصورت زیر توصیف میشود.

```
Procedure MIS
Begin
  Initialize probability vector of each automaton
  repeat
    //Phase 1
    InitialNode := Generate a random number between 1 and n (choosing the intial
    Node of MIS)
    CurrentNode := InitialNode
    Disable action 'CurrentNode' and all of its 'neighbors' in DLA's automata
    //Phase 2
    MIS := { }
    While Exist enable action
      //choose next node as a sample realization of active LA action probability
      vector, P
      NextNode := GetNextNode (CurrentNode )
      MIS := MIS + { NextNode }
      Disable action 'NextNode' and all of its 'neighbors' in DLA's automata
      CurrentNode := NextNode
    End while
    //Phase 3
    compute the cardinality of current MIS
    if Current_MIS_Cardinality < Best_MIS_Cardinality then
      //Reward selected actions of LAs in MIS according to L_{R-I} learning algorithm
      for each LA in MIS
        // <i> is the selected action of automata <j> in MIS
        p_i(t+1) = p_i(t) + a[1 - p_i(t)]
        p_k(t+1) = (1-a) p_k(t)      k ≠ i   k = 1,2,..., r
      next
    end if
    //Phase4
    enable all the disabled actions
  until (stop condition)
end procedure
```

شکل ۳: نسخه L_{R-I} الگوریتم اول

ابتدا بردار احتمال اقدامهای اتوماتاه یادگیرا در DLA مقدار دهی اولیه می شوند. احتمال انتخاب اقدامها مساوی و برابر با 1/k در نظر گرفته میشود. k تعداد اعمال اتوماتای یادگیر میباشد. در ابتدای الگوریتم کلیه اتوماتاهای یادگیر در DLA فعال میباشند. به یک گره که اتوماتای یادگیر آن غیر فعال باشد گره غیر فعال گفته میشود.

[18] Learning Rate

مرحله۱: اگر گره غیر فعال وجود ندارد در این صورت به مرحله ۳ میرویم. اگر گره فعال وجود دارد یکی از آنها بصورت تصادفی انتخاب شده و در MIS درج میگردد و اتوماتای یادگیر متناظر با این گره بعنوان اتوماتای یادگیر جاری (CurrentNode) در نظر گرفته می شود و سپس این گره (اقدام) (CurrentNode) و همهٔ گره های همسایه های آن در لیست اقدامها سایر اتوماتاهای یادگیر موجود در شبکه غیر فعال میشوند ولی از لیست اقدامها حذف نمیگردد..

مرحله ۲: یکی از اقدامهای مجاز [۱۹] اتوماتای یادگیر جاری انتخاب شده و در مجموعه MIS درج میگردد و اتوماتای یادگیر متناظر با اقدام انتخابی و کلیه اتوماتاهای همسایه این اتوماتای یادگیر غیر فعال می شوند و سپس به محله ۱ میرویم. برای انتخاب یک اقدام از تابع GetNextNode() استفاد شده است (شکل ۴) این تابع با استفاده از بردار احتمال P^j، یکی از اقدامهای مجاز اتوماتای یادگیر جاری را انتخاب می کند.

مرحله ۳: در این مرحله کاردینالیتی MIS ایجاد شده محاسبه و با کاردینالیتی بهترین MIS که تابحال بدست آمده است مقایسه می شود و در صورتیکه بزرگتر یا مساوی آن باشد، به اقدامهای انتخاب شدهی اتوماتاهای یادگیر موجود در مجموعه MIS طبق الگوریتم یادگیری L_{R-I} پاداش داده می شود.

مرحله۴: اقدامهای غیر فعال شده در حین اجرای مرحله ۲، مجددا فعال شده و سپس شرط خاتمه الگوریتم بررسی میگردد. در صورتیکه شرط پایان الگوریتم برقرار نباشد مبادرت به ایجاد یک MIS جدید مینماییم. برای این منظور ابتدا کلیه اتوماتاهای یادگیر در DLA فعال میگردند و سپس به مرحله ۱ میرویم. مقدار بردار احتمال اقدامهای اتوماتاهای یادگیر در DLA آخرین مقدار خود را دارا خواهند بود. اگر تعداد MIS های ایجاد شده توسط الگوریتم (تکرار های الگوریتم) از تعداد معینی فراتر رفته باشد و یا احتمال مجموعه[۲۰] که عبارتست از حاصلضرب احتمال انتخاب اقدامهای موجود در مجموعه MIS، از یک استانه از پیش تعیین شده (برای مثال ۰,۹) بیشتر شود، در اینصورت الگوریتم خاتمه خواهد یافت.

```
Function GetNextNode(j: Integer)
  // j is the Current Automata
  Choose a action as a sample realization of j's action probability
  vector, P^j
End function
```

شکل ۴: تابع GetNextNode برای انتخاب یکی از اقدامهای مجاز اتوماتای یادگیرجاری در الگوریتم اول

۴-۲- الگوریتم پیشنهادی دوم

در این الگوریتم برای انتخاب اقدام اتوماتای جاری، بر خلاف الگوریتم اول، که فقط از بردار احتمال اقدام استفاده می شود، از بردار احتمال اقدام و همچنین مقدار عکس درجهٔ اقدام مورد نظر (درجهٔ گره) نیز استفاده میگردد. استفاده از این مقدار برای انتخاب اقدام بعدی بهبود قابل ملاحظهای در کارایی و نرخ همگرایی الگوریتم داشته و جوابهای بهینه و یا خیلی نزدیک به جواب بهینه[۲۱] را تولید میکند. مقدار عکس درجهٔ گره i در گراف ورودی توسط $D^{-1}(i)$ نشان داده میشود.

برای استفاده از این تابع در انتخاب اقدام اتوماتای فعال، بردار احتمال اقدام اتوماتای j، P^j را بطور موقت طبق روابط زیر به بردار P'^j تغییر می دهیم و پس از انتخاب اقدام، بردار احتمال اقدام مجددا به مقدار قبلی خود P^j برگردانده می شود. این تغییرات در هر تکرار انجام میگیرد. روابط زیر چگونگی محاسبه بردار P'^j را از بردار احتمال اقدام P^j نشان می دهد

بردار احتمال اقدام اتوماتای یادگیر j : $P^j = [p_1{}^j, p_2{}^j, ..., p_r{}^j]^T$

---

[19] اقدامهاي غير فعال كه مجاز به انتخاب آنها نيستيم، اقدامهايي مي باشند كه از اتوماتاي يادگير فعال با توجه به شرايط لازم براي ايجاد مجموعه مستقل ماكزيمال، امكان انتخاب آنها وجود ندارد.

[20] Set Probability

[21] Optimal

بردار احتمال اقدام تغییر یافته اتوماتای یادگیر j : $P'^{j} = \{ p_i'^{j} \mid p_i'^{j} = \frac{[p_i{}^{j} \times D^{-1}(i)]^{\beta}}{\sum_{i=1}^{r} [p_i{}^{j} \times D^{-1}(i)]^{\beta}} \quad : \quad i = 1,2,...,r \}$

در این رابطه $P_i^{\,j}$، احتمال انتخاب اقدام i توسط اتوماتای یادگیر j می باشد و $D^{-1}(i)$، عکس درجهٔ گره i در گراف ورودی بوده و $\beta \geq 1$ اهمیت نسبی درجهٔ گره بعدی در انتخاب یک اقدام را معین میکند. بعبارتی دیگر احتمال انتخاب اقدامهایی که دارای احتمال انتخاب بیشتر و درجه کمتر باشند، بالاتر رفته و بالعکس. آزمایشها نشان داده اند که با اعمال تغییرات فوق در بردار احتمال اقدام و با توجه به در نظر گرفته شدن درجه گره‌ها، نرخ همگرایی الگوریتم پیشنهادی بمیزان قابل ملاحظه‌ای افزایش می یابد.

الگوریتم ارائه شده در این قسمت دارای دو نسخه می باشد که در نسخه اول از الگوریتم یادگیر L_{R-I} و در نسخه دوم از الگوریتم یادگیر $L_{R\varepsilon P}$ استفاده شده است. نسخه اول الگوریتم پیشنهادی در شکل ۵ نشان داده شده است.

```
Procedure MIS
Begin
  Initialize the probability vector of each automaton
  repeat
    //Phase 1
    InitialNode := Generate a random number between 1 and n (choosing the intial
    Node of MIS)
    CurrentNode := InitialNode
    Disable action 'CurrentNode' and all of its 'neighbors' in DLA's automata
    //Phase 2
    MIS := { }
    While Exist enable action
      //choose the next node as a sample realization of active LA modified action
      probability vector, P'
      NextNode := GetNextNode (CurrentNode )
      MIS := MIS + { NextNode }
      Disable action 'NextNode' and all of its 'neighbors' in DLA's automata
      CurrentNode := NextNode
    End while
    //Phase 3
    compute the cardinality of current MIS
    if Current_MIS_Cardinality < Best_MIS_Cardinality then
      //Reward selected actions of LAs in MIS according to L_{R-I} learning algorithm
      for each LA in MIS
        // <i> is the selected action of automata <j> in MIS
        p_i(t+1) = p_i(t) + a[1 - p_i(t)]
        p_k(t+1) = (1-a) p_k(t)      k ≠ i    k = 1,2,..., r
      next
    end if
    //Phase4
    enable all the disabled actions
  until (stop condition)
end procedure
```

شکل ۵: نسخه L_{R-I} الگوریتم دوم

در هر تکرار از مرحله ۲ یکی از اقدامهای مجاز اتوماتای یادگیر فعال انتخاب می شود که برای این منظور از تابع GetNextNode() استفاد شده است(شکل ۶). این تابع با استفاده از بردار احتمال P'^{j}، یکی از اقدامهای مجاز اتوماتای یادگیر فعال را انتخاب می کند.

```
Function GetNextNode(j: Integer)
  // j is the Current Automaton
```

//modify Action Probability vector of <j> (P^j) as:

$$P'^j = \{p'^j_i \mid p'^j_i = \frac{[p_i{}^j \times D^{-1}(i)]^\beta}{\sum_{i=1}^{r}[p_i{}^j \times D^{-1}(i)]^\beta} \quad : \quad i = 1,2,...,r\}$$

Choose an action as a sample realization of j's modified action probability vector, P'^j

Restore the previous value of P^j

End function

شکل ۶: تابع GetNextNode برای انتخاب یکی از اقدامهای مجاز اتوماتای فعالبرای نسخه L_{R-I} الگوریتم دوم.

۵- نتایج آزمایشات

در این قسمت نتایج حاصل از اجرای الگوریتمهای ارائه شده بر روی نمونه‌های مساله مجموعه مستقل ماکزیمال موجود در سایت DIMACS نشان داده می شود [4] [2]. نتایج گزارش شده میانگین ۱۰ مرتبه اجرای الگوریتم بوده و برای پارامترهای a و b و β بترتیب مقادیر ۰,۱ و $a/2$ و ۵ اختیار شده است. این مقادیر بصورت تجربی بدست آمده‌اند. آزمایشها با استفاده از کامپیوتر خانگی AMD 1300 MHZ با ۱۲۸ مگابایت حافظه اصلی انجام گرفته است.

الگوریتم های پیشنهادی اول و دوم با استفاده از الگوریتمهای یادگیری L_{R-I} و $L_{R\varepsilon P}$ آزمایش شده اند. در جداول ارایه شده ستونهای با نامهای مساله، رئوس و دسته ماکزیمم بترتیب دلالت بر نام گراف (نام نمونه مساله)، تعداد گره‌های آن و جواب بهینهٔ دارند. این اطلاعات از طریق سایت DIMACS قابل دسترسی می باشند. ستون بهترین جواب شامل اندازه بزرگترین مجموعه ماکزیمال پیدا شده توسط الگوریتم در ۱۰ اجرای مختلف می باشد. ستونهای جواب میانگین و زمان بترتیب شامل جواب میانگین و میانگین زمان واقعی برحسب ثانیه برای ۱۰ بار اجرای الگوریتم می باشند.

جدول ۱: جدول نتایج برای نسخه L_{R-I} الگوریتم اول.

مساله	رئوس	دسته ماکزیمم	بهترین جواب	جواب میانگین	زمان (ثانیه)
MANN_a9	۴۵	۱۶	۹	۸	۲
MANN_a27	۳۷۸	۱۲۶	۸۷	۸۵	۵,۵
Keller4	۱۷۱	۱۱	۸	۷,۵	۲
San200_0.9_1	۲۰۰	۷۰	۵۲	۴۸	۷
San200_0.9_2	۲۰۰	۶۰	۲۷	۲۳	۹
San200_0.9_3	۲۰۰	۴۴	۲۱	۱۸	۹
San400_0.9_1	۴۰۰	۱۰۰	۵۸	۵۲	۲۵

جدول ۲: جدول نتایج برای نسخه $L_{R\varepsilon P}$ الگوریتم اول

مساله	رئوس	دسته ماکزیمم	بهترین جواب	جواب میانگین	زمان (ثانیه)
MANN_a9	۴۵	۱۶	۸	۶	۱
MANN_a27	۳۷۸	۱۲۶	۸۳	۸۰	۷
Keller4	۱۷۱	۱۱	۸	۷	۱,۵
San200_0.9_1	۲۰۰	۷۰	۵۳	۴۷	۵
San200_0.9_2	۲۰۰	۶۰	۲۱	۲۰	۶
San200_0.9_3	۲۰۰	۴۴	۱۹	۱۷	۵
San400_0.9_1	۴۰۰	۱۰۰	۴۸	۴۵	۱۲

جدول ۳: جدول نتایج برای نسخه L_{R-I} الگوریتم دوم

مساله	رئوس	دسته ماکزیمم	بهترین جواب	جواب میانگین	زمان (ثانیه)
MANN_a9	۴۵	۱۶	۱۶	۱۴	۰,۰۱
MANN_a27	۳۷۸	۱۲۶	۱۲۶	۱۱۵	۳,۶
Keller4	۱۷۱	۱۱	۱۱	۱۱	۰,۴
San200_0.9_1	۲۰۰	۷۰	۶۶	۶۳	۰,۵۵
San200_0.9_2	۲۰۰	۶۰	۴۵	۴۱	۰,۵۴
San200_0.9_3	۲۰۰	۴۴	۳۳	۳۲	۰,۴۲
San400_0.9_1	۴۰۰	۱۰۰	۸۱	۶۸	۵,۵۸

جدول ۴: جدول نتایج برای نسخه $L_{R\varepsilon P}$ الگوریتم دوم

مساله	رئوس	دسته ماکزیمم	بهترین جواب	جواب میانگین	زمان (ثانیه)
MANN_a9	۴۵	۱۶	۱۴	۱۳	۰,۰۱
MANN_a27	۳۷۸	۱۲۶	۱۱۶	۱۱۰	۳
Keller4	۱۷۱	۱۱	۱۱	۸	۰,۲
San200_0.9_1	۲۰۰	۷۰	۶۱	۵۸	۱
San200_0.9_2	۲۰۰	۶۰	۴۴	۳۷	۰,۳۸
San200_0.9_3	۲۰۰	۴۴	۳۱	۳۰	۰,۳۳
San400_0.9_1	۴۰۰	۱۰۰	۷۵	۶۱	۶,۵

با بررسی و مقایسه نتایج ازمایشها که در جدولهای ۱ و ۲ امده است میتوان نتیجه گرفت که کارایی و نرخ همگرایی نسخه L_{R-I} الگوریتم اول از نسخه $L_{R\varepsilon P}$ این الگوریتم بیشتر می باشد و مؤید این نکته است که استفاده از الگوریتم یادگیری L_{R-I} برای الگوریتم

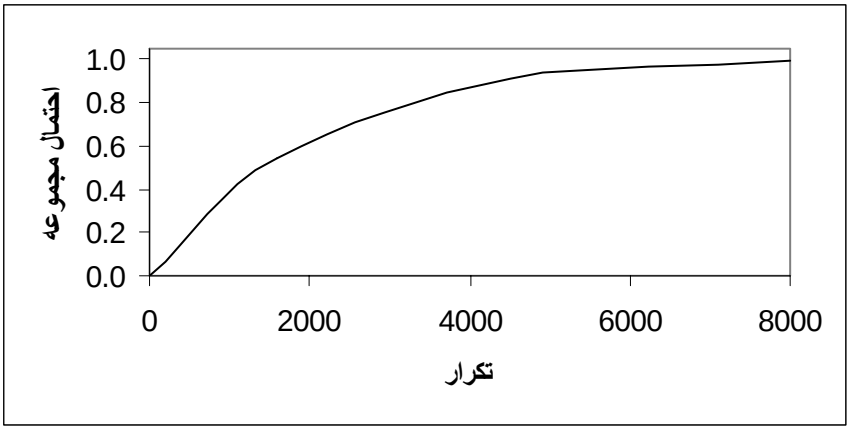
اول مناسبتر است و جوابهای بهتری تولید می کند. البته با مقایسه جوابهای تولید شده توسط هر دو نسخه الگوریتم اول و جوابهای بهینه، ضعف الگوریتم اول مشاهده می شود که در قسمت قبل به آن اشاره کردیم. در جدولهای ۳ و ۴ نتایج حاصل از اجرای دو نسخه L_{R-I} و $L_{R\varepsilon P}$ الگوریتم دوم آمده است و همانطور که دیده میشود نسخه L_{R-I} این الگوریتم در مقایسه با نسخه $L_{R\varepsilon P}$ این الگوریتم بهتر عمل می کند و نتایجی که تولید میکند به جواب بهینه نزدیکتر است. آزمایشها نشان داده است که نسخه L_{R-I} الگوریتم دوم از نرخ همگرایی بالاتری نسبت به سه الگوریتم پیشنهادی دیگر برخوردار است[10]. در جدول ۵ نسخه L_{R-I} الگوریتم دوم با الگوریتمهای ARH ,[23]CMPF,[22]و PCAOP[24] مقایسه شده است.

جدول ۵: مقایسه نتایج بدست آمده از نسخه L_{R-I} الگوریتم دوم با برخی از الگوریتمهای موجود.

مساله	رئوس	دسته ماکزیمم	الگوریتم دوم	CMPF	ARH	PCAOP
MANN_a9	۴۵	۱۶	۱۶	۱۶	۱۶	۱۶
MANN_a27	۳۷۸	۱۲۶	۱۲۶	۱۲۵	۱۱۷	۱۲۶
Keller4	۱۷۱	۱۱	۱۱	۱۱	۸	۱۱
San200_0.9_1	۲۰۰	۷۰	۶۶	۴۷	۴۵	۷۰
San200_0.9_2	۲۰۰	۶۰	۴۵	۴۰	۳۹	۴۲
San200_0.9_3	۲۰۰	۴۴	۳۳	۳۴	۳۱	۳۲
San400_0.9_1	۴۰۰	۱۰۰	۸۱	۷۵	۵۰	۸۲

با توجه به نتایج بدست آمده میتوان گفت الگوریتم پیشنهادی از کارایی خوبی برخوردار بوده و در مقایسه با دو الگوریتم CMPF و ARH غالبا جوابهای بهتری تولید می کند و نظر به اینکه الگوریتم پیشنهادی در مقایسه با الگوریتم PCAOP از سرعت نسبتا بیشتری برخوردار است، به این الگوریتم نیز ترجیح داده می شود. طبق جدول ۱ زمان لازم برای رسیدن به جواب مساله Keller4 برابر ۰,۴ ثانیه می باشد که در مقایسه با زمان ۰,۹ ثانیه برای حل این مساله توسط الگوریتم PCAOP، تقریبا ۵۰ درصد کمتر می باشد.

در یکی از بخشهای این مقاله به احتمال مجموعه که عبارتست از حاصلضرب احتمال انتخاب اقدامهای موجود در مجموعه مستقل ماکزیمال اشاره شد. الگوریتم در صورتی به جواب بهینه همگرا خواهد شد که حاصلضرب احتمال انتخاب اقدامهای موجود در مجموعه مجموعه مستقل ماکزیمال با افزایش تکرارهای الگوریتم به مقدار ۱ همگرا شود. نمودار احتمال مجموعه برای یک اجرای نمونه نسخه L_{R-I} الگوریتم دوم در شکل ۸ نشان داده شده است.



[22] Continuous Multivariable Polynomial Formulations
[23] Annealed Replication Huristic
[24] Probabilistic Constructive Approach To Optimization Problems

شکل ۸: نمودار احتمال مجموعه

۶- نتیجه گیری

در این مقاله الگوریتمهای جدیدی مبتنی بر اتوماتاهای یادگیر توزیع شده برای حل مساله مجموعه مستقل ماکزیمال معرفی و با تعدادی از الگوریتمهای گزارش شده مقایسه گردید. نتایج آزمایشهای انجام گرفته برتری الگوریتمهای پیشنهادی بر الگوریتمهای موجود برای حل مساله مجموعه مستقل ماکزیمال نشان داد.

مراجع

[1] J. Abello, "Finding independent sets in a graph using continuous multivariable polynomial formulations", J. Global Optim. 21 (2001) 111-137 05C69 (05C85 90C35).

[2] Jennifer L. Wong , Farinaz Koushanfar , Seapahn Meguerdichian , Miodrag Potkonjak, "A probabilistic constructive approach to optimization problems", Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design, November 04-08, 2001, San Jose, California.

[3] http://www2.toki.or.id/book/AlgDesignManual/BOOK/BOOK4/NODE173.HTM, 1997.

[4] http://dimacs.rutgers.edu/Challenges.

[5] A. Ferreira and N. Schabanel, "A randomized BSP/CGM algorithm for the maximal independent set", In Proceedings of ISPAN'99, Fremantle, Australia, pages 284--289, June 1999. IEEE Press.

[6] N. Alon, L. Babai, and A. Itai. "A fast and simple randomized parallel algorithms for maximal independent set problem", J. of algorithms, 7:567-583,1986.

[7] Andrew V. Goldberg, Serge A. Plotkin, and Gregory E. Shannon. "Parallel symmetry - breaking in sparse graphs", In 19^{th} STOC, volume 19,pages 315-324, May 1987.

[8] R.M. Karp and A. Widgerson. "A fast parallel algorithm for the maximal independent set problem", JACM, 32(4):762-773, Oct 1985.

[9] M. Luby. "A simple parallel algorithm for maximal independent set problem", SIAM J. computer., 15(4):1036-1053,Nov 1986.

[10] M. Alipour and M. R. Meybodi, " Solving Maximal Independent Set Problem using Distributed Learning Automata", Computer Engineering Department Technical Report, Amirkabir University, Tehran, Iran, Oct. 2004.