دهمین
کنفرانس سالانه انجمن کامپیوتر ایران
۲۷ تا ۲۹ بهمن ۱۳۸۳

حل مسأله بزرگترین مجموعه مستقل توسط اتوماتای یادگیر سلولی

سید علیرضا متولیان محمدرضا میبدی

آزمایشگاه محاسبات نرم
دانشکده مهندسی کامپیوتر و فناوری اطلاعات
دانشگاه صنعتی امیرکبیر، تهران، ایران

چکیده

مسأله بزرگترین مجموعه مستقل در یک گراف، عبارتست از یافتن بزرگترین زیرمجموعه‌ای از یک گراف بطوریکه هیچ دو رأسی از این مجموعه با یالی بهم متصل نباشند. این مسأله از جمله مسایل *NP_Complete* می‌باشد و بهمین دلیل الگوریتمهای مکاشفه‌ای و تقریبی متعددی از جمله تابکاری فلزات، شبکه‌های عصبی و الگوریتمهای ژنتیکی تاکنون برای آن ارائه شده‌اند. اتوماتای یادگیر سلولی یک ابزار جستجوی تصادفی است که می‌توان از آن در حل مسایل *NP_Complete* استفاده نمود. در این مقاله یک الگوریتم مبتنی بر اتوماتای یادگیر سلولی برای حل مسأله بزرگترین مجموعه مستقل ارائه شده و کارایی آن برروی تعدادی گراف نمونه استاندارد آزمایش شده است.

کلمات کلیدی: بزرگترین مجموعه مستقل، اتوماتای یادگیر، اتوماتای یادگیر سلولی.

۱- مقدمه

مسأله *بزرگترین مجموعه مستقل*[1]، عبارتست از یافتن بزرگترین زیرمجموعه از رأسهای یک گراف بطوریکه هیچ دو رأسی از این مجموعه با یالی بهم متصل نباشند(به عبارت دیگر این رئوس از هم مستقل‌اند). به عبارت دقیق‌تر، مسأله بزگترین مجموعه مستقل برای گراف $G=(V,E)$ عبارتست از یافتن مجموعه $S \subseteq V$ از رئوس گراف با شرط $|S| = \max\{|S'| \mid S' \subseteq V \wedge \forall u,v \in S' : \langle u,v \rangle \notin E\}$.

[1] Maximum Independent Set

این مسأله از کاربردهای فراوانی برخوردار است که اهم آنها عبارتند از انتخاب کد و تصحیح خطا در تئوری کدینگ، حل بن بست در مباحث شبکه، تشخیص خطا در سیستم‌های چندپردازنده‌ای و بینایی ماشین و شناسایی الگو[1]. این مسأله از اولین مسایلی است که **NP_Complete** بودن آن ثابت گردید[2]. در سال ۱۹۹۱، **Feige** و همکارانش[3] نشان دادند که حتی تخمین این مسأله با یک ضریب ثابت نیز **NP_Hard** می‌باشد.

با توجه به عدم موفقیت رهیافتهای قطعی در حل ابعاد بزرگ مسأله، توجه زیادی معطوف حل آن با استفاده از رهیافتهای *مکاشفه‌ای*[2] گردید. **Aart** و **Korst** [4] و **Homer** و **Peinado** [5] تابکاری فلزات را برای حل مسأله بزرگترین مجموعه مستقل بکار بردند و کارایی بالای این تکنیک را به اثبات رساندند. در سال ۱۹۹۳، **Carter** و **Park** [6] با ارائه یک الگوریتم ژنتیکی بدین نتیجه رسیدند که این تکنیک از نظر بار پردازش بسیار سنگین بوده و در حالت کلی با روشهای مکاشفه‌ای قدیمی‌تر قابل مقایسه نیستند. اما **Back** و **Khuri** [7] برخلاف آنها به نتیجه‌ای متضاد رسیدند. الگوریتم ژنتیکی آنها که از تابع جریمه برای نتایج غیر ممکن استفاده می‌کرد از کارایی خوبی برخوردار بود. برای مطالعه جامع انواع روشهای بکار رفته در حل این مسأله به [1] مراجعه نمایید.

دراین مقاله یک الگوریتم جدید مبتنی بر اتوماتای یادگیر سلولی برای حل مسأله بزرگترین مجموعه مستقل ارائه می‌شود. اعمال این الگوریتم بر گرافهای **DIMACS** [8]، نشان می‌دهد که الگوریتم پیشنهادی از کارایی خوبی برخودار بوده وقادر به یافتن جوابهای بهینه یا بسیار نزدیک به بهینه مسأله می‌باشد. علاوه براین، الگوریتم پیشنهادی با الگوریتم ارائه شده توسط **Back** و **Khuri** [7] نیز مقایسه گردیده‌است. نتایج این مقایسه نشان دهنده کارایی، نوسان اندک کیفیت پاسخ در دفعات مختلف اجرا و به خصوص کیفیت خوب پاسخهای الگوریتم پیشنهادی می‌باشد.

ادامه مقاله بصورت زیر سازماندهی شده است. در بخش‌های ۲ و ۳ اتوماتای یادگیر و اتوماتای یادگیر سلولی بطور خلاصه معرفی می‌گردد. سپس در بخشهای ۴ و ۵، الگوریتم پیشنهادی و نتایج شبیه‌سازیها گزارش می‌شوند. در انتها نیز نتیجه‌گیری و مراجع مورد استفاده، ذکر شده‌اند.

۲- اتوماتای یادگیر

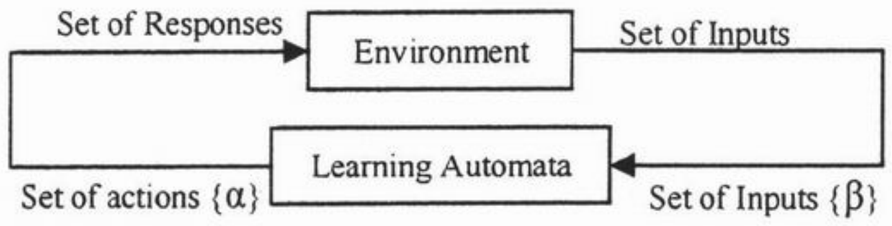
اتوماتای یادگیر یک مدل انتزاعی است که تعداد محدودی عمل را می‌تواند انجام دهد. هر عمل انتخاب شده توسط محیطی احتمالی ارزیابی می‌گردد و پاسخی به اتوماتای یادگیر داده می‌شود. اتوماتای یادگیر از این پاسخ استفاده نموده و عمل خود را برای مرحله بعد انتخاب می‌کند. اتوماتاهای یادگی به دو دسته عمده اتوماتای یادگیر با ساختار ثابت و اتوماتای یادگیر با ساختار متغیر تقسیم می‌شوند.

۲-۱- محیط

محیط را می‌توان توسط سه‌تایی $E \equiv \{\alpha, \beta, c\}$ تعریف نمود که در آن $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه ورودیها، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_r\}$ مجموعه خروجیها و $c \equiv \{c_1, c_2, ..., c_r\}$ مجموعه احتمالهای جریمه شدن می‌باشد. هرگاه β_i دو مقداری

[2] Heuristic

باشد، محیط از نوع **P** می‌باشد. در چنین محیطی $\beta_i = 1$ به عنوان جریمه و $\beta_i = 0$ به عنوان پاداش در نظر گرفته می‌شود. در محیط از نوع **Q**، $\beta_i(n)$ می‌تواند بطور گسسته یکی از مقادیر محدود در فاصله [0,1] را اختیار کند و در محیط از نوع **S**، $\beta_i(n)$ متغیر تصادفی در فاصله [0,1] است، یعنی $\beta_i(n) \in [0,1]$.



شکل ۱) ارتباط بین اتوماتای یادگیر و محیط

c_i احتمال اینکه عمل α_i نتیجه نامطلوب داشته باشد می‌باشد. در یک محیط پایدار مقادیر c_i بدون تغییر باقی می‌ماند. حال آنکه در یک محیط ناپایدار این مقادیر در طی زمان تغییر می‌کنند. شکل ۱ ارتباط بین اتوماتای یادگیر و محیط را نشان می‌دهد. برای بحث کامل در مورد انواع اتوماتاهای یادگیر به **[9]** رجوع کنید.

۲-۲- اتوماتای یادگیر با ساختار متغیر

اتوماتای یادگیر با ساختار متغیر توسط ۵ تایی $LA \equiv \{\alpha, \beta, p, T, c\}$ که در آن $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه عمل‌های اتوماتا، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_r\}$ مجموعه ورودیهای اتوماتا، $p \equiv \{p_1, p_2, ..., p_r\}$ بردار احتمال انتخاب عمل، $T \equiv p(n+1) = T[\alpha(n), \beta(n), p(n)]$ الگوریتم یادگیری و $c \equiv \{c_1, c_2, ..., c_r\}$ مجموعه احتمالهای جریمه‌شدن می‌باشند. در این نوع از اتوماتاها، اگر عمل α_i در مرحله nام انتخاب شود و پاسخ مطلوب از محیط دریافت نماید، احتمال $p_i(n)$ افزایش و سایر احتمالها کاهش می‌یابند. برای پاسخ نامطلوب $p_i(n)$ کاهش و سایر احتمالها افزایش می‌یابند. در هر حال، تغییرات بگونه‌ای صورت می‌پذیرد که حاصل جمع $p_i(n)$ها همواره ثابت و مساوی یک باقی بماند. الگوریتم زیر یک نمونه از الگوریتمهای یادگیری خطی می‌باشد:

الف- پاسخ مطلوب: (۱)

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

ب- پاسخ نامطلوب: (۲)

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

در روابط فوق، a پارامتر پاداش و b پارامتر جریمه می‌باشد. با توجه به مقادیر a و b سه حالت را می‌توان در نظر گرفت. اگر a و b با هم برابر باشند، الگوریتم را L_{RP}، چنانچه b از a خیلی کوچکتر باشد، الگوریتم را $L_{R\varepsilon P}$ و اگر b صفر باشد، الگوریتم را L_{RI} می‌گویند. برای مطالعه بیشتر در مورد این اتوماتاها به **[9]** رجوع کنید.

۳- اتوماتای یادگیر سلولی[3] (CLA)

[3] Cellular Learning Automata

اتوماتای یادگیر سلولی، مدلی ریاضی برای سیستم‌هایی است که از اجزاء ساده‌ای تشکیل شده‌اند و رفتار هر جزء براساس رفتار همسایگانش و نیز تجربیات گذشته‌اش تعیین و اصلاح می‌شود. اجزاء ساده تشکیل دهنده این مدل، از طریق تعامل با یکدیگر می‌توانند رفتار پیچیده‌ای را از خود نشان دهند. هر اتوماتای یادگیر سلولی، از یک اتوماتای سلولی تشکیل شده است که هر سلول در آن به یک اتوماتای یادگیر مجهز می‌باشد که حالت فعلی این سلول را مشخص می‌کند. مانند اتوماتای سلولی، قانونی محلی در محیط حاکم است که تعیین می‌کند که آیا عمل انتخاب شده توسط یک اتوماتا در یک سلول بایستی پاداش داده شود و یا جریمه گردد. عمل جریمه کردن و یا پاداش دادن منجر به بروز درآوردن ساختار CLA به منظور نیل به یک هدف مشخص می‌شود. این مدل برای اولین بار در **[10]** معرفی گردید. این مدل جدید دارای کاربردهای گوناگونی نظیر حل مسایل مشکل، پردازش تصاویر و ... می‌باشد**[11]**. بیگی و میبدی در **[12]** این مدل را بطور مفصل مورد مطالعه قرار داده و برخی قضایای مهم درباب همگرایی آن به پاسخ بهینه را اثبات نموده‌اند.

یک CLA بصورت پنج‌تایی $\langle \Delta, A, \Omega, R, L \rangle$ نشان داده می‌شود. $\Delta = \{\lambda_1, \lambda_2, ..., \lambda_n\}$ مجموعه سلولهای موجود در CLA می‌باشد که در یک شبکه کارتزین قرار گرفته‌اند. $A = \{a_1, a_2, ..., a_k\}$ مجموعه اعمال مجاز یک سلول است. $A^t(\lambda_i)$ عمل انجام گرفته در سلول λ_i در زمان t را نشان می‌دهد و R قانون حاکم بر سیستم می‌باشد. $\Omega = \{\Omega_1, \Omega_2, ..., \Omega_m\}$ مجموعه سلولهای همسایه یک سلول در CLA بوده و L اتوماتای یادگیری است که هر سلول به آن مجهز است. $\Omega(\lambda_i)$ سلولهای همسایه سلول λ_i بوده که دارای دو خصوصیت زیر می‌باشد:

$$1)\ \lambda_i \notin \Omega(\lambda_i) \qquad \forall \lambda_i \in \Delta$$
$$2)\ \lambda_i \in \Omega(\lambda_j) \qquad iff\ \ \lambda_j \in \Omega(\lambda_i) \qquad \forall \lambda_i, \lambda_j \in \Delta \tag{۳}$$

فرض می‌کنیم که $W(\lambda_i) = \Omega(\lambda_i) \cup \{\lambda_i\}$ باشد. در این صورت، قانون حاکم بر سیستم می‌تواند بصورت تابعی به شکل $A^{t+1}(\lambda_i) = R\{A^t(x) \mid x \in W(\lambda_i)\}$ تعریف گردد.

عملکرد اتوماتای یادگیر را می‌توان بصورت زیرشرح داد: در ابتدا هر اتوماتای یادگیر در CLA یکی از اعمال خود را انتخاب می‌کند. این عمل می‌تواند براساس مشاهدات قبلی و یا بصورت تصادفی انتخاب شود. عمل انتخاب شده با توجه به اعمال انتخاب شده توسط سلولهای همسایه و قانون حاکم بر CLA جریمه و یا پاداش داده می‌شود. با توجه به اینکه عمل انتخاب شده پاداش گرفته و یا جریمه شده است، ساختار داخلی اتوماتا بروز می‌شود. عمل بروزرسانی تماما اتوماتاها در CLA همزمان صورت می‌گیرد. بعد از بروزرسانی، هر اتوماتا در CLA دوباره یک عمل از مجموعه عملهای خود را انتخاب و انجام می‌دهد. نتیجه عمل منجر به پاداش و یا جریمه آن عمل می‌گردد. فرآیند انتخاب عمل و دادن پاداش یا جریمه تا زمانیکه سیستم به یک حالت پایدار برسد و یا یک معیار از قبل تعریف شده‌ای برقرار گردد، ادامه می‌یابد. عمل بروزرسانی اتوماتاهای موجود در سلول های CLA توسط الگوریتم یادگیری انجام می‌گیرد.

۴- راه‌حل پیشنهادی

الگوریتم پیشنهادی سعی می‌کند یک مجموعه مستقل را با اندازه از پیش تعیین شده m، در گراف مسأله پیدا نماید. به همین علت در این الگوریتم **CLA** دارای m اتوماتای یادگیر است که هریک n(تعداد رئوس گراف) عمل دارند. هریک از این عملها متناظر است با یکی از نودهای گراف و انتخاب عمل iام توسط یک اتوماتای یادگیر، معادل است با درج نود متناظر با آن عمل، در

مجموعه مستقل. اتوماتاهای یادگیر همگی از نوع L_{RI} می‌باشند. همسایگی اتوماتاهای یادگیر در **CLA** براساس همسایگی گره های گراف تعیین می‌شود. دو اتوماتای یادگیر در **CLA** بایکدیگر همسایه‌اند اگر و فقط اگر میان گره های متناظر با عملهای انتخابی آنها در گراف، یالی وجود داشته باشد.

در طول اجرای برنامه هر اتوماتای یادگیر متناسب با بردار احتمال انتخاب عمل خود، یکی از عملهایش(نودهای گراف) را انتخاب می‌نماید. سپس براساس قوانین تعریف شده برای **CLA** پاداش یا جریمه آن تعیین شده و اتوماتای یادگیر، بردار احتمال انتخاب عمل خود را متناسباً بروز می‌نماید. الگوریتم هنگامی خاتمه می‌یابد که همه اتوماتاهای یادگیر پاداش بگیرند(بدین معنی که به جواب مطلوب رسیده‌ایم)، و یا تعداد تکرارهای الگوریتم از یک حدآستانه فراتر رود که در این صورت الگوریتم نتوانسته جوابی برای مسأله بیابد.

فرض کنید مجموعه A، مجموعه اتوماتاهای یادگیر **CLA**، $m=|A|$ تعداد آنها و $n=|V|$ تعداد عملهای آنها می‌باشد. اگر مجموعه عملهای هر اتوماتای یادگیر را بصورت $\alpha=\{\alpha^1,\alpha^2,...,\alpha^n\}$ و مجموعه پاداشهای محیط را بصورت دو حالتهٔ $\beta=\{0,1\}$ در نظر بگیریم(1 به معنی جریمه و 0 به معنی پاداش)، آنگاه قانون بروزرسانی **CLA** بصورت زیر خواهد بود:

$$\forall i\in A\ ,\ \beta_i(t)=\begin{cases}1 & if\ \ \forall j\in A, j\neq i\ :\ \left(\alpha_i(t)=\alpha_j(t)\right)\vee\left(\langle\alpha_i(t),\alpha_j(t)\rangle\in E\right)\\ 0 & Otherwise\end{cases} \qquad (۴)$$

که $\beta_i(t)$ امتیاز(پاداش یا جریمه) و $\alpha_i(t)$ عمل انتخاب شده توسط اتوماتای یادگیرiام در زمان t می‌باشند.

با توجه به توضیحات فوق شبه کد الگوریتم بدین شرح است:

```
Initialize a CLA with m LAís each has n actions
For each LA i in CLA Do In Parallel
  For j = 1 To n Do
    p_ij = 1/n
  End For
  //α_i is the current action of LA i
  Select action α_i according to the Action Probability Vector P_i
End For
While(At Least some LA is penalized and MaxSteps has not been exceeded) Do
  For each LA i in CLA Do In Parallel
    If GetResponse(i) = Reward Then
      p_iα_i(t+1) = p_iα_i(t) + a[1 − p_iα_i(t)]
      For j = 1 To n, j ≠ α_i Do
        p_ij(t+1) = (1 − a) p_ij(t)
      End For
    End If
  End For
  For each LA i in CLA Do In Parallel
    Select action α_i according to the Action Probability Vector P_i
  End For
End While
If (MaxSteps has not been Exceeded) Then
  MIS={α_i | α_i is the current action of LA i}
End If
```

برای بدست آوردن بزرگترین مجموعه مستقل باید الگوریتم را برای مقادی مختلف m اجرا نماییم. برای این کار می‌توان از یک روش نصف کردن فاصله استفاده نمود. ابتدا $m=\frac{n}{2}$ را فرض کرده و الگوریتم **CLA** را روی آن اجرا می کنیم. اگر برای این مقدار m مجموعه مستقلی بدست آمد، $m=\frac{n+n/2}{2}$ و اگر بدست نیامد $m=\frac{1+n/2}{2}$ قرار داده و **CLA** را اجرا می کنیم. عمل

نصف کردن فاصله و تغییر حدود بالا و پایین جواب را آنقدر تکرار می‌کنیم تا فاصله به صفر برسد. در این صورت مجموعه مستقل متناظر با بیشترین مقدار m جواب مسأله خواهد بود.

۵- نتایج پیاده‌سازی

برای بررسی کارایی الگوریتم پیشنهادی، آن را بر تعدادی گراف نمونه از مجموعه گرافهای **DIMACS** [7] اعمال نمودیم. تعداد گرههای این گرافها توانی از عدد ۲ بوده و در انتهای نام آنها (بعد از علامت نقطه) ذکر شده‌اند. جداول ۱ و ۲ نتایج اجرای الگوریتم را برروی گرافهای فوق الذکر نشان می‌دهد.

جدول ۱) نتایج اجرای الگوریتم برای گرافهای دسته‌های **1dc** و **1et**

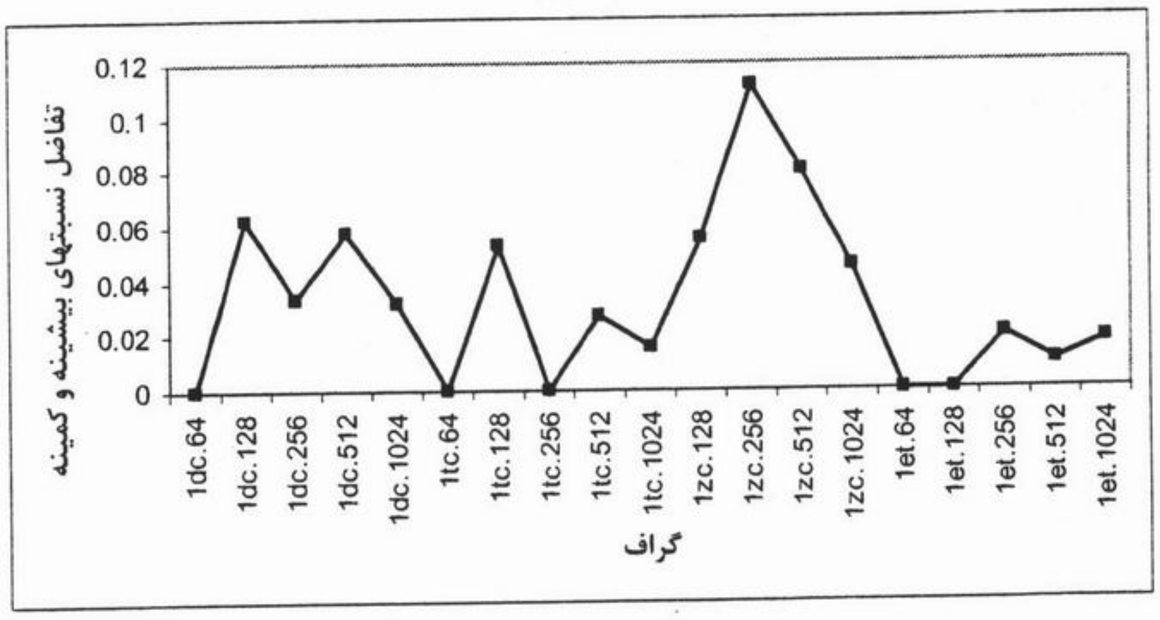
گراف	1dc 64	1dc 128	1dc 256	1dc 512	1dc 1024	1et 64	1et 128	1et 256	1et 512	1et 1024
بیشترین مقدار ممکن	۱۰	۱۶	۳۰	۵۲	۹۴	**۱۸**	**۲۸**	**۵۰**	**۱۰۰**	**۱۷۱**
میانگین جواب	۱۰	۱۴/۷	۲۶/۲	۳۸/۸	۷۱	۱۸	۲۸	۴۹/۶	۹۴/۴	۱۵۵/۶
بیشترین جواب	۱۰	۱۵	۲۷	۴۱	۷۲	۱۸	۲۸	۵۰	۹۵	۱۵۷
کمترین جواب	۱۰	۱۴	۲۶	۳۸	۶۹	۱۸	۲۸	۴۹	۹۴	۱۵۴
نرخ تغییرات	۰	۰/۰۱۵۹	۰/۰۰۶۸	۰/۰۳۳۲	۰/۰۱۵۶	۰	۰	۰/۰۰۵۴	۰/۰۰۲۸	۰/۰۰۶۰
نسبت کارایی بیشینه	۱	۰/۹۳۷۵	۰/۹	۰/۷۸۸۵	۰/۷۶۶۰	۱	۱	۱	۰/۹۵	۰/۹۱۸۱
نسبت کارایی کمینه	۱	۰/۸۷۵	۰/۸۶۶۷	۰/۷۳۰۸	۰/۷۳۴۰	۱	۱	۰/۹۸	۰/۹۴	۰/۹۰۰۶
تفاضل نسبتها	۰	۰/۰۶۲۵	۰/۰۳۳۳	۰/۰۵۷۷	۰/۰۳۱۹	۰	۰	۰/۰۲	۰/۰۱	۰/۰۱۷۵

جدول ۱) نتایج اجرای الگوریتم برای گرافهای دسته‌های **1tc** و **1zc**

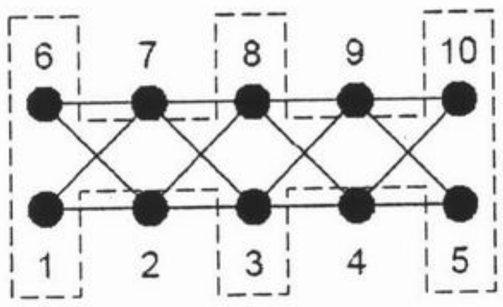
گراف	1tc 64	1tc 128	1tc 256	1tc 512	1tc 1024	1zc 128	1zc 256	1zc 512	1zc 1024
بیشترین مقدار ممکن	**۲۰**	**۳۸**	**۶۳**	**۱۱۰**	**۱۹۶**	**۱۸**	**۳۶**	**۶۲**	**۱۱۰**
میانگین جواب	۲۰	۳۴/۷	۶۰	۱۰۷/۵	۱۸۱/۸	۱۶/۳	۲۸/۹	۵۰/۱	۸۸/۲
بیشترین جواب	۲۰	۳۶	۶۰	۱۰۸	۱۸۳	۱۷	۳۱	۵۳	۹۱
کمترین جواب	۲۰	۳۴	۶۰	۱۰۶	۱۸۰	۱۶	۲۷	۴۸	۸۶
نرخ تغییرات	۰	۰/۰۱۳۱	۰	۰/۰۰۸۸	۰/۰۰۷۱	۰/۰۱۴۳	۰/۰۴۹۶	۰/۰۳۷۵	۰/۰۲۲۲
نسبت کارایی بیشینه	۱	۰/۹۴۷۴	۰/۹۵۲۴	۰/۹۹۰۹	۰/۹۳۳۷	۰/۹۴۴۴	۰/۸۱۱	۰/۸۵۴۸	۰/۸۲۷۳
نسبت کارایی کمینه	۱	۰/۸۹۴۷	۰/۹۵۲۴	۰/۹۶۳۶	۰/۹۱۸۴	۰/۸۸۸۹	۰/۷۵	۰/۷۷۴۲	۰/۷۸۱۸
تفاضل نسبتها	۰	۰/۰۵۲۶	۰	۰/۰۲۷۲	۰/۰۱۵۳	۰/۰۵۵۶	۰/۱۱۱۱	۰/۰۸۰۶	۰/۰۴۵۵

در جداول فوق نسبت کارایی بیشینه عبارت است از نسبت بهترین جواب بدست آمده توسط الگوریتم به بهترین جواب ممکن برای آن گراف. بهمین ترتیب، نسبت کارایی کمینه عبارت است از نسبت بدترین جواب بدست آمده، به بهترین جواب. با توجه به داده‌های جدول، میانگین نسبت کارایی بیشینه ۹۲/۵٪ و میانگین نسبت کارایی کمینه ۸۹/۲٪ می‌باشند که نشان دهنده کارایی بالای الگوریتم در رسیدن به جوابهای نزدیک بهینه می‌باشند. علاوه براین، نرخ تغییرات که عبارت است از نسبت واریانس به میانگین نیز بسیار اندک بوده و بطور میانگین ۱/۲٪ می‌باشد. این مقادیر علاوه بر کارایی بالای الگوریتم، نشان دهنده یک ویژگی مهم دیگر الگوریتم نیز می‌باشند و آن تغییرات اندک پاسخ الگوریتم در اجراهای مختلف برنامه می‌باشد. برای مشاهده بهتر، نمودار ۱ فاصله نسبی میان بهترین جواب و بدترین جواب بدست آمده برای گرافهای گوناگون را نشان می‌دهد. همانطور که از نمودار پیداست متوسط تغییرات نسبت، ۳/۲۷٪ می‌باشد که مقدار بسیار اندکی است. با توجه به این مشاهدات می‌توانیم نتیجه بگیریم که الگوریتم پیشنهادی دارای قابلیت اطمینان بالایی بوده و همیشه به جوابهایی در حدود ۹۰٪(میانگین کارایی) جواب بهینه می‌رسد.

برای بررسی بیشتر کارکرد الگوریتم پیشنهادی، این الگوریتم را برروی گرافهای نمونه مطرح شده در [7] نیز آزمایش می‌نماییم. گراف معیار در [7] یک گراف قابل تعمیم به اندازه‌های بزرگتر و با تعداد گرههای زوج می‌باشد. مثلاً به ازاء $n = 10$ گراف شکل ۲ را خواهیم داشت:



نمودار ۱) تفاضل نسبت کارایی بیشینه و کمینه برای گرافهای مختلف



شکل ۲) گراف منظم مطرح در [7] برای n=10

همان‌گونه که از ساختار گراف پیداست، اگر n(تعداد گره های گراف) بر ۴ بخش‌پذیر نباشد، بزرگترین مجموعه مستقل آن $\frac{n}{2}+1$ عضو داشته و اگر n بر ۴ بخش‌پذیر باشد، $\frac{n}{2}$ عضو خواد داشت.

جدول ۳ مقایسه‌ایست میان نتایج بدست آمده در [7] و الگوریتم پیشنهادی مبتنی بر اتوماتای یادگیر سلولی برای گرافهایی از نوع فوق و با اندازه‌های ۱۰۲(**misp102**) و ۲۰۲(**misp202**). همانگونه که از جدول ۵ مشخص است، میانگین جوابهای بدست آمده توسط الگوریتم مبتنی بر **CLA** از مقادیر بدست آمده توسط الگوریتم ژنتیکی بهتر می‌باشد و به نظر می‌رسد این برتری با افزایش تعداد گره های گراف، فزونی می‌یابد. علاوه براین، تفاضل نسبت‌ها برای پاسخهای بدست آمده در الگوریتم **CLA** بسیار کوچکتر بوده و برای **misp102** کمتر از ۴۰٪ و برای **misp202** کمتر از ۲۲٪ تفاضل نسبت‌ها الگوریتم ژنتیکی است. با مقایسه نرخ تغییرات نیز درمی‌یابیم که نرخ تغییرات بدست آمده در الگوریتم **CLA** بسیار کوچکتر بوده و برای **misp102** کمتر از ۴۰٪ و برای **misp202** کمتر از ۱۷٪ نرخ تغییرات در الگوریتم ژنتیکی است. در نهایت مشاهده می‌شود که بهترین پاسخهای **CLA** با بهترین پاسخهای الگوریتم ژنتیکی برابر است.

جدول ۳) مقایسه الگوریتم مبتنی بر CLA با الگوریتم ژنتیکی Back

	الگوریتم پیشنهادی		الگوریتم Back	
گراف	Misp102	Misp202	Misp102	Misp202
بیشترین مقدار ممکن	۵۲	۱۰۲	۵۲	۱۰۲
میانگین جواب بدست آمده	۴۷/۹	۹۴/۵	۴۴/۹۴	۸۸/۹
بهترین جواب بدست آمده	۵۰	۹۶	۵۰	۹۶
بدترین جواب بدست آمده	۴۶	۹۳	۴۰	۸۲
نسبت کارایی بیشینه	۰/۹۶۱۵	۰/۹۴۱۲	۰/۹۶۱۵	۰/۹۴۱۲
نسبت کارایی کمینه	۰/۸۸۴۶	۰/۹۱۱۸	۰/۷۶۹۲	۰/۸۰۴۰
تفاضل نسبتها	۰/۰۷۶۹	۰/۰۲۹۴	۰/۱۹۲۳	۰/۱۳۷۲
نرخ تغییرات	۰/۰۳۴۶	۰/۰۱۴۷	۰/۰۸۷۱	۰/۰۸۶۷

۶- نتیجه‌گیری

در این مقاله یک الگوریتم جدید مبتنی بر اتوماتای یادگیر سلولی برای حل مسأله بزرگترین مجموعه مستقل ارائه گردید. نتایج شبیه‌سازیها و آزمایش الگوریتم بر روی گرافهای نمونه **DIMACS** [8]و نیز مقایسه آن با الگوریتم ارائه شده در با الگوریتم ارائه شده توسط **Back** و **Khuri** [7]نشان می‌دهند که الگوریتم پیشنهادی از کارایی و قابلیت اطمینان بالایی برخوردار است.

۷- مراجع

[1] I. M. Bomze, M. Budinich, P. M. Pardalos, M. Pelillo; "The Maximum Clique Problem"; *Handbook of Combinatorial Optimization* (*Supplement Volume A*), Kluwer Academic Publishers, Boston, MA, 1999.

[2] R. M. Karp; "Reducibility among Combinatorial Problems"; *Complexity of Computer Computations*, p.p. 85-103, Plenum Press, New York, 1972.

[3] U. Feige, S. Goldwasser, S. Safra L. Lov· sz, M. Szegedy; "Approximating clique is almost NP-complete"; *In Proceeding of the 32nd Annual IEEE Symposium on the Foundations of Computer Science*, pages 2-12, 1991.

[4] E. Aart, J. Korst; "Simulated annealing and Boltzmann Machines"; *J. Wiley & Sons, Chichester*, UK, 1989.

[5] S. Homer, M. Peinado; "Experiments with polynomial-time CLIQUE approximation algorithms on very large graphs"; *Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge*; Vol. DIMACS 26, p.p. 147-167, American Mathematical Society, 1996.

[6] B. Carter, K. Park; "How good are genetic algorithms at finding large cliques: An experimental study"; *Tech. Rep. BU-CS-93-015*, Computer Science Dept., Boston University, 1993.

[7] T. Back, S. Khuri; "An evolutionary heuristic for the maximum independent set problem"; *Proc 1st IEEE Conf. Evolutionary Comput.*, p.p. 531-535, 1994.

[8] DIMACS Challenge Problems: "Independent Sets in Graphs", 2000; Refer to: http://www.research.att.com/~njas/doc/graphs.html.

[9] K. S. Narendra, M. A. L. Thathachar; "Learning automata: an introduction"; Prentice-Hall, Englewood Cliffs, 1989.

[10] M. R. Meybodi, H. Beigy and M. Taherkhani, ì Cellular Learning Automata and Its Applications", Journal of Science and Technology, University of Sharif, No. 25, pp.54-77, Autumn/Winter 2003-2004.

[11] M. R. Meybodi and M. R. Kharazmi ì Cellular Learning Automata and Its Application to Image Processingî, Journal of Amirkabir, Vol. 14, No. 56A, pp. 1101-1126, 2004.

[12] H. Beigy and M. R. Meybodi, ì A Mathematical Framework for Cellular Learning Automataî, *Advances on Complex Systems, Vol. 7, No. 3*, pp. 1-25, 2004.