یک الگوریتم مبتنی بر اتوماتای یادگیر سلولی برای حل مساله بزرگترین برش در گراف

مهدی عنایت زارع　　　محمدرضا میبدی

آزمایشگاه محاسبات نرم

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

دانشگاه صنعتی امیرکبیر

تهران　　ایران

mmeybodi@aut.ac.ir, enayatzare@cic.aut.ac.ir

چکیده: مساله بزرگترین برش در گراف دارای کاربردهای فراوانی از جمله طراحی مدارهای مجتمع متراکم و فیزیک آماری می‌باشد. بزرگترین برش در گراف عبارت است از افراز مجموعه رأس‌های گراف به دو زیرمجموعه غیرمشترک به‌گونه‌ای که تعداد (وزن) یال‌هایی که یک‌سر آنها در یک زیر مجموعه و سر دیگرشان در زیر مجموعه دیگر قرار گرفته است, بیشینه شود. مساله بزرگترین برش یکی از مسایل NP-Complete می‌باشد و به‌همین دلیل الگوریتم‌های تقریبی متعددی برای حل آن ارایه شده است. در این مقاله یک الگوریتم مبتنی بر اتوماتای یادگیر سلولی جدید برای حل این مساله پیشنهاد می‌گردد. الگوریتم‌های پیشنهادی با الگوریتم‌های تقریبی سهنی, ژئومنس، الگوریتمهای مبتنی بر اتوماتای سلولی یادگیر و الگوریتم‌های ترکیبی و ژنتیک مقایسه شده است. طبق نتایج به‌دست آمده الگوریتم‌های پیشنهادی نتایج بهتری را در مقایسه با الگوریتم‌های فوق‌الذکر تولید می‌کند.

کلمات کلیدی: مساله بزرگترین برش، الگوریتم‌های تقریبی، اتوماتای یادگیر سلولی

An Algorithm based on Cellular Learning Automata for Solving Maximum Cut Problem in Graph

M. Enayatzare　　　M. R. Meybodi

Computer Engineering and Information Technology Department
Amirkabir University of Technology
Tehran　　Iran

mmeybodi@aut.ac.ir, enayatzare@cic.aut.ac.ir

Abstract: The Maximum cut problem has applications in many fields including VLSI circuit design and statistical physics. Maximum cut is to partitioning the vertex set of graph into two disjoint part so that the number (or weight) of edges joining vertices in different parts as large as possible. Maximum cut is known to be NP-Complete and therefore many approximation algorithms are reported for solving this problem. In this paper, we proposed an algorithm for solving this problem using cellular learning automata is proposed. In order to study the performance of the proposed algorithm computer simulations have been conducted and the results are compared with the results obtained for Sehni, Geomens, Combinatorial and Genetic algorithms and two existing algorithms based on cellular learning automata. The results of comparison show the superiority of the proposed algorithm over the existing algorithms.

Keywords: Maximum Cut Problem, Approximate Algorithms, Cellular Learning Automata

۱- مقدمه

بزرگترین برش[1] گراف که دارای کاربردهای متعددی از جمله طراحی مدارهای مجتمع متراکم و فیزیک آماری می‌باشد[۲] عبارت است از افراز رأس‌های گراف به دو قسمت به‌گونه‌ای که تعداد (وزن) یال‌هایی که یک‌سر آنها در هر کدام از این قسمت‌ها قرار گرفته است, بیشینه شود. بطور رسمی, در یک گراف ساده G با مجموعه‌ی رأس‌های V, مجموعه‌ی یال‌های E و وزن‌های نامنفی W_{ij} روی یال‌های $\{i, j\} \in E$, مساله بزرگترین برش عبارت است از یافتن مجموعه‌های S_1 و S_2 از راس‌ها به‌قسمی‌که اولاً $V = S_1 \cup S_2$, ثانیاً $S_1 \cap S_2 = \phi$ و ثالثاً وزن برش (S_1, S_2) بیشینه شود. منظور از وزن برش (S_1, S_2) وزن یال‌هایی است که یک سرشان در S_1 و سر دیگرشان در S_2 قرار دارد. برای سادگی معمولاً فرض می‌کنند برای هر $\{i, j\} \in E$, $W_{ij}=1$. مساله بزرگترین برش یکی از مسایلNP-Complete می‌باشد[۱۴] و به‌همین دلیل الگوریتم با پیچیدگی چندجمله‌ای برای آن وجود ندارد. با توجه به این‌که در بسیاری از کاربردها یافتن یک راه حل نزدیک به بهینه نیز می‌تواند مفید باشد, الگوریتم‌های تقریبی[2] متعددی برای حل آن ارایه شده است. از جمله این الگوریتم‌ها می‌توان به الگوریتم‌های سهنی[3][۲۱], ویتانی[4][۲۳], پولژاک[5][۲۰], هاگلین[6][۱۲], هوف‌میستر[7][۱۳] و ژئومنس[8][۱۱] اشاره کرد. علاوه بر الگوریتم‌های تقریبی, الگوریتم‌های مکاشفه‌ای و تصادفی متعددی نیز برای مسأله بزرگترین برش ارایه شده است. در این الگوریتم‌ها تلاش بر این است که با استفاده از روش‌های مکاشفه‌ای[9], راه‌حلی سریع و یا دقیق‌تر برای این مساله ارایه کنند. نمونه‌ای از این الگوریتم‌ها را می‌توان در [۹], [۱۰] و [۱۵] مشاهده کرد. در این مقاله یک الگوریتم مبتنی بر اتوماتای یادگیر سلولی جدید برای حل این مساله پیشنهاد می‌گردد. الگوریتم‌های پیشنهادی با الگوریتم‌های تقریبی سهنی, ژئومنس، دو الگوریتم مبتنی بر اتوماتای سلولی یادگیر که آنها را CLA1 و CLA2 مینامیم[26] و الگوریتم‌های ترکیبی و ژنتیک مقایسه شده است. طبق نتایج به‌دست آمده الگوریتم‌های پیشنهادی نتایج بهتری را در مقایسه با الگوریتم‌های فوق‌الذکر تولید می‌کند.

ادامه مقاله به این صورت سازمان‌دهی شده است. در بخش‌های ۲ تا ۴ به ترتیب به معرفی اجمالی اتوماتای سلولی[10], اتوماتاهای یادگیر[11] و اتوماتای یادگیر سلولی می‌پردازیم. در بخش ۵ الگوریتم پیشنهادی شرح داده می‌شود و در بخش ۶ نتایج آزمایش‌ها ارایه می‌گردد. بخش نهایی مقاله نتیجه‌گیری می‌باشد.

۲- اتوماتای سلولی

اتوماتای سلولی (CA) [۲۴] [۲۵] در اواخر دهه ۱۹۴۰ توسط ون‌نیومن[12] مطرح و پس از او توسط ریاضی‌دانی بنام اولام[13] به عنوان مدلی برای بررسی رفتار سیستم‌های پیچیده پیشنهاد شد. اتوماتای سلولی در حقیقت سیستم‌های دینامیکی گسسته‌ای هستند که رفتارشان کاملاً بر اساس ارتباط محلی استوار است. در اتوماتای سلولی، فضا بصورت یک شبکه تعریف می‌گردد که به هر خانه آن یک سلول گفته می‌شود. زمان بصورت گسسته پیش می‌رود و قوانین آن بصورت سرتاسری است که از طریق آن در هر مرحله هر سلول، وضعیت جدید خود را با در نظر گرفتن همسایه‌های مجاور خود بدست می‌آورد. قوانین اتوماتای سلولی، نحوه تأثیر پذیرفتن سلول از سلول‌های همسایه خود را مشخص می‌کند. یک سلول را همسایه سلول دیگر گوئیم هرگاه بتواند آن را در یک مرحله و براساس قانون حاکم تحت تأثیر قرار دهد. در بدست آوردن وضعیت کنونی سلول علاوه بر وضعیت قبلی سلول‌های همسایه، می‌توان وضعیت قبلی خود سلول را نیز دخالت داد.

تعریف ۱ اتوماتای سلولی d بعدی یک چندتایی $CA = (Z^d, \phi, N, F)$ است به‌طوری‌که:

---

[1] Maximum Cut
[2] Approximation
[3] Sahni
[4] Vitanyi
[5] Poljak
[6] Hoglin
[7] Hofmeister
[8] Geomens
[9] Heuristic
[10] Cellular Automata (CA)
[11] Learning Automata (LA)
[12] Von Neumann
[13] Ulam

- Z^d یک شبکه از d تایی‌های مرتب از اعداد صحیح می‌باشد. این شبکه می‌تواند یک شبکه متناهی، نیمه متناهی یا نامتناهی باشد.
- $\phi = \{1,\dots,m\}$ یک مجموعه متناهی از حالت‌ها می‌باشد.
- $N = \{\bar{x}_1,\dots,\bar{x}_{\bar{m}}\}$، $\bar{x}_i \in Z^d$، یک زیر مجموعه متناهی از Z^d می‌باشد که بردار همسایگی خوانده می‌شود. بردار همسایگی، موقعیت نسبی همسایگان را برای هر سلول u در شبکه سلولی به صورت زیر مشخص می‌کند:

$$N(u) = \{u + \bar{x}_i \mid i = 1,\dots,\bar{m}\}$$

تابع $N(u)$ دو شرط زیر را ارضا می‌کند:

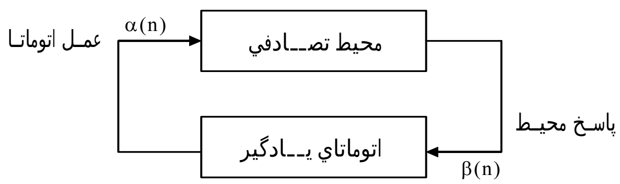
$$\forall u \in Z^d \Rightarrow u \in N(u)$$
$$\forall u,v \in Z^d \Rightarrow u \in N(v) \wedge v \in N(u)$$

- $F : \underline{\phi}^{\bar{m}} \rightarrow \underline{\phi}$ قانون محلی CA می‌باشد

در مدل‌سازی سیستم‌های فیزیکی و بیولوژیکی، گاهی لازم است که قوانین را بصورت احتمالی در نظر بگیریم. رفتار احتمالی را می‌توان به عنوان نویز در سیستم تعبیر نمود. یکی از اشکالات اتوماتای سلولی، تعیین فرم قطعی قوانین مورد نیاز برای یک کاربرد خاص است. راه‌حل‌های متفاوتی در برخورد با این مشکلات به نظر می‌رسد. یکی از این راه‌حل‌ها احتمالاتی کردن قوانین می‌باشد. به این ترتیب که تمام قانون‌های امکان‌پذیر را در نظر بگیریم و برای فعال شدن آنها احتمالی در نظر بگیریم. اما مشکلی که باز گریبان‌گیر ما خواهد شد آن است که شناسایی همین احتمال‌ها در سیستم‌های ناشناخته عملی نمی‌باشد. پس باید به سمتی حرکت کنیم که به نحوی خود ابزار با گذشت زمان بتواند قوانین مناسب را استخراج کند.

۳- اتوماتاهای یادگیر

یک اتوماتای یادگیر[۱۹]، ماشینی است که می‌تواند تعدادی متناهی عمل را انجام دهد. هر عمل انتخاب شده توسط یک محیط احتمالی ارزیابی می‌شود و نتیجه ارزیابی در قالب پاسخی مثبت یا منفی به اتوماتا داده می‌شود و اتوماتا از این پاسخ در انتخاب عمل بعدی تاثیر می‌گیرد. هدف نهایی این است که اتوماتا یاد بگیرد تا از بین اعمال خود بهترین عمل را انتخاب کند. بهترین عمل، عملی است که احتمال دریافت پاداش از محیط را به حداکثر برساند. کارکرد اتوماتای یادگیر در تعامل با محیط، در شکل ۱ مشاهده می‌شود.



شکل ۱- ارتباط بین محیط و اتوماتای یادگیر

محیط را می‌توان توسط سه‌تایی $E \equiv \{\alpha, \beta, c\}$ نشان داد که در آن $\alpha \equiv \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ مجموعه ورودی‌ها، $\beta \equiv \{\beta_1, \beta_2, \dots, \beta_m\}$ مجموعه خروجی‌ها و $c \equiv \{c_1, c_2, \dots, c_r\}$ مجموعه احتمال‌های جریمه می‌باشد. هرگاه β مجموعه دو عضوی باشد، محیط از نوع P می‌باشد. در چنین محیطی $\beta_1 = 1$ به عنوان جریمه و $\beta_2 = 0$ به عنوان پاداش در نظر گرفته می‌شود. در محیط از نوع Q، $\beta(n)$ می‌تواند به طور گسسته یک مقدار از مقادیر محدود در فاصله [۰،۱] و در محیط از نوع S، $\beta(n)$ متغیر تصادفی در فاصله [۰،۱] است. c_i احتمال این است که عمل α_i نتیجه نامطلوب[1] داشته باشد. در محیط ایستا[2] مقادیر c_i بدون تغییر می‌مانند، حال آنکه در محیط غیرایستا[3] این مقادیر در طی زمان تغییر می‌کنند.

---
[1] Unfavorable
[2] Stationary
[3] Non-Stationary

اتوماتای یادگیر با ساختار ثابت توسط پنج‌تایی $\{\alpha, \beta, F, G, \phi\}$ نشان داده می‌شود 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, $\phi \equiv \{\phi_1, \phi_2, \cdots, \phi_s\}$ مجموعه وضعیت‌های داخلی اتوماتا، $F: \phi \times \beta \to \phi$ تابع تولید وضعیت جدید اتوماتا و $G: \phi \to \alpha$ تابع خروجی می‌باشد که وضعیت کنونی اتوماتا را به خروجی بعدی می‌نگارد.

اتوماتای یادگیر با ساختار متغیر را می‌توان توسط چهارتایی $\{\alpha, \beta, p, T\}$ نشان داد که $\alpha = \{\alpha_1, ..., \alpha_r\}$ مجموعه عمل‌های اتوماتا، $\beta = \{\beta_1, ..., \beta_m\}$ مجموعه ورودی‌های اتوماتا، $p = \{p_1, .., p_r\}$ بردار احتمال انتخاب هریک از عمل‌ها و $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ الگوریتم یادگیری می‌باشد. الگوریتم زیریک نمونه از الگوریتم‌های یادگیری خطی است. فرض کنید عمل α_i در مرحله nام انتخاب شود.

- پاسخ مطلوب

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

- پاسخ نامطلوب

$$p_i(n+1) = (1-b)p_i(n)$$
$$p_j(n+1) = (b/r-1) + (1-b)p_j(n) \quad \forall j \quad j \neq i$$

در روابط بالا، a پارامتر پاداش و b پارامتر جریمه می‌باشند. با توجه به مقادیر a و b سه حالت زیر را می‌توان در نظر گرفت. زمانی‌که a و b با هم برابر باشند، الگوریتم را L_{RP} می نامیم، زمانی‌که b از a خیلی کوچک‌تر باشد، الگوریتم را $L_{R\varepsilon P}$ می‌نامیم. و زمانی‌که b مساوی صفر باشد الگوریتم را L_{RI} می‌نامیم[۲۲].

۴- اتوماتای یادگیر سلولی

بسیاری از مسایل را نمی‌توان با استفاده از یک اتوماتای یادگیر تنها حل کرد, بلکه قدرت اصلی اتوماتای یادگیر زمانی آشکار می‌شود که آنها به صورت دسته‌جمعی بکار روند. با توجه به این مساله و ضعف‌های عنوان شده برای اتوماتای سلولی، با ترکیب این دو مدل، مدل جدیدی با نام اتوماتای یادگیر سلولی ایجاد گردید[۱۶]. در زیر تعریف فرمال اتوماتای یادگیر سلولی ارائه شده است[۳].

تعریف ۲ (اتوماتای یادگیر سلولی): اتوماتای یادگیر سلولی d بعدی یک چندتایی $CLA = (Z^d, \phi, A, N, F)$ است به طوری‌که:

- Z^d یک شبکه از dتایی‌های مرتب از اعداد صحیح می‌باشد. این شبکه می‌تواند یک شبکه متناهی، نیمه متناهی یا نامتناهی باشد.
- ϕ یک مجموعه متناهی از حالت‌ها می‌باشد.
- A، یک مجموعه از اتوماتاهای یادگیر است که هر یک از آنها به یک سلول از اتوماتای سلولی نسبت داده می‌شود.
- $N = \{\bar{x}_1, ..., \bar{x}_{\bar{m}}\}$ یک زیر مجموعه متناهی از Z^d می‌باشد که بردار همسایگی نامیده می‌شود.
- $F: \underline{\phi}^{\bar{m}} \to \underline{\beta}$ قانون محلی اتوماتای یادگیر سلولی می‌باشد به طوری‌که $\underline{\beta}$ مجموعه مقادیری است که می‌تواند به عنوان سیگنال تقویتی پذیرفته شود.

عملکرد اتوماتای یادگیر سلولی را می‌توان به شرح زیر بیان کرد. در هر لحظه هر اتوماتای یادگیر در اتوماتای یادگیر سلولی یک عمل از مجموعه اعمال خود را انتخاب می‌کند. این عمل می‌تواند بر اساس مشاهدات قبلی و یا به صورت تصادفی انتخاب شود. عمل انتخاب شده با توجه به اعمال انتخاب شده توسط سلول‌های همسایه و قانون حاکم بر اتوماتای یادگیر سلولی پاداش داده و یا جریمه می‌شود. با توجه به اینکه عمل انتخاب شده پاداش گرفته و یا جریمه شده است, اتوماتا رفتار خود را تصحیح کرده و ساختار داخلی اتوماتا بهنگام می‌گردد. معمولاً عمل بروزرسانی تمام اتوماتاها به صورت همزمان انجام می‌شود. بعد از بروزرسانی، هر اتوماتا در اتوماتای یادگیر سلولی دوباره یک عمل از مجموعه اعمال خود را انتخاب کرده و انجام می‌دهد. فرآیند انتخاب عمل و دادن پاداش و یا جریمه تا زمانی که سیستم به حالت پایدار برسد و یا یک معیار از قبل تعریف شده‌ای برقرار شود, ادامه می‌یابد. عمل بهنگام‌سازی ساختار اتوماتاهای موجود در اتوماتای یادگیر سلولی توسط الگوریتم یادگیری انجام می‌شود.

اتوماتای یادگیر سلولی را یکنواخت می‌گوییم، اگر برای تمام سلول‌ها، تابع همسایگی، قانون محلی و اتوماتاهای یادگیر یکسان باشند در غیر این صورت اتوماتای یادگیر سلولی غیریکنواخت نامیده می‌باشد. نوع دیگر اتوماتای یادگیر سلولی، اتوماتای یادگیر سلولی باز (OCLA) می‌باشد[۶]. در OCLA علاوه بر محیط محلی[1] یک محیط سراسری[2] نیز برای آن در نظر گرفته شده است[۶] [۷]. در OCLA دادن جریمه و یا پاداش به عمل انتخاب شده توسط یک سلول علاوه بر اعمال انتخابی توسط همسایگانش به پاسخ محیط سراسری نیز بستگی دارد. در[۷] اثبات شده است این مدل همانند **CLA** بسته [۳] [۱۶]، برای قوانین جابجایی‌پذیر، می‌تواند به نقاط بهینه محلی همگرا شود. در صورتی‌که اتصالات بین اتوماتاهای یادگیر منظم (مانند آرایه یک بعدی و یا آرایه دو بعدی) نباشد به آن اتوماتای یادگیر سلولی نامنظم[3][۱] گفته می‌شود. اگر بروزرسانی سلول‌ها به صورت همگام برای تمامی سلول‌ها صورت بگیرد، اتوماتای یادگیر سلولی همگام[4] و در غیر این صورت ناهمگام[5][۴] [۵] نامیده می‌شود. برای بعضی از کاربردهای اتوماتای یادگیر سلولی می‌توان به [۱] [۳] [۴] [۵] [۶] [۷] [۱۶] [۱۷] [۱۸] مراجعه نمود.

۵- الگوریتم پیشنهادی

در این بخش یک الگوریتم تقریبی مبتنی بر اتوماتای یادگیر سلولی برای حل مساله بزرگترین برش در گراف پیشنهاد می‌گردد. در این الگوریتم ابتدا یک اتوماتای یادگیر سلولی غیریکنواخت همگام نامنظم متناظر با گراف مساله ایجاد می‌شود. بطور مثال برای گراف شکل۲-الف اتوماتای یادگیرسلولی نامنظم شکل ۲-ب ایجاد می‌شود. هر یک از اتوماتاهای یادگیر در اتوماتای یادگیر سلولی دارای دو عمل "عضو شدن در مجموعه S_1" و "عضو شدن در مجموعه S_2" می‌باشند. شماره هر سلول در اتوماتای یادگیر سلولی شماره راس متناظر با آن سلول در گراف مساله می‌باشد (از ۱ تا n). اتوماتای یادگیر در هر سلول از نوع L_{RP} با ضریب پاداش و جریمه ۰٫۰۱ می‌باشد. در ابتدا احتمال انتخاب اعمال هر یک از اتوماتاهای یادگیر مساوی و برابر ۱/۲ در نظر گرفته می‌شود.

شکل ۲- الف- یک گراف بدون‌جهت بدون وزن   **شکل ۲-ب- اتوماتای یادگیر سلولی متناظر با شکل ۲-الف**

الگوریتم پیشنهادی که آن را **CLA-3** نیز مینامیم دارای دو مرحله میباشد. در مرحله اول یک تطابق بیشینه[6] در گراف مساله پیدا میشود **ودر مرحله دوم با استفاده از اتوماتاي یادگیر سلولي** اقدام به یافتن یک برش ماکزیمال مینماید.

مرحله اول(یافتن یک تطابق بیشینه): زیرمجموعه M از مجموعه یال‌های گراف یک تطابق نامیده می‌شود, اگر عضوهای آن یال‌های پیوندی بوده و هیچ دوتای آن‌ها مجاور نباشند و این‌طور بیان می‌شود که دو سر یال‌های M تحت M مطابق شده‌اند. اگر یالی از M مجاور رأس v باشد, می‌گوییم M رأس v را آلوده کرده است.M یک تطابق ماکزیمم است اگر هیچ تطابق M'ای با شرط $|M|<|M'|$ در گراف مورد نظر وجود نداشته باشد. ما برای یافتن بزرگترین تطابق در گراف از الگوریتم ۱-فاکتورگیری برای گراف‌های کامل با تعداد راس‌های زوج استفاده می‌کنیم[۱۳]. فرض کنید تعداد راس‌های گراف مساله (n) زوج باشد (در صورتی‌که تعداد راس‌های گراف فرد باشند یک راس اضافی به گراف افزوده و بین آن راس و تمام راس‌های دیگر گراف یال به وزن صفر ایجاد می‌کنیم.). سپس گراف مورد نظر را با افزودن یال‌هایی به وزن صفر به

[1] Local Environment
[2] Global Environment
[3] Irregular Cellular Learning Automata
[4] Synchronous Cellular Learning Automata
[5] Asynchronous Cellular Learning Automata
[6] Matching

یک گراف کامل n راسی (K_n) تبدیل می‌کنیم. برای گراف حاصل, $M_j = \{\{j,n\},\{\{j-i, j+i\}, i=1,2...,\frac{n}{2}-1\}\}, j=1,2...,n-1$، تعداد n-1 تطابق به اندازه $n/2$ به‌دست می‌دهد. البته در رابطه فوق‌الذکر عمل جمع به صورت چرخشی در مجموعه $\{1,2\ldots,n\text{-}1\}$ انجام می‌پذیرد. حال K را مجموعه n-1 تطابق حاصل از الگوریتم بالا و $M \in K$ را یک تطابق با بزرگترین وزن (اندازه) فرض می‌کنیم. از این تطابق در الگوریتم پیشنهادی برای یافتن یک برش بزرگ از گراف مساله استفاده می‌کنیم.

مرحله دوم: در اتوماتای یادگیر سلولی که برای حل این مساله در نظر گرفته شده است, دادن پاداش یا جریمه به عمل انتخابی توسط یک سلول, علاوه بر اعمال انتخاب شده توسط سلول و همسایگانش, به اعمال انتخاب شده در مرحله قبل توسط همسایگانش نیز بستگی دارد. در اولین مرحله از الگوریتم, اتوماتای یادگیر هر سلول از بین اعمال "عضو شدن در مجموعه S_1" و یا "عضو شدن در مجموعه S_2" یکی را انتخاب می‌کند. عمل انتخابی یک سلول در این مرحله به عنوان عمل قبلی این سلول برای مرحله بعد محسوب می‌شود. در ادامه الگوریتم، در هر مرحله هر سلول از اتوماتای یادگیر سلولی یکی از اعمال خود را انتخاب می‌کند و سپس عمل انتخاب شده توسط سلول طبق قانونی که در ادامه توضیح داده می‌شود پاداش گرفته و یا جریمه می‌شود و این مراحل آنقدر ادامه می‌یابد تا در یک مرحله هیچ سلولی جریمه نشود. در پایان این مرحله اندازه برش حاصل به این صورت بدست می‌آید. ابتدا برای هر سلول تعداد سلول‌های همسایه‌ای که عمل انتخابی‌شان متفاوت با عمل آن سلول می‌باشد تعیین می‌گردد. نصف مجموع تعداد این گونه همسایه‌ها برای تمام سلول‌ها, اندازه برش می‌باشد.

هر سلول برای دادن پاداش و یا جریمه به‌صورت زیر عمل می‌کند (قانون حاکم بر اتوماتای یادگیر سلولی):

- اگر سلول مورد نظر به‌وسیله تطابق به دست آمده آلوده شده باشد و همسایه‌ای که سلول مدنظر با آن مطابق شده است, متناظر با یک راس اصلی گراف مساله باشد (سلول همسایه متناظر با راس اضافه شده برای زوج کردن تعداد راس‌ها نباشد) آنگاه:
  - اگر عمل انتخاب شده به‌وسیله سلول با عمل انتخاب شده به‌وسیله سلول همسایه مطابقش یکسان باشد آنگاه:
    - ✓ اگر درجه سلول از سلول همسایه مطابقش بیشتر باشد آنگاه:
      - عمل انتخابی توسط سلول جریمه می‌شود.
    - ✓ در غیر این‌صورت:
      - اگر تعداد همسایگانی از سلول که عمل خود را تغییر نداده‌اند (عملشان مشابه با عمل انتخابی آنها در مرحله قبل می‌باشد) و عمل انتخابی آنها نیز مشابه با عمل انتخابی به وسیله سلول موردنظر است بیشتر از تعداد همسایگانی باشند که عمل خود را تغییر نداده‌اند و عمل انتخابی آنها نیز متفاوت با عمل انتخابی بوسیله سلول است آنگاه عمل سلول جریمه می‌شود و در غیر این‌صورت عمل انتخابی توسط سلول پاداش دریافت می‌کند.
  - در غیر این‌صورت (اگر عمل انتخاب شده به‌وسیله سلول با عمل انتخاب شده به‌وسیله سلول همسایه مطابقش یکسان نباشد) آنگاه:
    - ✓ اگر تعداد همسایگانی از سلول که عمل خود را تغییر نداده‌اند (عملشان مشابه با عمل انتخابی آنها در مرحله قبل می‌باشد) و عمل انتخابی آنها نیز مشابه با عمل انتخابی به وسیله سلول موردنظر است بیشتر از تعداد همسایگانی باشند که عمل خود را تغییر نداده‌اند و عمل انتخابی آنها نیز متفاوت با عمل انتخابی بوسیله سلول است آنگاه عمل سلول جریمه می‌شود و در غیر این‌صورت عمل انتخابی توسط سلول پاداش دریافت می‌کند.
- اگر سلول مورد نظر به‌وسیله تطابق آلوده نشده باشد و یا با سلولی مطابق شده باشد که متناظر با یک راس اصلی از گراف نمی‌باشد آنگاه:
  - اگر تعداد همسایگانی از سلول که عمل خود را تغییر نداده‌اند (عملشان مشابه با عمل انتخابی آنها در مرحله قبل می‌باشد) و عمل انتخابی آنها نیز مشابه با عمل انتخابی به وسیله سلول موردنظر است بیشتر از تعداد همسایگانی باشند که عمل خود را تغییر نداده‌اند و عمل انتخابی آنها نیز متفاوت با عمل انتخابی بوسیله سلول است آنگاه عمل سلول جریمه می‌شود و در غیر این‌صورت عمل انتخابی توسط سلول پاداش دریافت می‌کند.

شبه کد این الگوریتم در شکل ۳ نشان داده شده است.

```
Algorithm CLA-3
Input: Graph G(V,E)
Output: the size of the Max-Cut
Begin
    Find the maximum matching in input graph using 1-factorization algorithm for complete graphs
    Construct an irregular CLA for input graph
    Repeat
        For all cells do in parallel
            Select an action
            SN ← the number of neighbors that have not changed their actions and their actions is equal to the
                                                                        action of the cell
            DN ← the number of neighbors that have not changed their actions and their actions is not equal to
                                                                        the action of  the cell
            If (cell is saturated by maximum matching and its matched neighboring cell is an original cell)
                If (cell's action is equal to the action of its matched neighboring cell)
                    If (cell's degree is less than the degree of its matched neighboring cell)
                        Penalize the cell's action
                    Else // cell's degree is greater  than the degree of its matched neighboring cell
                        If(SN>DN)
                            Penalize the cell's action
                        Else
                            Reward the cell's action
                        End if
                    End if
                Else // cell's action is equal to the action of its matched neighboring cell
                    If(SN>DN)
                        Penalize the cell's action
                    Else
                        Reward the cell's action
                    End if
                End if
            Else // cell is not  saturated by maximum matching or  its matched neighboring cell is an auxiliary
            cell
                If (SN>DN)
                    Penalize the cell's action
                Else
                    Reward the cell's action
                End if
            End if
    Until Actions of all cells are rewarded
    Return the size of the Max Cut
End
```

شکل ۳- الگوریتم پیشنهادی

۶- نتایج آزمایش‌ها

گراف‌های مورد بررسی گراف‌های موجود در مجموعه استاندارد DIMACS [۸] با تعداد ۱۰, ۲۰, ..., ۱۰۰, ۱۵۰, ۲۰۰ و ۲۵۰ راس می‌باشند که یال‌های آنها به‌صورت تصادفی و با احتمال ۰,۵ انتخاب شده‌اند. نتایج حاصل از الگوریتم‌های پیشنهادی با نتایج الگوریتم‌های سهنی[۲۱], ژئومنس[۱۱], الگوریتم ترکیبی[۱۳] و ژنتیک[۱۵] مقایسه شده است. معیارهای مقایسه اندازه برش به دست آمده و زمان اجرا بر حسب ثانیه می‌باشد. برای اجرای الگوریتم‌ها یک کامپیوتر شخصی با پردازنده اینتل P4 2.4GHz و با ۵۱۲ مگابایت حافظه اصلی مورد استفاده قرار گرفته است. هر یک از نتایج که در جدول ۱ و ۲ و نمودارهای ۴ تا ۱۱ گزارش شده است, متوسط ۵۰ بار اجرا می‌باشد.

همان‌گونه که نمودارها نشان می‌دهند, الگوریتم CLA1 از نظر اندازه برش بدست آمده بهتر از الگوریتم‌های سهنی و ژنتیک و بد تر از الگوریتم‌های ترکیبی و ژئومنس میباشد. در مقابل الگوریتم CLA1 در مقایسه با الگوریتم‌های ژئومنس و ژنتیک سریع‌تر ولی در مقایسه با الگوریتم‌های سهنی و ترکیبی کندتر عمل میکند. الگوریتم CLA2 نسبت به سایر الگوریتم‌ها از نظر اندازه برش به دست آمده بهتر عمل می‌کند ولی زمان اجرای آن در مقایسه با الگوریتم‌های ترکیبی, سهنی و ژنتیک بیشتر است. الگوریتم CLA2 در مقایسه با الگوریتم ژئومنس

دارای اندازه برش بهتر و همچنین زمان اجرای کمتری میباشد. الگوریتم CLA1 سریع می‌باشد و در عین حال نتایج قابل قبولی را از نظر اندازه برش تولید می‌کند. همان‌گونه که نمودارها نشان می‌دهند الگوریتم CLA1 از نظر اندازه برش از الگوریتم های سهنی, ترکیبی و ژنتیک بهتر و به الگوریتم ژئومنس نزدیک است. زمان اجرای الگوریتم CLA1 از زمان اجرای الگوریتم‌های سهنی, ژنتیک و ژئومنس کمتر میباشد ولی از الگوریتم ترکیبی بیشتراست.

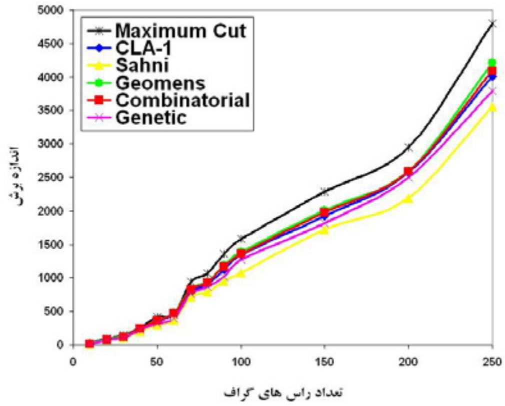
مقایسه CLA1، CLA2 و الگوریتم پیشنهادی با یکدیگر نیز نشان می‌دهد که الگوریتم CLA2 از نظر اندازه برش بدست آمده از دو الگوریتم دیگر بهتر عمل می‌کند ولی زمان اجرای آن در مقایسه با دو الگوریتم دیگر بیشتر است. از بین الگوریتم CLA1 و الگوریتم پیشنهادی، الگوریتم پیشنهادی نتایج بهتری را از نظر اندازه برش تولید می‌کند. الگوریتم پیشنهادی در مقایسه با دو الگوریتم CLA1 و CLA2 دارای زمان اجرای کمتری میباشد. از بین الگوریتمهای CLA1 و CLA2 الگوریتم CLA1 سریع‌تر می‌باشد.

جدول ۱- اندازه برش به‌دست آمده برای الگوریتم‌های مختلف برای گراف‌هایی با تعداد راس‌های مختلف

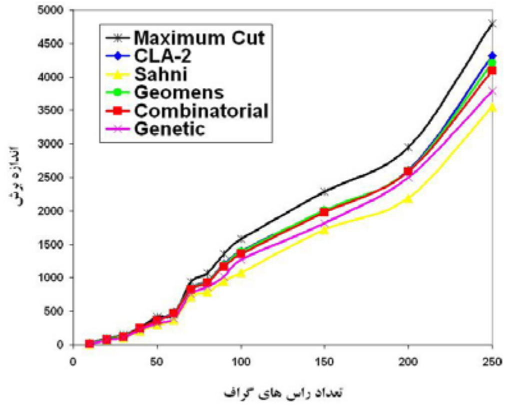
Genetic	Combinatorial	Geomens	Sahni	CLA-3	CLA-2	CLA-1	اندازه برش	الگوریتم / تعداد راس‌ها
۱۳	۱۶	۲۲	۱۴	۲۰	۲۵	۱۹	۲۵	۱۰
۷۸	۷۹	۸۵	۷۷	۸۲	۸۸	۸۲	۹۶	۲۰
۱۱۰	۱۱۶	۱۲۹	۱۰۳	۱۲۴	۱۳۳	۱۱۸	۱۴۶	۳۰
۲۲۲	۲۴۵	۲۳۰	۲۰۳	۲۴۸	۲۴۶	۲۳۷	۲۶۱	۴۰
۳۲۴	۳۵۹	۳۶۳	۲۹۷	۳۶۰	۳۷۰	۳۵۰	۴۱۳	۵۰
۳۹۹	۴۷۲	۴۷۷	۳۷۱	۴۷۵	۴۹۰	۴۶۱	۴۵۳	۶۰
۷۶۲	۸۱۹	۸۳۰	۷۱۱	۸۲۷	۸۴۲	۷۹۳	۹۴۵	۷۰
۸۶۵	۹۲۶	۹۳۷	۷۹۲	۹۳۵	۹۵۵	۹۱۱	۱۰۶۷	۸۰
۱۰۱۲	۱۱۶۳	۱۱۸۹	۹۵۱	۱۱۷۹	۱۲۰۷	۱۱۲۶	۱۳۵۴	۹۰
۱۲۷۱	۱۳۶۰	۱۳۸۶	۱۰۸۴	۱۳۷۱	۱۴۰۱	۱۳۴۱	۱۵۷۸	۱۰۰
۱۸۱۶	۱۹۷۷	۲۰۰۷	۱۷۲۳	۱۹۸۶	۲۰۱۱	۱۹۲۵	۲۲۸۵	۱۵۰
۲۵۰۵	۲۵۹۰	۲۵۹۲	۲۱۹۳	۲۵۸۹	۲۶۰۸	۲۵۷۳	۲۹۵۲	۲۰۰
۳۷۹۶	۴۰۹۶	۴۲۰۸	۳۵۵۹	۴۱۷۰	۴۳۱۹	۴۰۰۹	۴۷۹۲	۲۵۰

جدول ۲- زمان اجرا بر حسب ثانیه برای الگوریتم‌های مختلف برای گراف‌هایی با تعداد راس‌های مختلف

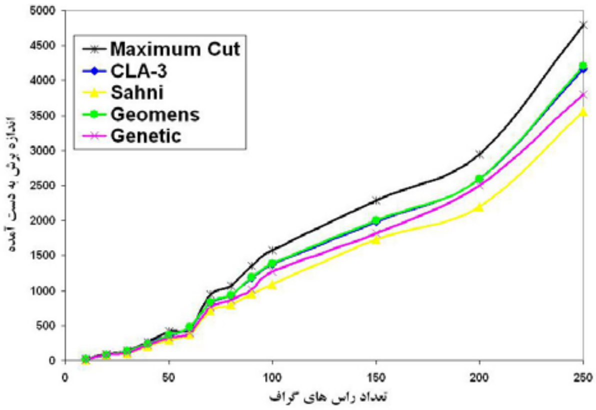
Genetic	Combinatorial	Geomens	Sahni	CLA-3	CLA-2	CLA-1	الگوریتم / تعداد راس‌ها
۱۱,۷۵۲	۰,۰۰۹	۲۶,۷۳۷	۰,۷۱۷	۰,۶۷۳	۱۱,۱۰۵	۲,۲۰۲	۱۰
۱۲,۰۰۹	۰,۰۱۰	۳۵,۱۹۹	۰,۷۹۵	۰,۶۵۹	۱۲,۳۵۹	۲,۹۳۵	۲۰
۱۲,۶۳۲	۰,۰۱۱	۳۹,۰۱۱	۰,۸۶۶	۰,۷۷۸	۱۴,۸۰۶	۳,۶۱۸	۳۰
۱۳,۱۰۷	۰,۰۱۱	۴۸,۲۲۰	۰,۹۵۰	۰,۸۵۲	۱۶,۰۰۳	۴,۰۱۷	۴۰
۱۳,۸۰۵	۰,۰۱۱	۵۴,۲۴۴	۱,۰۲۳	۰,۹۹۴	۱۷,۴۳۲	۴,۸۲۴	۵۰
۱۳,۹۵۵	۰,۰۱۲	۶۱,۰۷۳	۱,۱۷۴	۱,۰۸۵	۱۸,۸۵۶	۵,۲۳۱	۶۰
۱۴,۰۷۱	۰,۰۱۲	۶۹,۸۶۰	۱,۱۹۰	۱,۱۰۴	۱۹,۴۰۲	۵,۵۱۰	۷۰
۱۴,۱۵۱	۰,۰۱۳	۷۰,۴۵۴	۱,۳۶۷	۱,۱۸۶	۲۰,۱۱۲	۵,۷۰۳	۸۰
۱۴,۲۲۶	۰,۰۱۳	۷۳,۱۰۳	۱,۵۰۲	۱,۳۷۳	۲۱,۴۰۲	۵,۹۰۲	۹۰
۱۴,۲۶۸	۰,۰۱۳	۷۸,۳۶۲	۱,۶۴۸	۱,۴۱۸	۲۲,۰۹۸	۶,۰۴۵	۱۰۰
۱۸,۹۰۸	۰,۰۲۸	۸۶,۶۰۰	۲,۱۳۴	۱,۹۵۹	۳۵,۱۸۴	۷,۱۱۶	۱۵۰
۲۳,۲۵۰	۰,۰۵۵	۹۴,۰۲۳	۳,۰۹۸	۲,۷۸۰	۴۴,۸۲۵	۸,۲۰۸	۲۰۰
۲۸,۴۵۰	۰,۰۸۹	۱۰۸,۵۸۷	۴,۲۴۴	۳,۸۵۳	۵۶,۰۹۴	۱۰,۰۰۵	۲۵۰



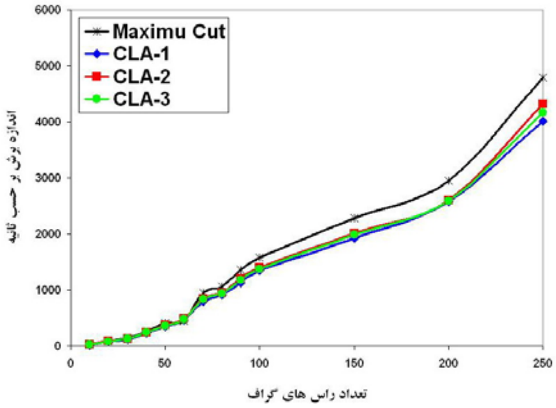
شکل ۴- مقایسه الگوریتم پیشنهادی اول با الگوریتم‌های سهنی، ژئومنس، ترکیبی و ژنتیک بر اساس اندازه برش به‌دست آمده



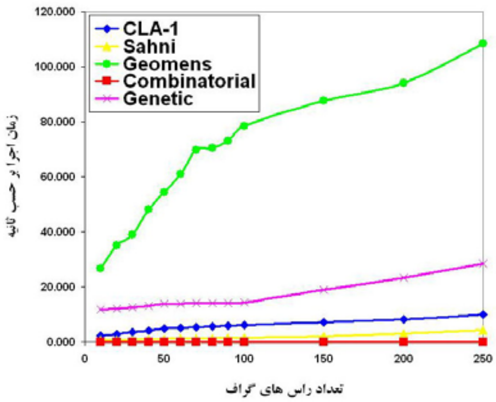
شکل ۵- مقایسه الگوریتم پیشنهادی دوم با الگوریتم‌های سهنی، ژئومنس، ترکیبی و ژنتیک بر اساس اندازه برش به‌دست آمده



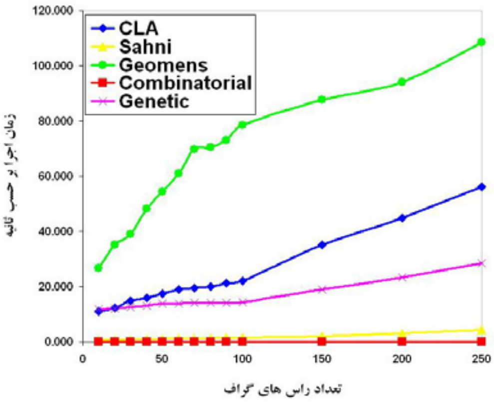
شکل ۶- مقایسه الگوریتم پیشنهادی سوم با الگوریتم‌های سهنی، ژئومنس و ژنتیک بر اساس اندازه برش به‌دست آمده



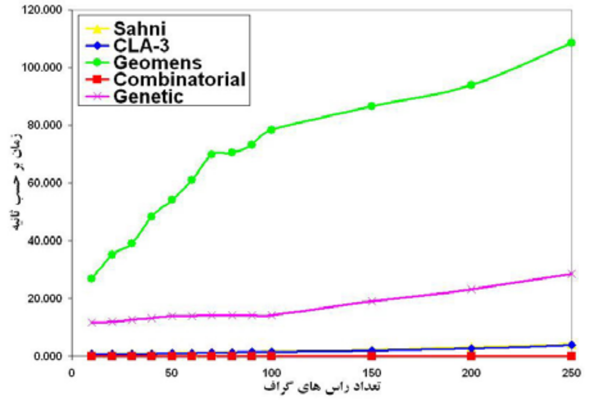
شکل ۷- مقایسه سه الگوریتم پیشنهادی با همدیگر بر اساس اندازه برش به دست آمده



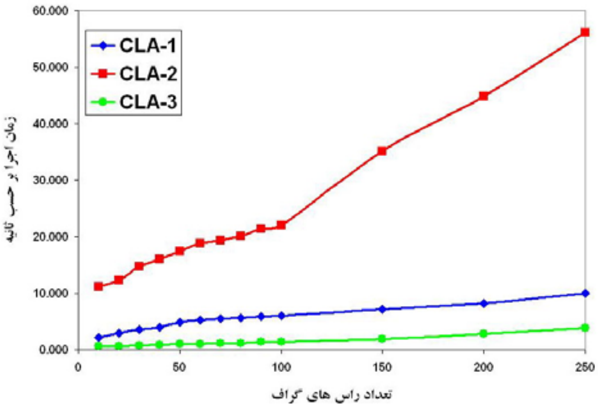
شکل ۸- مقایسه الگوریتم پیشنهادی اول با الگوریتم‌های سهنی، ژئومنس، ترکیبی و ژنتیک بر اساس زمان اجرا بر حسب ثانیه



شکل ۹- مقایسه الگوریتم پیشنهادی دوم با الگوریتم‌های سهنی، ژئومنس، ترکیبی و ژنتیک بر اساس زمان اجرا بر حسب ثانیه



شکل ۱۰- مقایسه الگوریتم پیشنهادی سوم با الگوریتم‌های سهنی، ژئومنس، ترکیبی و ژنتیک بر اساس زمان اجرا بر حسب ثانیه



شکل ۱۱- مقایسه سه الگوریتم پیشنهادی با همدیگر بر اساس زمان اجرا بر حسب ثانیه

۷- نتیجه‌گیری

در این مقاله با استفاده از اتوماتای یادگیر سلولی نامنظم یک الگوریتم مبتنی بر اتوماتای یادگیر سلولی جدید برای حل مساله بزرگترین برش گراف پیشنهاد و کارایی آن با الگوریتم‌های تقریبی سهنی و ژئومنس، الگوریتم‌های ترکیبی و ژنتیک و الگوریتمهای مبتنی بر اتوماتای یادگیر سلولی مقایسه گردید. طبق نتایج به‌دست آمده الگوریتم‌های پیشنهادی نتایج بهتری را نسبت به الگوریتم‌های پیشین تولید کرده است.

مراجع

[1] Asnaashari, M. and Meybodi, M. R., "Irregular Cellular Learning Automata and Its Application to Clustering in Sensor Networks", Proceedings of 15th Conference on Electrical Engineering (15th ICEE), Volume on Communication, Telecommunication Research Center, Tehran, Iran, 2007.

[2] Barahona, F., Grotschel, M., Junger, M. and Reinelt, G. “An Application of Combinatorial Optimization to Statistical Physics and Circuit Layout Design”, Oper. Res., Vol. 36, pp. 493–513, 1988.

[3] Beigy, H. and Meybodi, M. R., “A Mathematical Framework for Cellular Learning Automata”, Advanced in Complex Systems, Vol. 7, No. 3&4, pp. 294-319, 2004.

[4] Beigy, H. and Meybodi, M. R., "Asynchronous Cellular Learning Automata" Automatica, Journal of International Federation of Automatic Control, 2007, to appear.

[5] Beigy, H. and Meybodi, M. R., “Asynchronous Cellular Learning Automata”, Proceedings of 10th Annual CSI Computer Conference Iran, Telecommunication Research Center, Tehran, Iran, pp. 271-280, 2005.

[6] Beigy, H. and Meybodi, M. R., "Open Synchronous Cellular Learning Automata", Advances in Complex Systems, 2007, to appear.

[7] Beigy, H. and Meybodi, M. R., “Open Synchronous Cellular Learning Automata”, Proceedings of the 8th world Multi-conference on Systemic, Cybernetics and Informatics(SCI2004), pp. 9-15, 2004.

[8] DIMACS directories. Second DIMACS implementation challenge. http://dimacs.rutgers.edu /Challenges/index.html.

[9] Dolezal, O., Hofmeister, T. and Lefmann, H., “A Comparison of Approximation Algorithms for the Maxcut-Problem”, Manuscript, Universitat Dortmund, Lehrstuhl Informatik 2, Dortmund, Germany. 1999.

[10] Festa, P., Pardalos, P. M., Resende, M. G. C. and Ribero, C. C., “Randomized Heuristics for the MAX-CUT Problem”, Optimization Methods and Software, Vol. 17, No. 6, pp. 1033-1058, 2002

[11] Goemans, M. X. and Williamson, D. P., “Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming”, Journal of Association for Computing Machinery, Vol. 42, No. 6, pp. 1115-1145, 1995.

[12] Haglin, D. J. and Venkatesan, S. M., “Approximation and Intractability Results for the Maximum Cut Problem and its Variants”, IEEE Trans. Comput., Vol. 40, pp. 110–113, 1991.

[13] Hofmeister, T. and Lefmann, H., “A Combinatorial Design Approach to MAXCUT,” Proceedings of the 13th Symposium on Theoretical Aspects of Computer Science, pp. 441-452, 1996.

[14] Karp, R., "Reducibility among Combinatorial Problems", Complexity of Computer Computations, pp. 85-104, 1972.

[15] Khurti, A., Bäck, Th. and Heitkötter, J., “An Evolutionary Approach to Combinatorial Optimization Problems”, Proceeding of 22nd Annual ACM Computer Science Conference, pp. 66-73, 1994.

[16] Meybodi, M. R., Beigy, H. and Taherkhani, M., "Cellular Learning Automata and Its Applications", Journal of Science and Technology, University of Sharif, No. 25, pp.54-77, Autumn/Winter 2003-2004.
[17] Meybodi, M. R. and Kharazmi, M. R., "Cellular Learning Automata and Its Application to Image Processing", Journal of Amirkabir, Vol. 14, No. 56A, pp. 1101-1126, 2004.
[18] Meybodi, M. R. and Mehdipour, F., "Application of Cellular Learning Automata with Input to VLSI Placement", Journal of Modarres, University of Tarbeit Modarres, Vol. 16, pp. 81-95, summer 2004.
[19] Narendra, K. S. and Thathacha, M. A. L., "Learning Automata: An Introduction", Prentice Hall, 1989.
[20] Poljak, S. and Turźik, D., "A Polynomial Algorithm for Constructing a Large Bipartite Subgraph With an Application to a Satisfiability Problem", Can. J. Math, Vol. 34, pp. 519–524, 1982.
[21] Sahni, S. and Gonzalez, T., "P-Complete Approximation Problems", Journal of ACM, Vol. 23, No. 3, pp. 555-565, 1976.
[22] Thathachar, M. A. L. and Sastry, P. S., "Varieties of Learning Automata: An Overview", IEEE Transaction on Systems, Man and Cybernetics-Part B: Cybernetics, Vol. 32, No. 6, pp. 711-722, 2002.
[23] Vitanyi, P. M., "How Well Can a Graph is n-Colored?", Disc. Math., Vol. 34, pp. 69-80, 1981.
[24] Wolfram, S., "Cellular Automata", Los Alamos Science, Vol. 9, pp. 2-21, 1983.
[25] Wolfram, S., "Universality and Complexity in Cellular Automata", Physica D., No. 10, pp. 1-35, 1984.
[26] Enayatzare, M. and Meybodi, M. R. "Solving Maximum Cut Problem using Cellular Learning Automata", Technical Report, Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran, 2007.