حل مسئله تناظرگراف توسط اتوماتانهای یادگیر*

حمید بیگی محمد رضا میبدی

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی امیر کبیر

تهران، ایران

(نسخه اولیه)

چکیده

مسئله تناظر گراف دارای کاربردهای بسیاری از جمله صحت مدارهای VLSI، صحت مدارهای PC- Board ، معادل بودن دو برنامه، شناسایی الگو و بینایی ماشین میباشد. مسئله تناظر گراف در گروه مسائل NP قرار دارد بدین معنی که تاکنون الگوریتمی با پیچیدگی زمانی چند جمله ای برای حل این مسئله گزارش نشده است. در بسیاری از کاربردها، گراف ها دارای اغتشاش میباشند وبطور دقیق با هم متناظر نیستند. در این حالت هدف پیدا کردن تناظریست که کمترین خطا را تولید نماید. این مسئله در گروه مسائل NP-Hard قرار دارد و دارای هزینه اجرایی از مرتبه نمایی میباشد. بدین دلیل محققین زیادی کوششهای فراوانی در جهت پیدا کردن الگوریتمی که دارای پیچیدگی زمانی کمتری باشد صرف کرده اند. در این راستا، الگوریتم های تقریبی زیادی از جمله از طریق پیدا نمودن بزرگترین Clique، روشهای Relaxation، جستجوی درخت، آنالیز جبری، شبکه های عصبی و الگوریتم های ژنتیکی برای حل مسئله تناظر دو گراف پیشنهاد شده است. این الگوریتم ها علیرغم تقریبی بودن جواب های قابل قبولی را تولید میکنند. با توجه به اینکه مسائل فوق ، در گروه مسائل NP-Search در بهینه سازی ترکیبی قرار دارد پیدا کردن یک ابزار جستجو برای کاهش پیچیدگی زمانی الگوریتم و تضمین صحت جواب امری لازم و ضروری میباشد. اتوماتانهای یادگیر یک ابزار جستجوی عمومی میباشند که برای حل تعدادی از مسائل NP-Complete از جمله افراز اشیا، افراز گراف و بهینه سازی صفحه کلید بکار برده شده است. اتوماتان یادگیر از نوع مهاجرت اشیا یکی از انواع اتوماتانهای یادگیر میباشد. در این گزارش، چهار اتوماتان مهاجرت اشیا برای حل مسئله تناظر دو گراف ارائه شده است. تا آنجایی که نگارندگان این گزارش آگاهی دارند تا کنون استفاده از اتوماتانهای یادگیر برای حل این مسئله گزارش نشده است. الگوریتم های ارائه شده روی گراف های با اندازه و درصد اغتشاش های متفاوت آزمایش شده اند و جواب های خوبی را تولید کرده اند. در حالتیکه گراف ها بدون اغتشاش هستند جواب های تولید شده (در صورت همگرایی) کاملا دقیق هستند. با توجه به نتایج شبیه سازیها، الگوریتم های ارائه شده دارای متوسط پیچیدگی زمانی از مرتبه $O(n^2)$ هستند که بهبود قابل ملاحظه ای در مرتبه بزرگی حل این مسئله نشان میدهد.

کلمات کلیدی : تناظردوگراف، تناظر گراف با تصحیح خطا، اتوماتانهای یادگیر، اتوماتانهای مهاجرت اشیا، مسائل ذاتا مشکل

۱- مقدمه

یک گراف بصورت زوج $G=(V, E)$ نمایش داده میشود که V مجموعه گرههای گراف و E مجموعه کمانهای آن میباشد. دوگراف $G_1 = (V_1, E_1)$ و $G_2 = (V_2, E_2)$ را متناظر میگویند اگر تابع یک‌به‌یک‌وپوشا $f: V_1 \rightarrow V_2$ وجود داشته باشد که برای هر کمان (u, w) موجود در گراف G_1 ، کمان $(f(u), f(w))$ در گراف G_2 وجود داشته باشد. کلاس پیچیدگی این مسئله مشخص نیست امامتعلق به گروه مسایل NP است ولی هنوز تعلق آن به گروه مسایل NP-Complete اثبات نشده است و تاکنون الگوریتمی با پیچیدگی زمانی چند جمله‌ای برای حل این مسئله گزارش نشده است. یک روش برای حل این مسئله استفاده از الگوریتم Backtracking میباشد که در بدترین حالت برای گراف باندازه n نیازبه بررسی O (n!) تناظردارد. مسئله تناظر گراف[1] دارای کاربردهای بسیاری است که بعضی از آنها عبارتنداز: صحت مدارهای VLSI [۱]-[۱۰]، صحت مدار های PC-Board [۶]، معادل بودن دو

*قسمتی از این پروژه با حمایت مالی مرکز تحقیقات فیزیک نظری- پژوهشکده سیستم های هوشمند انجام گردیده است.

1- Graph Isomorphism

برنامه [۱۷]، شناسایی الگو[۱۸] و بینایی ماشین [۳][۲۱]. مشلادر کاربردهای بینایی ماشین، شی والگوتوسط دوگراف مختلف نشان داده میشوند. پس از تبدیل اشیابه گرافهای متناظر، مسئله تشخیص اشیابه پیدا کردن تناظربین دوگراف تبدیل میشود. درشناسایی اشیاممکن است دوگراف هم اندازه نباشند. دراین حالت هدف بزرگترین زیرگرافی درگراف بزرگتراست بطوریکه این زیرگراف باگراف کوچکتر متناظرباشد. به این مسئله، تناظرزیر گراف[1] میگویند. این مسئله متعلق به گروه مسایل NP-Hard میباشد و دارای پیچیدگی از مرتبه نمایی میباشد.

در بسیاری از کاربردها، گراف ها دارای اغتشاش[2] میباشند و دو گراف بطور دقیق با هم متناظر نمیباشند. یعنی ممکن است هم تعداد گره های دو گراف و هم کمانهای آنها با هم متفاوت باشند. در اینحالت، هدف پیدا نمودن تناظری است که کمترین خطا را تولید نماید که به این تناظر، تناظر بهینه میگویند و به مسئله پیدا کردن تناظر بهینه دو گراف غیر متناظر، تناظر گراف با تصحیح خطا[3] گفته میشود[۳][۲۱]. این مسئله درگروه مسائل NP-Hard قرار دارد و دارای پیچیدگی از مرتبه نمایی میباشد.

با توجه به اینکه مسائل فوق در گروه مسائل دانا مشکل[4] قرار دارند و زمان حل آنها با افزایش اندازه مسئله بصورت نمایی افزایش پیدا میکند. لذا هدف، پیدا کردن الگوریتمی است که دارای پیچیدگی زمانی کمتری باشد. در این راستا، الگوریتم های بسیاری پیشنهاد شده اند. هر چند این الگوریتم ها تقریبی[5] هستند اما جواب های قابل قبولی را در مقایسه با الگوریتم های دقیق تولید میکنند. بعضی از این الگوریتم ها عبارتند از: پیدا کردن تناظر از طریق پیدا نمودن بزرگترین Clique[۲۰]، روشهای Relaxation [۲۱]، جستجوی درخت[۴]، آنالیز جبری[۲۰]، شبکه های عصبی [۲] ، الگوریتم های ژنتیکی [۷][۲۰] و الگوریتم های تصادفی [۲۲].

با توجه به اینکه مسایل فوق ، در گروه مسایل NP-Search در بهینه سازی ترکیبی[6] قرار دارد پیدا کردن یک ابزار جستجو برای کاهش پیچیدگی زمانی الگوریتم و تضمین صحت جواب (پیدا کردن جواب بهینه) امری لازم وضروری میباشد. اتوماتانهای یادگیر یک ابزار جستجوی عمومی میباشد که برای حل تعدادی از مسایل NP-Complete از جمله افراز اشیا[۱۴]، افراز گراف [۱۵]، بهینه سازی صفحه کلید[۱۶] و پیدا کردن ساختاربهینه شبکه عصبی [۲۳] بکار برده شده است. اتوماتان یادگیر از نوع مهاجرت اشیا[7] یک نوع از اتوماتانهای یادگیر میباشد. در این اتوماتان، اشیا روی وضعیت های مختلف اتوماتان قرار میگیرند و با حرکت اشیا (در این گزارش، گره های گراف) تناظر های مختلف ایجاد میشود و با جستجو در فضای ایجاد شده میتوان تناظر بهینه را پیدا نمود. در این گزارش، چهار اتوماتان یادگیر از مهاجرت اشیا برای حل مسئله تناظر دو گراف ارائه شده است.

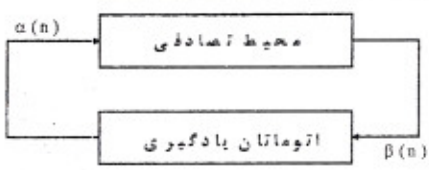
تا آنجایی که نگارندگان این گزارش اگاهی دارند تا کنون استفاده از اتوماتانهای یادگیر برای حل این مسئله گزارش نشده است. الگوریتم های ارائه شده روی گراف های با اندازه و درصد اغتشاش های متفاوت آزمایش شده اند و جواب های بسیار قابل قبولی را تولید کرده اند. در حالتیکه گراف ها بدون اغتشاش هستند جواب های تولید شده (در صورت همگرایی) کاملا دقیق هستند. با توجه به نتایج شبیه سازیها، الگوریتم های ارائه شده دارای پیچیدگی زمانی متوسطی از مرتبه $O(n^3)$ هستند که بهبود قابل ملاحظه ای را در مرتبه بزرگی حل این مسئله نشان میدهد.

بخش های بعدی مقاله بصورت زیر سازماندهی شده است. اتوماتانهای یادگیر در بخش ۲ آمده است. در بخش ۳ اتوماتانها و الگوریتم های پیشنهادی برای تناظر دو گراف ارائه شده است. در بخش ۴ نتایج شبیه سازیها و در پایان نتیجه گیری آمده است.

۲- اتوماتانهای یادگیر

یادگیری در اتوماتانهای یادگیر، انتخاب یک اقدام[8] بهینه از میان یک مجموعه از اقدام های مجاز اتوماتان میباشد. این اقدام روی یک محیط تصادفی اعمال میشود و محیط به این اقدام اتوماتان بوسیله یک پاسخ تصادفی از مجموعه پاسخ های مجاز جواب میدهد. پاسخ محیط بصورت آماری به اقدام اتوماتان وابسته است. اصطلاح محیط شامل اجتماع تمام شرایط خارجی و تاثیرات آنها روی عملکرد اتوماتان میباشد.

یک محیط بصورت یک سه تایی $\langle \underline{\alpha}, \underline{C}, \underline{\beta} \rangle$ نشان داده میشود. مجموعه، $\underline{\alpha}=\{\alpha_1, \ldots, \alpha_r\}$ مجموعه ورودیها، مجموعه $\underline{C}=\{c_1, \ldots, c_r\}$ مجموعه احتمالات (c_i احتمال شکست اقدام α_i میباشد) و مجموعه $\underline{\beta}=\{\beta_1, \beta_2\}$ خروجی دودویی محیط میباشد[۱۳]. اتصال یک اتوماتان با محیط در شکل ۱ نشان داده شده است.



شکل ۱: اتصال اتوماتان یادگیری با محیط

اتوماتان های یادگیری به دو خانواده اتوماتان یادگیری با ساختار ثابت[9] و اتوماتان یادگیری با ساختار متغیر[10] دسته بندی میشوند. اتوماتا نهای ***کرینسکی***[11] ***و کرایلوف***[12] مثالهایی از خانواده اتوماتانهای با ساختار ثابت هستند. یک اتوماتان یادگیری با ساختار ثابت را میتوان با یک پنج تایی $\langle \underline{\alpha}, \underline{\Phi}, \underline{\beta}, F, G \rangle$ نشان داد. که $\underline{\alpha}=\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ مجموعه اقدام های مجاز برای اتوماتان یادگیر، $\underline{\Phi}=\{\Phi_1, \Phi_2, \ldots, \Phi_s\}$ مجموعه وضعیت های اتوماتان، $\underline{\beta}=\{0, 1\}$ مجموعه ورودیها (در این

1- Subgraph Isomorphism
2- Noise
3- Error-Correcting Graph Isomorphism
4- Intractable Problems
5- Approximate Algorithms
6-Combinatorial Obtimization
7- Object Migrating Automata (OMA)

8- Action
9- Fixed Structure Learning Automata (FSLA)
10- Variable Structure Learning Automata (VSLA)
11- Krinsky Automata
12- Krylov Automata

مجموعه **یک** نمایانگر شکست[1] و **صفر** نمایانگر موفقیت[2] میباشد)، F: $\Phi \times \beta \to \Phi$ تابع نگاشت وضعیت ها و $G: \Phi \to \alpha$ تابع نگاشت خروجی میباشد.

اقدام اتوماتان بعنوان ورودی به محیط داده میشود و محیط بعد از اعمال اقدام داده شده توسط اتوماتان یک پاسخ تصادفی که میتواند نشاندهنده موفقیت یا شکست اقدام اعمال شده باشد را تولید میکند که بعنوان ورودی به اتوماتان داده میشود. اتوماتان با توجه به پاسخ محیط اقدام مربوطه را جریمه میکند و یا به آن پاداش میدهد. اگر احتمال تغییر وضعیت ها در اتوماتان ثابت باشد آنرا اتوماتان یادگیری با ساختار ثابت و در غیر اینصورت آنرا اتوماتان یادگیری با ساختار متغیر مینامند.

اتوماتانهای یادگیر دارای کاربردهای فراوانی میباشد بعضی از این کاربردها عبارتند از: مسیریابی در شبکه‌های ارتباطی[۹]، فشرده سازی تصاویر[۵]، شناسایی الگو [۱۹]، برنامه‌ریزی فرایندها[3] در یک شبکه کامپیوتری[۱۳]، تئوری صف [۱۲] ، کنترل دسترسی در شبکه‌های انتقال ناهمزمان[4] [۱۲]، کمک به آموزش شبکه های عصبی [۱۱] ، دسته بندی و افراز اشیاء [۱۴] و پیدا کردن ساختار بهینه برای شبکه های عصبی [۲۳].

برای یک گراف با اندازه n، n! نگاشت مختلف وجود دارد و در صورتیکه از اتوماتانهای یادگیر برای پیدا نمودن تناظر دو گراف استفاده شود اتوماتان باید n! اقدام داشته باشد که تعداد زیاد اقدام ها سرعت همگرایی اتوماتان را کم میکند بهمین جهت اتوماتان مهاجرت اشیاء[5] توسط *اومن*[6] و *ما*[7] پیشنهاد شده است[۱۴].

اتوماتان مهاجرت اشیاء: اتوماتان مهاجرت اشیاء توسط پنج تایی $\langle \underline{\alpha}, \underline{\Phi}, \underline{\beta}, F, \underline{G} \rangle$ نشان داده میشود[۱۴]. $\underline{\alpha}=\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ مجموعه اقدام های مجاز، $\underline{\Phi}=\{\Phi_1, \Phi_2, \ldots, \Phi_s\}$ مجموعه وضعیت های اتوماتان، $\underline{\beta}=\{0, 1\}$ مجموعه ورودیها، $F: \Phi \times \beta \to \Phi$ تابع نگاشت وضعیت ها و $G: \Phi \to \alpha$ تابع نگاشت خروجی میباشند. این نوع اتوماتان برای دسته بندی اشیاء[۱۴]، انتساب حروف به کلیدها[۱۵] و افراز گراف[8] [۱۶] مورد استفاده قرار گرفته است.

در این اتوماتان هر اقدام یک دسته را نشان میدهد. در اتوما تا نهای با ساختار ثابت پاسخ محیط به اتوماتان سبب میشود که اتوماتان از یک وضعیت به وضعیت دیگر منتقل شود در صورتیکه در اتوماتان مهاجرت اشیاء، اشیاء به وضعیت ها انتساب داده میشوند و پاسخ محیط به اتوماتان سبب گردش اشیاء در بین وضعیت های اتوماتان میگردد و از طریق این گردش دسته بندی اشیاء انجام میگیرد.

اگر شی W_i درخروجی r ام اتوماتان مهاجرت اشیاء قرار داشته باشد این شی متعلق به دسته شماره r است. برای خروجی α_k مجموعه وضعیت $\{\Phi_{(k-1)N+1}, \ldots, \Phi_{kN}\}$ در نظر گرفته میشود که N عمق حافظه را نشان میدهد. بدون از دست دادن عمومیت بحث میتوان $\Phi_{(k-1)N+1}$ را داخلی ترین وضعیت و Φ_{kN} راخارجی ترین وضعیت این خروجی در نظر گرفت. اگر دو شی W_i و W_j بترتیب در وضعیت های $\Phi_{(k-1)N+1}$ و $\Phi_{(k-1)N+m}$ (برای $m > 1$) قرار داشته باشند در اینصورت قطعیت تعلق شی W_i به این دسته از قطعیت تعلق شی W_j بیشتر است. بنابر این برای خروجی α_k، وضعیت $\Phi_{(k-1)N+1}$، وضعیت با بیشترین قطعیت و وضعیت Φ_{kN}، وضعیت با کمترین قطعیت نامیده میشود.

۳- اتوماتان یادگیر تعیین تناظر بین دو گراف

در این قسمت چهار اتوماتان از نوع مهاجرت اشیاء برای حل مسئله تناظر دو گراف معرفی میگردد. وظیفه این اتوماتانها، دسته بندی گره های گراف است بطوریکه با این دسته بندی کمترین خطا تولید گردد. در ابتدا به روشهای ساختن گراف های تصادفی و چگونگی محاسبه اختلاف بین دو گراف میپردازیم و در ادامه به ارائه اتوماتانهای پیشنهادی میپردازیم.

۳-۱- ساختن گرافهای تصادفی

معمولا برای ساختن یک گراف تصادفی با تعداد گره های مشخص از یکی از دوروش زیر استفاده میشود. روش اول از یک عدد تصادفی برای مشخص نمودن وجود یا عدم وجود یک کمان در گراف استفاده میکند. روش دوم با یک تعداد کمان مشخص شروع میکند و بصورت تصادفی این کمانها را بین گره های مختلف قرار میدهد. در این مقاله از روش اول برای ساختن گرافها استفاده شده است.

یک گراف با وزن و جهت دار G را میتوان بصورت زوج $G = (V(G), W(G))$ نشان داد که در آن $V(G)$ مجموعه گره های گراف و $W(G)$ ماتریس همجواری گراف میباشد. درصورتیکه $|W_{ij}(G)| > 0$ باشد کمانی از گره i به گره j با وزن $W_{ij}(G)$ وجود دارد. گراف های بدون جهت ، دارای ماتریس همجواری، متقارن میباشد. جایگشت های مختلف[9] سطرها (در نتیجه ستونهای متناظر) در ما تریس همجواری باعث تغییر ترتیب برچسب گره های گراف میگردد. بنابر این میتوان گفت دو گراف $G = (V(G), W(G))$ و $H = (V(H), W(H))$ متناظر هستند اگر وتنها اگر ترتیب برچسب گره های گراف H یک جایگشت از برچسب گره های گراف G باشد یعنی دو ماتریس همجواری $W(G)$ و $W(H)$ از طریق رابطه زیر بهم نگاشت داده میشوند[۸].

$$W(H) = P \cdot W(G) \cdot P^T \qquad (۱)$$

که P ماتریس جایگشت میباشد. در هر سطر(ستون) این ماتریس تنها یک عدد ۱ وجود دارد و بقیه عناصر این سطر (ستون) صفر هستند. عضو P_{ji} از ماتریس P نشاندهنده نگاشت گره V_i از گراف G به گره V_j از گراف H میباشد. براساس مقدار ماتریس P، نگاشت $\sigma: V(G) \to V(H)$ بصورت زیر تعریف میگردد.

$$\sigma = \left\{ (v_i, v_j) \,\middle|\, P_{ji} = 1,\ v_i \in V(G),\ v_j \in V(H) \right\} \qquad (۲)$$
$$\forall\ i, j = 1, 2, \ldots, n$$

برای مثال اگر نگاشت σ بصورت $\sigma = \{(A, b), (B, d), (C, c), (D, a)\}$ باشد ماتریس جایگشت بصورت زیر خواهد شد.

1- Unfavorable
2- Favorable
3- Process
4- Asynchronous transfer mode (ATM)
5- Object Migrating Automata (OMA)
6- Oommen
7- Ma
8- Graph Partitioning
9- Permutation

$$P = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

با استفاده از نگاشت σ ، خطای نگاشت دو گراف G و H را بصورت زیر تعریف میکنیم.

$$J(\sigma) = \left\| W(H) - P \cdot W(G) \cdot P^T \right\| \qquad (۳)$$

که $\| \cdot \|$ نرم ماتریس(هر نوع نرم) میباشد. بنابراین مسئله پیدا کردن تناظر بین دو گراف به حل معادله $J(\sigma) = 0$ تبدیل میشود. زمانیکه دو گراف بطور دقیق با هم متناظر نباشند (تناظر دو گراف با تصحیح خطا) مسئله پیدا کردن تناظر بین دو گراف به پیدا کردن کمینه تابع $J(\sigma)$ تبدیل میگردد. برای گراف با اندازه n ، محاسبه $J(\sigma)$ نیاز به زمانی از مرتبه $\theta(n^3)$ دارد. برای کم کردن زمان محاسبه $J(\sigma)$ ، میتوان نشان داد که تحت نگاشت σ، رابطه زیر برقرار است.

$$\left[P \cdot W(G) \cdot P^T\right]_{i,j} = [W(G)]_{\sigma(i),\ \sigma(j)} \qquad (۴)$$

که $[A]_{ij}$ عضو سطر i و ستون j ماتریس A میباشد. اگر در رابطه (۳) از نرم L_1 ($\|W\| = \sum_i \sum_j |W_{i,j}|$) استفاده کنیم شکل ساده تری برای محاسبه مقدار اختلاف بین دو گراف G و H بدست میاید. برای این منظور خطای نگاشت گره k از گراف G به گره σ(k) از گراف H بصورت زیر تعریف میشود.

$$J_k(\sigma) = \sum_{m=1}^{n} \left| [W(H)]_{k,m} - [W(G)]_{\sigma(k),\ \sigma(m)} \right| + \sum_{m=1}^{n} \left| [W(H)]_{m,k} - [W(G)]_{\sigma(m),\ \sigma(k)} \right| \qquad (۵)$$

درصورتیکه گراف بدون جهت باشد خطای نگاشت گره k برابر است با

$$J_k(\sigma) = 2 \times \sum_{m=1}^{n} \left| [W(H)]_{k,m} - [W(G)]_{\sigma(k),\ \sigma(m)} \right| \qquad (۶)$$

ودرنتیجه خطای تناظر دو گراف برای نگاشت σ بصورت زیر تعریف میگردد.

$$J(\sigma) = \sum_{k=1}^{n} J_k(\sigma) \qquad (۷)$$

با توجه به معادلات فوق روشن است که محاسبه $J(\sigma)$ نیاز به زمانی از مرتبه $\theta(n^3)$ دارد.

۳-۲- اتوماتانهای یادگیر برای تناظر دو گراف

اتوماتان یادگیر برای تعیین تناظر گراف را میتوان بصورت یک شش تایی $\langle \underline{V}, \underline{\alpha}, \underline{\Phi}, \underline{\beta}, E, \underline{G} \rangle$ نشان داد که در آن

۱- $\underline{V} = \{V_1, V_2, \ldots, V_n\}$ مجموعه گره های گراف $G=(V_G, W_G)$ میباشد. این گره ها روی وضعیت های مختلف اتوماتان حرکت میکنند و با حرکت خود تناظرهای مختلف را ایجاد میکنند.

۲- $\underline{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_K\}$ مجموعه اقدام های مجاز برای اتوماتان یادگیر است. این اتوماتان K اقدام دارد(تعداد اقدام های این اتوماتان با تعداد گرههای گراف برابر است) اگر گره u از گراف G در اقدام m قرار گرفته باشد در اینصورت گره u ازگراف G با گره w_m از گراف H متناظر است.

۳- $\underline{\Phi} = \{\Phi_1, \Phi_2, \ldots, \Phi_{KN}\}$ مجموعه وضعیت ها و N عمق حافظه برای اتوماتان میباشد. مجموعه وضعیت های این اتوماتان به K زیر مجموعه $\{\Phi_1, \ldots, \Phi_N\}$ ، $\{\Phi_{N+1}, \ldots, \Phi_{2N}\}$ و ... و $\{\Phi_{(K-1)N+1}, \ldots, \Phi_{KN}\}$ افراز میشود وگره های گراف براساس اینکه در کدام وضعیت قرار داشته باشند دسته بندی میگردند. اگر گره u از گراف G در مجموعه وضعیت های $\{\Phi_{(j-1)N+1}, \ldots, \Phi_{jN}\}$ قرار داشته باشد در اینصورت گره u ازگراف G با گره w_j از گراف H متناظر است. در مجموعه وضعیت های اقدام j، وضعیت $\Phi_{(j-1)N+1}$ را وضعیت داخلی و وضعیت Φ_{jN} را وضعیت مرزی مینامیم و گرهی را که در وضعیت $\Phi_{(j-1)N+1}$ قرار دارد را گره با اهمیت بیشتر و گرهی را که در وضعیت Φ_{jN} قرار دارد را با اهمیت کمتر نامیده میشود.

۴- $\underline{\beta} = \{0, 1\}$ مجموعه ورودیهای اتوماتان میباشد. در این مجموعه **یک** نمایانگر شکست و **صفر** نمایانگر موفقیت میباشد.

۵- $F: \Phi \times \beta \rightarrow \Phi$ تابع نگاشت وضعیت ها میباشد. این تابع از روی وضعیت فعلی و ورودی اتوماتان وضعیت بعدی آنرا تولید می نماید. در واقع این تابع چگونگی گردش گره های گراف رادر وضعیت های اتوماتان مشخص میکند. این تابع برای اتوماتانهای مختلف متفاوت میباشد که شرح کارکرد آن برای اتوماتانهای مختلف در قسمت بعد خواهد آمد.

۶- $G: \Phi \rightarrow \alpha$ تابع نگاشت خروجی میباشد. این تابع تصمیم میگیرد که به ازای هر وضعیت ، اتوماتان چه اقدامی را انجام دهد. در اگر اینصورت گره u ازگراف G در مجموعه وضعیت های $\{\Phi_{(j-1)N+1}, \ldots, \Phi_{jN}\}$ قرار داشته باشددر اینصورت اقدام j انتخاب میگردد(گره u ازگراف G با گره w_j از گراف H متناظر است).

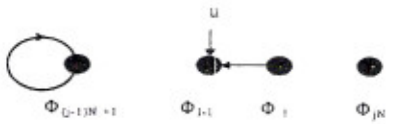
در ادامه شرح کارکرد تابع نگاشت وضعیت برای اتوماتانهای مختلف بیان میگردد.

۳-۲-۱: اتوماتان مهاجرت اشیا با اتصال های مشابه اتوماتان Tsetline

این اتوماتان از اتصال های مشابه اتوماتان Tsetline [۱۳] برای حالت پاداش استفاده میکند و برای سهولت در ارائه مطالب، اتوماتان با K اقدام ، عمق حافظه N که برای تناظر دو گراف با اندازه n استفاده شده است توسط Tsetline-GI(K, N, n) نمایش داده میشود. برای تشریح تابع نگاشت وضعیت ها دو حالت زیر را در نظر میگیریم.

۱- اگرگره u ازگراف G در یکی از وضعیت های $\{\Phi_{(j-1)N+1}, \ldots, \Phi_{jN}\}$ قرار داشته باشد و خطای تناظر بین گره u ازگراف G و گره w_j از گراف H ($J_u(\sigma)$) از مقدار آستانه (مقدار آستانه بصورت تطبیقی مشخص میگردد و مقدارآن در هر لحظه برابر است با میانگین خطای نگاشت برای تمامی گره ها) کوچکتر باشد به این گره پاداش داده میشود و اهمیت این گره بیشتر شده وبه سمت وضعیتهای داخلی تر این اقدام حرکت میکند. نحوه حرکت چنین گرهی درشکل زیر نشان داده شده است.

الف) وضعیت گره قبل از گرفتن پاداش



ب) وضعیت گره پس از گرفتن پاداش

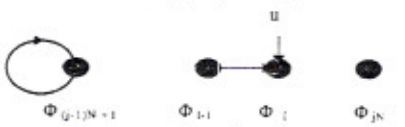
شکل ۲: نحوه دادن پاداش به یک گره

اگرگره u در وضعیت $\Phi_{(j-1)N+1}$ قرار داشته باشد و پاداش بگیرد در همان وضعیت باقی میماند.

۲-اگر گره u ازگراف G بـا درمجموعـه وضعیـت هـای $\{\Phi_{(j-1)N+1}, \ldots, \Phi_{jN}\}$ قـرار داشـته باشـد و $J_u(\sigma)$ از مقـدار آسـتانه بزرگـتر باشـد در اینصورت تناظر برقرار شده مناسب نیست و ایـن گـره جریمـه میشود نحوه حرکت چنین گرهی برای دو حالت مختلف در زیر آمده است.

الف) گره u ازگراف G در مجموعه وضعیت های $\{\Phi_{(j-1)N+1}, \ldots, \Phi_{jN-1}\}$ قرار دارد . جریمه نمودن این گـره سـبب کـم شـدن اهمیت ایـن گـره میشود. نحوه حرکت چنین گرهی در شکل زیر نشان داده شده است.

الف) وضعیت گره قبل از جریمه شدن

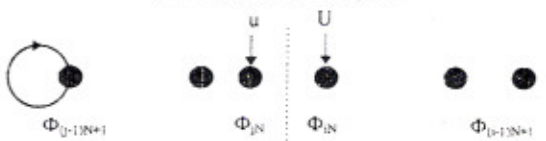


ب) وضعیت گره پس از جریمه شدن

شکل۳ : نحوه جریمه شدن یک گره

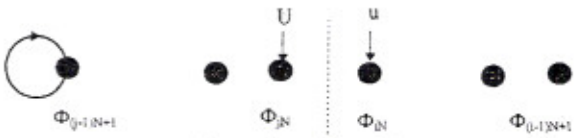
ب) گرهu ازگرافG در وضعیت Φ_{jN} قرار دارد. دراین حالت گرهU ازگـراف G راپیـدا میکنیـم بطوریکهاگردرتناظرت σ جـای u و U عـوض شـوند بیشـترین کاهش درمقدارخطاحاصل گردد. دراینصورتاگرگـرهU در وضعیتمرزیقرارداشتهباشدجای u و U عوض میشودو درغیر اینصـورت ابتدا U به وضعیت مرزی اقدام خود منتقـل و سـپس جابجـایی صورت میگیرد. نحوه حرکت چنین گرهی در شکل ۵ نشان داده شده است.

الف) وضعیت گره قبل از جریمه شدن



ب) انتقال گره U به وضعیت مرزی

شکل ۴: نحوه جریمه شدن یک گره



ج) وضعیت گره پس از جریمه شدن

شکل ۴ (ادامه): نحوه جریمه شدن یک گره

با استفاده از اتوماتان فوق الگوریتمی بصورت زیر بـرای تنـاظر دو گـراف ارائه شده است.

```
procedure Reward (u)
  if (State (u) - 1) mod N <> 0 then
    dec (State (u))
  end if
end Reward

procedure Penalize (u)
  if State (u) mod N <> 0 then
    inc (State (u))
  else
    bestError = ∞
    for U = 1 to |V_G| do
      Create mapping σ' from σ by sawpping u and U
      if J_U(σ') < bestError then
        bestError = J_U(σ')
        bestNode = U
      end if
    end for
    State (bestNode) = Action (bestNode) * N
    Swap (State (u), State (bestNode))
  end if
end Penalize

function GraphIsomorphism (G, H)
  Create an automatan Tsetline-GI (|V_G|, N, |V_G|)
  Create a random mapping σ, such that every node u ∈ V_G is
  assigned into one action of Tsetline-GI (|V_G|, N, |V_G|)
  repeat
    for m = 1 to n do
      J_m(σ) = 2 × Σ_{i=1}^{c} |W_G(σ(k), σ(m)) - W_G(k, m)|
    end for
    u = Select (random (M))    // random number in range [1,
    M]
    if J_u(σ) < T then
      Reward (u)
    else
      Penalize (u)
    end if
  until J (σ) = 0
  return σ
end GraphIsomorphism
```

شکل۵: الگوریتم تناظر گراف بااستفاده ازاتوماتان یادگیر Tsetline-GI (K,N,n)

تابع Select (m)، برچسب گره ای (u) را بعنـوان جـواب برمیگردانـد کـه مقـدار خطـای نگاشـت ایـن گـره m ($J_u(\sigma)$) امیـن مقـدار بـزرگ را در مجموعه خطاهای نگاشت ها دارا باشد. این تابع بصورت زیر میباشد.

```
function Select (m)
  Select node u such that J_u(σ) is mth largest element in the set
  [J_1(σ), J_2(σ), . . ., J_n(σ)]
  return u
end Select
```

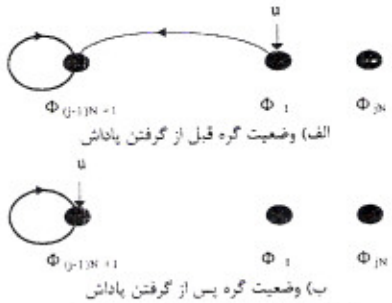
شکل ۶: الگوریتم انتخاب m امین خطای بزرگ

یک راه خوب برای افزایش احتمال همگرایی الگوریتم تناظر گراف این است که اگر اندازه مسئله از مقدار مشخصی کوچکتر باشد بجای انتخاب تصادفی یک گره، از الگوریتم قطعی برای انتخاب استفاده گردد. در الگوریتم های ارائه شده در این گزارش، گرهی انتخاب میگردد که بیشترین خطا را تولید نماید.

۳-۲-۲: اتوماتان مهاجرت اشیا با اتصال های مشابه اتوماتان Krinsky

این اتوماتان از اتصال های مشابه اتوماتان Krinsky [۱۳] برای حالت پاداش استفاده میکند و برای سهولت در ارائه مطالب، اتوماتان با K اقدام ، عمق حافظه N که برای تناظر دو گراف با اندازه n استفاده شده است توسط Krinsky-GI(K, N, n) نام گذاری شده است. برای تشریح تابع نگاشت وضعیت ها دو حالت زیر را در نظر میگیریم.

۱- اگرگره u ازگراف G در مجموعه $\{\Phi_{(j-1)N+1}, ..., \Phi_{jN}\}$ قرار داشته باشد و $J_u(\sigma)$ از مقدار آستانه ای کوچکتر باشد به این گره پاداش داده میشود. نحوه حرکت چنین گرهی در شکل زیر نشان داده شده است.



شکل ۷: نحوه دادن پاداش به یک گره

اگرگره u در وضعیت $\Phi_{(j-1)N+1}$ قرار داشته باشد و پاداش بگیرد در همان وضعیت باقی میماند. شکل زیر الگوریتم دادن پاداش به گره u را برای این اتوماتان نشان میدهد.

```
procedure Reward (u)
  State (u) = (Action (u) - 1) * N
end Reward
```

شکل ۸: نحوه دادن پاداش به گره u

۲-اگر گره u ازگراف G در مجموعه وضعیت های $\{\Phi_{(j-1)N+1}, ..., \Phi_{jN}\}$ قرار داشته باشد و $J_u(\sigma)$ از مقدار آستانه ای بزرگتر باشد در اینصورت گره u جریمه میشود. نحوه جریمه شدن این گره همانند جریمه شدن گره در اتوماتان Tsetline-GI (K, N, n) میباشد.

۳-۲-۳: اتوماتان مهاجرت اشیا با اتصال های مشابه اتوماتان Krylov

اتصال های این اتوماتان درحالت موفقیت مشابه اتصالهای اتوماتان Krylov [۱۳] برای حالت شکست میباشد. برای سهولت در ارائه مطالب، اتوماتان با K اقدام ، عمق حافظه N که برای تناظر دو گراف با اندازه n استفاده شده است توسط Krylov-GI(K, N, n) نمایش داده میشود. برای تشریح تابع نگاشت وضعیت ها دو حالت زیر را در نظر میگیریم.

۱-اگر گره u ازگراف G در مجموعه وضعیت های $\{\Phi_{(j-1)N+1}, ..., \Phi_{jN}\}$ قرار داشته باشد و $J_u(\sigma)$ از مقدار آستانه بزرگتر باشد در اینصورت گره u جریمه میشود نحوه جریمه شدن این گره مانند جریمه شدن گره در اتوماتان Tsetline-GI (K, N, n) میباشد.

۲- اگرگره u ازگراف G در وضعیت Φ_1 از مجموعه $\{\Phi_{(j-1)N+1}, ..., \Phi_{jN}\}$ قرار داشته باشد و $J_u(\sigma)$ از مقدار آستانه ای کوچکتر باشد به این گره پاداش داده میشود. اگر گره u در وضعیت $\Phi_{(j-1)N+1}$ قرار داشته باشد با احتمال ۰/۵ به وضعیت $\Phi_{(j-1)N+2}$ و باحتمال ۰/۵ در همین وضعیت باقی میماند. اگر گره u در وضعیت Φ_{jN} قرار داشته باشد با احتمال ۰/۵ به وضعیت Φ_{jN-1} و باحتمال ۰/۵ جریمه میشود. در غیر اینصورت گره u با احتمال ۰/۵ به وضعیت‌بعدو با احتمال ۰/۵ به وضعیت قبل منتقل میشود. درشکل‌زیرالگوریتم‌دادن پاداش به گره u نشان داده شده است.

```
procedure Reward (u)
  if random > 0.5 then
    if (State (u) - 1) mod N <> 0 then
      dec (State (u))
    end if
  else
    if State (u) mod N <> 0 then
      inc (State (u))
    else
      Penalize (u)
    end if
  end if
end Reward
```

شکل ۹ : نحوه دادن پاداش به گره u

۳-۲-۴- اتوماتان مهاجرت اشیا AMO[1]

دراین‌اتوماتان، درهنگام‌پاداش‌یاجریمه‌همه واحدهادراتوماتان‌حرکت میکنند. یک‌اتوماتان‌AMO با K اقدام‌وعمق‌حافظه‌N که‌برای‌تناظر گراف‌باندازه n استفاده‌شده‌است‌توسط AMO-GI(K, N, n) نمایش‌داده میشود. برای‌تشریح‌تابع‌نگاشت‌وضعیت‌هادوحالت‌زیر را در نظر میگیریم.

۱- اگر گره u ازگراف G در وضعیت Φ_1 قرار داشته باشد و$J_u(\sigma)$ از مقدار آستانه کوچکتر باشد به این نگاشت پاداش داده میشود و همه گره ها بسمت وضعیت های داخلی تر حرکت میکنند. اگر گرهی در داخلی ترین وضعیت اقدام خود باشد در همان وضعیت باقی میماند. الگوریتم حرکت گره ها در شکل زیر آمده است.

```
procedure Reward (u)
  for U = 1 to |V_G| do
    if (State (U) - 1) mod N <> 0 then
      dec (State (U))
    end if
  end for
end Reward
```

شکل ۱۰: پاداش دادن به همه گره ها

۲- اگرگره u ازگراف G دروضعیت Φ_1 قرارداشته‌باشد و $J_u(\sigma)$ ازمقدار آستانه‌کوچکتر‌باشداین‌نگاشت‌جریمه‌میشودوهمه‌گره‌هابسمت‌وضعیت های مرزی‌حرکت‌میکند. این‌حرکت آنقدرادامه‌پیدا میکندتاحداقل‌یک گره‌دروضعیت مرزی قرار گیرد. در اینصورت حالت گره U از گراف G را به صورتی پیدا میکنیم که اگر در نگاشت‌ u‌جای u و U عوض شوند بیشترین کاهش درمقدار خطاحاصل گردد. دراینصورت اگرگره U دروضعیت مرزی قرار داشته باشد جای u و U عوض میشود و در غیر

1- All Moving Objects (AMO)

اینصورت ابتدا U به وضعیت مرزی اقدام خود منتقل و سپس جابجایی صورت میگیرد. نحوه حرکت‌چنین گرهی‌درشکل‌زیرنشانداده شده است.

```
procedure Penalize (u)
  repeat
    for U = 1 to |V_G| do
      if State (U) mod N <> 0 then
        inc (State (U))
      end if
    end for
  until at least one node appears in the boundary state
  bestError = ∞
  for U = 1 to |V_G| do
    Create mapping σ' from σ by sawpping u and U
    if J_U(σ') < bestError then
      bestError = J_U(σ')
      bestNode = U
    end if
  end for
  State (bestNode) = Action (bestNode) * N
  State (u) = Action (u) * N
  Swap (State (u), State (bestNode))
end Penalize
```

شکل ۱۱ : نحوه جریمه کردن گره u

۳-۳ : استفاده از اتوماتانهای یادگیر از نوع مهاجرت اشیا برای تناظر دو گراف با تصحیح خطا: همه الگوریتم های فوق را میتوان برای گراف های دارای اغتشاش بکار برد تنها با این تفاوت که شرط پایان را باید بصورتی مشخص نماییم که زمان اجرای الگوریتم پس از پیدا نمودن تناظر بهینه محدود باشد. با توجه به اینکه اتوماتانهای یادگیر یک ابزار جستجوی عمومی میباشند بنابراین احتمال گرفتاری آنها در حداقل های محلی بسیار کم (و در بیشتر موارد غیر ممکن) میباشد زیرا از اطلاعات محلی برای پیدا نمودن جواب بهینه استفاده نمیکند. بنابراین اگر مدت زیادی الگوریتم نتواند جواب را تغییر دهد این جواب بمنزله جواب بهینه است. شکل زیر الگوریتم پیدا نمودن تناظر دو گراف با تصحیح خطا را نشان میدهد.

```
function ErrorCorrectingGraphIsomorphism (G, H)
  Create an automatan Tsetline-GI (|V_G|, N, |V_G|)
  Create a random mapping σ, such that every node u ∈ V_G is
  assigned into one action of Tsetline-GI (|V_G|, N, |V_G|)
  repeat
    for m = 1 to n do
      J_m(σ) = 2 * Σ_{i=1}^{n} [W_G(σ(k), σ(m)) - W_G(k, m)]
    end for
    u = Select (random (M))    // random number in range [1, M]
    if J_u(σ) < T then
      Reward (u)
    else
      Penalize (u)
    end if
  until J (σ) = 0 or for a long period J (σ) is constant
end ErrorCorrectingGraphIsomorphism
```

شکل ۱۲ : الگوریتم تناظر دو گراف با تصحیح خطا

۴- نتایج شبیه سازیها

در این بخش الگوریتم های فوق روی گراف های مختلف و با اندازه های متفاوت آزمایش شده اند و نتایج آن درجداول زیر آمده است. در این گزارش از روش اول برای ساختن گراف تصادفی با وزن و بدون جهت G استفاده شده است که در آن اعداد تصادفی در فاصله [۰، ۹۹] بعنوان وزن هر کمان اختصاص داده میشود و وزن صفر بمعنای عدم وجود کمان میباشد. با تغییر تصادفی برچسب n گره از گراف G، گراف H تولید میگردد. در تمامی آزمایشهای زیر، هر الگوریتم روی ۱۰ گراف یکسان اجرا شده است تا توزیع زمان اجرای الگوریتم ها مشخص گردد. مقدار متوسط زمان اجرای الگوریتم ها و تعداد اجراهایی که همگرا نشده اند در جدولهای ۱ الی ۴ آورده شده است. در این آزمایشها، مقدار آستانه بصورت تطبیقی مشخص میگردد و مقدار آن در هر لحظه برابر است با میانگین خطای نگاشت برای همه گره ها. برای تولید گرا فهای دارای اغتشاش، وزن هر کمان را با عددی تصادفی در فاصله [-X, X] جمع میشود که X درصد اغتشاش میباشد.

جدول ۱: تناظر دو گراف برای ده اجرای مختلف توسط Tsetline-GI (K, N, n)

عمق اندازه گراف	متوسط تعداد تکرار الگوریتم					تعداد اجراهای همگرا نشده				
	۱	۳	۵	۷	۹	۱	۳	۵	۷	۹
۲۰۰	۸۰	۱۳۱	۱۸۹	۱۰۱	۱۱۱	۰	۰	۰	۱	۱
۳۰	۱۶۶	۱۳۴	۱۰۷	۱۱۲	۸۷	۰	۰	۰	۰	۰
۴۰	۷۰	۱۷۹	۹۵	۱۴۶	۱۴۴	۰	۰	۰	۰	۰
۵۰	۵۳	۷۹	۸۹	۱۱۲	۱۲۹	۰	۰	۰	۰	۰
۶۰	۶۹	۸۰	۱۱۳	۱۳۰	۱۵۰	۰	۰	۰	۰	۰
۷۰	۸۹	۱۱۴	۱۷۴	۱۶۸	۲۱۵	۰	۱	۰	۰	۰
۸۰	۸۴	۱۱۴	۱۴۹	۱۶۱	۱۹۹	۰	۰	۰	۰	۰
۹۰	۹۵	۱۳۳	۱۵۵	۱۸۳	۲۲۷	۰	۰	۰	۰	۰
۱۰۰	۹۸	۱۴۹	۱۸۵	۲۱۶	۲۷۸	۰	۰	۰	۰	۰
۱۱۰	۱۱۹	۱۶۱	۲۱۳	۲۴۱	۲۸۳	۰	۰	۰	۰	۰
۱۲۰	۱۰۰	۱۴۴	۱۹۳	۲۴۱	۲۶۲	۰	۰	۰	۰	۰

جدول۲ : تناظر دو گراف برای ده اجرای مختلف توسط AMO -GI (K, N, n)

عمق اندازه گراف	متوسط تعداد تکرار الگوریتم					تعداد اجراهای همگرا نشده				
	۱	۳	۵	۷	۹	۱	۳	۵	۷	۹
۲۰	۱۸۱	۱۱۳	۱۳۲	۱۷۵	۱۷۴	۰	۰	۲	۲	۲
۳۰	۴۹۳	۲۱۳	۱۵۴	۱۳۹	۱۱۹	۰	۰	۰	۰	۰
۴۰	۳۴	۳۸	۳۴	۳۴	۳۹	۰	۰	۰	۰	۰
۵۰	۳۵	۳۶	۳۶	۳۵	۳۶	۰	۰	۰	۰	۰
۶۰	۵۷	۵۶	۶۱	۵۹	۵۹	۰	۰	۰	۰	۰
۷۰	۸۴	۸۰	۸۴	۸۸	۸۰	۰	۰	۰	۰	۰
۸۰	۷۲	۷۷	۷۶	۷۵	۸۳	۰	۰	۰	۱	۰
۹۰	۱۱۰	۱۰۰	۱۲۴	۱۰۶	۱۰۴	۰	۰	۰	۰	۰
۱۰۰	۹۲	۹۴	۹۴	۹۵	۹۴	۰	۰	۰	۰	۰
۱۱۰	۹۵	۹۷	۹۷	۹۴	۹۵	۰	۰	۰	۰	۰
۱۲۰	۱۰۲	۱۰۲	۱۰۲	۱۰۱	۱۰۰	۰	۰	۰	۰	۰

جدول ۳: تناظر دو گراف برای ده اجرای مختلف توسط Krinsky -GI (K, N, n)

عمق اندازه گراف	متوسط تعداد تکرار الگوریتم					تعداد اجراهای همگرا نشده				
	۱	۳	۵	۷	۹	۱	۳	۵	۷	۹
۲۰	۷۲	۶۰	۶۳	۱۲۶	۶۷	۰	۰	۰	۰	۰
۳۰	۲۸۸	۱۳۸	۱۴۹	۲۱۷	۲۵۲	۰	۰	۰	۰	۰
۴۰	۴۵	۶۱	۷۱	۸۷	۱۰۱	۰	۰	۰	۰	۰
۵۰	۳۸۳	۴۸۸	۵۳۷	۵۰۱	۷۱۶	۰	۰	۰	۰	۰
۶۰	۵۳	۷۴	۹۲	۱۱۶	۱۳۲	۰	۰	۰	۰	۰
۷۰	۴۳۶	۱۶۸	۱۶۸	۳۱۱	۵۸۱	۰	۰	۰	۰	۰
۸۰	۳۰۲	۲۶۳	۴۳۶	۳۸۲	۲۸۶	۱	۱	۰	۱	۱
۹۰	۱۰۵	۱۴۶	۱۶۳	۲۰۲	۲۳۰	۰	۰	۰	۰	۰
۱۰۰	۱۱۰	۱۵۹	۳۴۸	۲۵۴	۲۷۹	۰	۰	۰	۰	۱
۱۱۰	۹۶	۱۳۹	۱۷۶	۲۲۵	۲۶۱	۰	۰	۰	۰	۰
۱۲۰	۱۰۱	۱۴۸	۱۸۵	۲۲۰	۲۶۲	۰	۰	۰	۰	۰

جدول ۴: تناظر دو گراف برای ده اجرای مختلف توسط Krylov-GI (K, N, n)

عمق اندازه گراف	متوسط تعداد تکرار الگوریتم					تعداد اجراهای همگرا نشده				
	۱	۳	۵	۷	۹	۱	۳	۵	۷	۹
۲۰	۱۶۲	۱۹۴	۱۴۴	۱۷۲	۱۹۴	۲	۱	۴	۱	۰
۳۰	۹۸	۳۷۹	۱۷۹	۲۰۰	۳۱۶	۰	۰	۰	۰	۰
۴۰	۳۷۰	۳۲۹	۴۰۸	۱۵۶	۲۱۹	۰	۰	۱	۰	۰
۵۰	۵۷۵	۳۱۱	۶۱۰	۶۱۳	۱۲۸۷	۱	۰	۱	۴	۲
۶۰	۶۴	۹۸	۱۲۶	۱۶۲	۱۷۷	۰	۰	۰	۰	۰
۷۰	۴۴۲	۳۰۸	۵۱۰	۳۰۰	۶۳۶	۱	۰	۲	۲	۳
۸۰	۱۲۰	۱۴۸	۲۲۵	۲۶۹	۲۹۹	۰	۰	۰	۰	۱
۹۰	۸۸	۱۲۴	۱۵۲	۱۸۸	۲۲۱	۰	۰	۰	۰	۰
۱۰۰	۱۸۴	۴۳۹	۳۲۳	۵۹۹	۵۵۴	۳	۱	۰	۱	۰
۱۱۰	۱۹۸	۲۰۸	۲۶۴	۲۶۵	۳۱۵	۰	۰	۱	۰	۰
۱۲۰	۹۸	۱۳۹	۱۹۲	۲۲۴	۲۵۶	۰	۰	۰	۰	۰

جداول ۵ تا ۱۶ متوسط تعداد تکرار و تعداد اجرا های همگرا نشده و خطا پس از همگرایی الگوریتم تناظر دو گراف با تصحیح خطـا بـرای اغتشـاش هـای مختلف را برای اتوماتانهای مختلف نشان میدهند.

جدول ۵: تناظر دو گراف با تصحیح خطا برای اغتشاش ۵٪ توسط Tsetline-GI (K,N,n)

عمق اندازه گراف	خطا پس از همگرایی	متوسط تعداد تکرار الگوریتم					تعداد اجراهای همگرا نشده				
		۱	۳	۵	۷	۹	۱	۳	۵	۷	۹
۲۰	۱۹۲۶	۹۵	۹۳	۱۴۷	۱۲۷	۱۶۲	۰	۰	۰	۰	۰
۳۰	۴۲۱۶	۱۶۴	۱۸۱	۱۷۰	۲۲۰	۱۴۲	۰	۰	۰	۰	۰
۴۰	۷۷۳۰	۱۹۲	۱۳۴	۳۳۸	۱۴۶	۳۶۵	۰	۰	۰	۰	۰
۵۰	۱۱۹۶۴	۲۹۱	۲۳۰	۱۵۷	۳۳۱	۶۱۶	۰	۱	۰	۰	۰
۶۰	۱۷۴۲۲	۱۱۸	۱۴۳	۱۶۴	۱۸۲	۲۱۷	۰	۰	۰	۰	۰
۷۰	۲۴۰۰۸	۵۵۰	۷۷۲	۱۱۶۷	۱۰۸۴	۶۲۲	۴	۴	۴	۳	۲
۸۰	۳۱۳۶۲	۱۱۴	۱۴۳	۱۶۷	۱۹۰	۲۱۶	۰	۰	۰	۰	۰
۹۰	۳۹۳۲۴	۴۳۴	۲۲۵	۲۷۲	۳۳۹	۳۴۹	۰	۰	۰	۰	۰
۱۰۰	۴۹۲۱۸	۱۳۶	۱۷۶	۲۱۲	۲۴۴	۲۸۵	۰	۰	۰	۰	۰
۱۱۰	۵۸۷۱۰	۱۵۰	۱۹۲	۲۲۵	۲۷۳	۳۱۱	۰	۰	۰	۰	۰
۱۲۰	۷۰۱۲۶	۲۱۴	۲۴۶	۲۸۶	۳۳۰	۴۴۸	۰	۰	۰	۰	۰

جدول۶ : تناظر دو گراف با تصحیح خطا برای اغتشاش ۵٪ توسط AMO -GI (K,N,n)

عمق اندازه گراف	خطا پس از همگرایی	متوسط تعداد تکرار الگوریتم					تعداد اجراهای همگرا نشده				
		۱	۳	۵	۷	۹	۱	۳	۵	۷	۹
۲۰	۱۸۵۴	۱۹۸	۹۴	۱۰۸	۱۵۱	۱۰۰	۲	۰	۰	۰	۰
۳۰	۴۲۵۲	۲۲۸	۴۱۷	۲۱۵	۳۳۹	۲۰۴	۰	۰	۰	۰	۰
۴۰	۷۷۶۰	۳۷۳	۲۱۳	۶۶۶	۷۶۳	۲۵۶	۰	۰	۰	۱	۰
۵۰	۱۲۰۶۶	۱۰۹	۱۲۳	۲۷۴	۱۱۱	۱۰۸	۰	۰	۰	۰	۰
۶۰	۱۷۶۷۶	۱۰۵	۱۰۷	۱۰۳	۱۰۴	۱۰۷	۰	۰	۰	۰	۰
۷۰	۲۳۸۴۸	۱۴۲	۱۶۰	۱۴۹	۱۷۳	۲۲۱	۰	۰	۰	۰	۰
۸۰	۳۰۷۲۶	۱۲۱	۱۲۰	۱۲۱	۱۲۰	۱۲۳	۰	۰	۰	۰	۰
۹۰	۳۹۴۴۰	۱۵۴	۱۵۸	۱۵۷	۱۵۲	۱۴۹	۰	۰	۰	۰	۰
۱۰۰	۴۸۸۶۶	۱۴۹	۱۴۶	۱۴۸	۱۵۷	۱۴۶	۰	۰	۰	۰	۰
۱۱۰	۵۹۴۰۲	۱۷۲	۱۸۵	۱۸۵	۱۸۵	۱۸۴	۰	۰	۰	۰	۰
۱۲۰	۷۰۱۵۴	۱۶۵	۱۶۴	۱۶۵	۱۶۵	۱۶۵	۰	۰	۰	۰	۰

جدول ۷: تناظر دو گراف با تصحیح خطا برای اغتشاش ۵٪ توسط Krinsky -GI (K,N,n)

عمق اندازه گراف	خطا پس از همگرایی	متوسط تعداد تکرار الگوریتم					تعداد اجراهای همگرا نشده				
		۱	۳	۵	۷	۹	۱	۳	۵	۷	۹
۲۰	۱۷۸۲	۶۳	۷۲	۸۲	۸۶	۹۶	۰	۰	۰	۰	۰
۳۰	۴۲۸۴	۳۲۹	۳۹۸	۳۳۷	۳۹۶	۳۸۰	۰	۰	۰	۰	۰
۴۰	۷۵۶۰	۱۰۲	۳۰۰	۱۳۲	۳۳۱	۱۵۹	۰	۰	۰	۰	۰
۵۰	۱۲۱۹۲	۹۶۴	۱۰۵۶	۸۲۷	۷۴۶	۸۶۴	۲	۰	۰	۱	۰
۶۰	۱۷۳۰۰	۱۳۸	۱۹۲۲	۱۸۰	۱۸۱	۲۳۰	۰	۰	۰	۰	۰
۷۰	۲۴۱۵۴	۱۰۸	۱۳۲	۱۴۶	۱۷۳	۲۰۹	۰	۰	۰	۰	۰
۸۰	۳۱۴۵۲	۱۷۳	۲۳۰	۲۲۰	۳۳۱	۲۹۷	۱	۰	۰	۱	۰
۹۰	۳۹۳۴۲	۱۴۷	۱۹۰	۲۳۴	۲۴۵	۳۰۷	۰	۰	۰	۰	۰
۱۰۰	۴۹۰۴۸	۱۶۸	۲۶۶	۳۳۴	۳۱۴	۴۸۶	۰	۲	۰	۰	۰
۱۱۰	۵۹۵۰۲	۱۴۷	۱۸۸	۲۲۷	۲۶۵	۲۸۹	۰	۰	۰	۰	۰
۱۲۰	۷۱۰۱۶	۱۴۷	۱۹۳	۲۲۹	۲۶۶	۳۱۰	۰	۰	۰	۰	۰

جدول ۸: تناظر دو گراف با تصحیح خطا برای اغتشاش ۵٪ توسط Krylov-GI (K, N, n)

عمق اندازه گراف	خطا پس از همگرایی	متوسط تعداد تکرار الگوریتم					تعداد اجراهای همگرا نشده				
		۱	۳	۵	۷	۹	۱	۳	۵	۷	۹
۲۰	۱۷۵۸	۷۶	۱۰۱	۱۱۳	۱۱۰	۱۲۸	۰	۰	۰	۰	۰
۳۰	۴۱۸۶	۳۴۹	۳۴۷	۳۰۹	۱۶۸	۵۴۳	۰	۰	۰	۰	۰
۴۰	۷۷۱۲	۸۶	۱۰۲	۱۲۲	۱۳۴	۱۴۵	۰	۰	۰	۰	۰
۵۰	۱۱۹۹۸	۱۰۵	۱۵۲	۳۶۷	۱۷۳	۳۱۴	۰	۰	۰	۰	۰
۶۰	۱۷۶۵۴	۱۲۲	۱۶۶	۲۱۹	۳۶۶	۴۸۷	۱	۰	۲	۰	۰
۷۰	۲۳۶۷۰	۷۰۱	۲۴۵	۲۵۴	۲۵۲	۴۴۹	۱	۲	۰	۱	۰
۸۰	۳۱۴۱۸	۳۱۰	۲۹۶	۲۷۴	۳۲۳	۴۱۳	۱	۱	۱	۱	۰
۹۰	۳۹۴۵۰	۱۱۵۷	۲۶۹	۴۱۸	۷۲۴	۴۷۵	۲	۱	۴	۲	۳
۱۰۰	۴۸۷۳۸	۲۶۸	۶۱۱	۵۲۹	۴۹۴	۵۲۲	۱	۲	۵	۲	۳
۱۱۰	۵۸۸۹۴	۱۹۹	۲۷۴	۵۱۲	۳۷۳	۵۷۰	۱	۰	۱	۰	۰
۱۲۰	۷۰۲۱۲	۱۶۹	۲۱۵	۲۵۵	۲۹۴	۳۴۹	۰	۰	۰	۰	۰

جدول ۹: تناظر دو گراف با تصحیح خطا برای اغتشاش ۱۰٪توسط Tsetline-GI (K, N, n)

عمق اندازه گراف	خطا پس از همگرایی	متوسط تعداد تکرار الگوریتم					تعداد اجراهای همگرا نشده				
		۱	۳	۵	۷	۹	۱	۳	۵	۷	۹
۲۰	۳۸۳۴	۱۵۳	۲۵۸	۲۲۰	۲۵۶	۲۱۵	۰	۰	۰	۰	۱
۳۰	۸۵۵۲	۸۶	۹۴	۱۰۵	۱۳۰	۱۴۴	۰	۰	۰	۰	۰
۴۰	۱۵۴۱۶	۶۸۶	۹۱۲	۹۵۴	۱۰۹۸	۱۱۴۶	۰	۰	۱	۰	۰
۵۰	۳۴۱۳۲	۱۴۵	۲۱۸	۱۵۳	۱۶۹	۱۹۷	۰	۰	۰	۰	۰
۶۰	۳۴۲۷۸	۱۷۶۸	۱۶۰۰	۱۰۵۲	۲۵۹۲	۱۴۶۹	۳	۶	۳	۴	۵
۷۰	۴۷۵۷۰	۱۳۲	۱۵۹	۱۸۲	۲۲۴	۲۶۲	۰	۰	۰	۰	۰
۸۰	۶۲۲۹۸	۱۳۲	۱۶۴	۱۹۲	۲۱۷	۲۶۴	۰	۰	۰	۰	۰
۹۰	۷۸۲۰۰	۱۶۷	۲۳۰	۲۳۴	۲۸۶	۲۹۸	۰	۰	۰	۰	۰
۱۰۰	۹۷۳۸۸	۲۱۰	۲۱۲	۲۶۷	۳۴۳	۳۶۷	۱	۰	۰	۰	۰
۱۱۰	۱۱۷۷۷۸	۱۹۲	۲۳۷	۲۷۱	۳۲۳	۳۵۲	۰	۰	۰	۰	۰
۱۲۰	۱۳۹۸۲۸	۱۷۶	۲۲۰	۲۷۳	۳۲۷	۳۴۸	۰	۰	۰	۰	۰

جدول ۱۰ : تناظر دو گراف با تصحیح خطا برای اغتشاش ۱۰٪ توسط AMO -GI (K, N, n)

عمق اندازه گراف	خطا پس از همگرایی	متوسط تعداد تکرار الگوریتم					تعداد اجراهای همگرا نشده				
		۱	۳	۵	۷	۹	۱	۳	۵	۷	۹
۲۰	۳۷۹۲	۹۲	۱۶۶	۱۸۱	۱۹۲	۱۹۴	۰	۰	۰	۰	۰
۳۰	۸۸۴۰	۱۰۶	۱۴۳	۱۵۵	۱۲۸	۱۲۶	۰	۰	۰	۰	۰
۴۰	۱۴۹۹۸	۸۰۳	۸۶۲	۳۹۲	۳۷۶	۳۹۲	۰	۰	۰	۰	۰
۵۰	۲۳۹۶۲	۱۱۷۱	۱۰۳۳	۹۹۲	۶۸۵	۶۹۴	۱	۴	۱	۱	۱
۶۰	۳۴۵۷۰	۵۰۲	۸۳۳	۷۴۹	۴۲۶	۹۴۳	۲	۱	۱	۰	۲
۷۰	۴۷۲۳۴	۱۵۰	۱۵۹	۱۳۶	۱۴۶	۱۴۹	۱	۰	۰	۰	۰
۸۰	۶۲۲۳۸	۱۳۸	۱۳۲	۱۳۱	۱۳۸	۱۵۰	۰	۰	۰	۰	۰
۹۰	۷۸۱۹۶	۶۵۱	۵۸۳	۳۱۴	۴۴۴	۳۵۹	۲	۰	۱	۰	۱
۱۰۰	۹۶۸۵۴	۱۳۷	۱۳۷	۱۳۷	۱۳۶	۱۳۹	۰	۰	۰	۰	۰
۱۱۰	۱۱۷۱۷۸	۲۹۲	۲۸۸	۲۶۲	۱۹۹	۲۵۷	۰	۰	۰	۱	۰
۱۲۰	۱۴۰۰۵۴	۱۸۲	۱۷۸	۱۷۴	۱۷۸	۱۷۷	۰	۰	۰	۰	۰

جدول ۱۱ : تناظر دو گراف با تصحیح خطا برای اغتشاش ۱۰٪توسط Krinsky -GI (K,N, n)

عمق اندازه گراف	خطا پس از همگرایی	متوسط تعداد تکرار الگوریتم					تعداد اجراهای همگرا نشده				
		۱	۳	۵	۷	۹	۱	۳	۵	۷	۹
۲۰	۳۶۷۰	۱۴۳	۲۰۱	۱۴۸	۱۱۱	۲۳۵	۰	۰	۰	۱	۰
۳۰	۸۷۶۲	۴۸۴	۳۲۴	۵۱۰	۴۵۷	۴۳۴	۰	۰	۰	۰	۰
۴۰	۱۴۸۵۶	۱۵۶	۲۱۲	۵۵۲	۱۶۴	۱۹۴	۰	۰	۰	۰	۰
۵۰	۲۴۳۴۰	۶۴۸	۱۴۷	۱۶۷	۱۸۱	۲۰۵	۰	۰	۰	۰	۰
۶۰	۳۴۳۱۸	۱۹۹	۱۲۱۴	۶۱۸	۱۱۰۰	۸۸۹	۰	۱	۰	۱	۱
۷۰	۴۸۰۱۴	۲۴۶	۲۲۴	۲۱۹	۳۰۷	۹۱۰	۰	۰	۱	۰	۰
۸۰	۶۲۹۷۸	۱۵۲	۱۸۵	۲۱۳	۲۳۰	۲۸۸	۰	۰	۰	۰	۰
۹۰	۷۸۰۹۴	۵۵۸	۳۰۵	۶۳۶	۶۰۵	۵۳۹	۱	۲	۰	۰	۱
۱۰۰	۹۶۹۷۰	۱۷۱	۲۲۶	۶۰۳	۲۹۵	۳۲۶	۰	۰	۰	۰	۰
۱۱۰	۱۱۷۲۶۰	۱۹۶	۲۴۳	۲۸۶	۴۰۲	۳۷۶	۰	۰	۰	۰	۰
۱۲۰	۱۳۸۹۵۰	۱۶۲	۲۰۵	۲۵۸	۳۰۰	۳۵۲	۰	۰	۰	۰	۰

جدول ۱۲ : تناظر دو گراف با تصحیح خطا برای اغتشاش ۱۰٪ توسط Krylov-GI (K, N, n)

عمق اندازه گراف	خطا پس از همگرایی	متوسط تعداد تکرار الگوریتم					تعداد اجراهای همگرا نشده				
		۱	۳	۵	۷	۹	۱	۳	۵	۷	۹
۲۰	۳۸۱۰	۱۸۵	۱۸۱	۳۴۸	۱۵۹	۳۰۸	۱	۰	۲	۰	۰
۳۰	۸۵۸۰	۱۵۸	۲۵۲	۲۵۴	۱۷۵	۳۷۰	۰	۰	۰	۰	۰
۴۰	۱۵۶۰۲	۵۰۸	۷۹۵	۳۷۳	۶۳۴	۶۲۶	۰	۰	۰	۰	۰
۵۰	۲۳۵۶۸	۹۳۸	۱۴۷۸	۶۰۶	۷۳۵	۱۱۹۲	۱	۱	۰	۰	۲
۶۰	۳۴۱۵۶	۳۸۶	۳۱۱	۲۵۵	۵۵۲	۱۴۴۳	۰	۰	۳	۱	۱
۷۰	۴۷۲۴۴	۱۱۱	۱۳۶	۱۶۱	۱۸۵	۲۰۶	۰	۰	۰	۰	۰
۸۰	۶۱۶۵۲	۱۶۱	۱۸۴	۲۰۳	۲۴۲	۲۶۰	۰	۰	۰	۰	۰
۹۰	۷۷۹۱۸	۱۴۰	۱۷۱	۲۰۸	۲۴۱	۲۸۶	۰	۰	۰	۰	۰
۱۰۰	۹۷۲۶۶	۳۹۱	۷۶۹	۷۶۰	۴۵۲	۵۴۶	۲	۰	۲	۰	۲
۱۱۰	۱۱۵۸۷۰	۱۷۷	۲۱۶	۲۶۹	۲۸۸	۳۳۰	۰	۰	۰	۰	۰
۱۲۰	۳۸۲۴۰	۱۶۱	۲۰۹	۲۵۱	۳۱۰	۳۳۳	۰	۰	۰	۰	۰

جدول ۱۳: تناظر دو گراف با تصحیح خطا برای اغتشاش ۱۵٪توسط Tsetline-GI (K, N, n)

عمق اندازه گراف	خطا پس از همگرایی	متوسط تعداد تکرار الگوریتم					تعداد اجراهای همگرا نشده				
		۱	۳	۵	۷	۹	۱	۳	۵	۷	۹
۲۰	۵۵۲۶	۱۵۳	۲۴۲	۱۸۰	۱۳۳	۱۴۳	۰	۰	۰	۰	۰
۳۰	۱۲۷۵۴	۲۴۷	۵۵۹	۴۳۹	۲۸۳	۴۰۳	۰	۰	۰	۰	۰
۴۰	۲۲۶۹۶	۸۶۱	۵۴۱	۲۴۲	۸۲۸	۷۳۸	۰	۰	۰	۰	۰
۵۰	۳۵۹۰۶	۱۰۹	۱۰۶	۱۰۵	۱۰۴	۱۱۳	۰	۰	۰	۰	۰
۶۰	۵۱۱۰۶	۲۳۸	۱۰۷۰	۱۲۹	۱۹۴	۵۹۳	۱	۰	۱	۰	۰
۷۰	۶۹۵۱۰	۱۴۳	۱۵۰	۱۶۸	۱۹۲	۴۵۳	۰	۰	۰	۰	۰
۸۰	۹۱۸۶۸	۴۸۱	۳۷۷	۳۲۱	۸۹۹	۷۸۹	۱	۳	۲	۲	۲
۹۰	۱۱۵۹۲۶	۱۴۶	۱۴۶	۱۴۷	۱۵۲	۱۳۸	۰	۰	۰	۰	۰
۱۰۰	۱۴۴۴۸۰	۲۲۷	۱۹۰	۲۶۱	۱۹۳	۲۳۴	۰	۰	۰	۰	۰
۱۱۰	۱۷۴۳۷۰	۳۶۲	۳۴۱	۳۵۷	۳۳۳	۳۱۶	۱	۰	۱	۱	۰
۱۲۰	۲۰۹۲۶۲	۱۹۶	۱۹۶	۱۹۳	۲۹۳	۱۹۳	۰	۰	۰	۰	۰

جدول۱۴ : تناظر دو گراف با تصحیح خطا برای اغتشاش ۱۵٪ توسط AMO -GI (K, N, n)

عمق اندازه گراف	خطا پس از همگرایی	متوسط تعداد تکرار الگوریتم					تعداد اجراهای همگرا نشده				
		۱	۳	۵	۷	۹	۱	۳	۵	۷	۹
۲۰	۵۵۲۶	۱۵۳	۲۴۲	۱۸۰	۱۳۳	۱۴۳	۰	۰	۰	۰	۰
۳۰	۱۲۷۵۴	۲۴۷	۵۵۹	۴۳۹	۲۸۳	۴۰۳	۰	۰	۰	۰	۰
۴۰	۲۲۶۹۶	۸۶۱	۵۴۱	۲۴۲	۸۲۸	۷۳۸	۰	۰	۰	۰	۰
۵۰	۳۵۹۰۶	۱۰۹	۱۰۶	۱۰۵	۱۰۴	۱۱۳	۰	۰	۰	۰	--
۶۰	۵۱۱۰۶	۲۳۸	۱۰۷۰	۱۲۹	۱۹۴	۵۹۳	۱	۰	۱	۰	۱
۷۰	۶۹۵۱۰	۱۴۳	۱۵۰	۱۶۸	۱۹۲	۴۵۳	۰	۰	۰	۰	۰
۸۰	۹۱۸۶۸	۴۸۱	۳۷۷	۳۲۱	۸۹۸	۷۸۹	۱	۳	۲	۲	۲
۹۰	۱۱۵۹۲۶	۱۴۹	۱۴۶	۱۴۷	۱۵۲	۱۳۸	۰	۰	۰	۰	۰
۱۰۰	۱۴۴۴۸۰	۲۲۷	۱۹۰	۲۶۱	۱۹۳	۲۳۴	۰	۰	۰	۰	۰
۱۱۰	۱۷۴۳۷۰	۳۶۲	۳۴۱	۳۵۷	۳۳۳	۳۱۶	۱	۰	۱	۱	۰
۱۲۰	۲۰۹۲۶۲	۱۹۶	۱۹۶	۱۹۲	۲۱۳	۱۹۳	۰	۰	۰	۰	۰

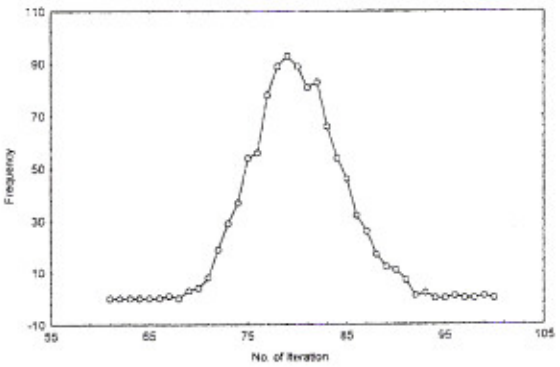
جدول ۱۵: تناظر دو گراف با تصحیح خطا برای اغتشاش ۱۵٪ توسط Krinsky -GI (K,N,n)

عمق اندازه گراف	خطا پس از همگرایی	متوسط تعداد تکرار الگوریتم					تعداد اجراهای همگرا نشده				
		۱	۳	۵	۷	۹	۱	۳	۵	۷	۹
۲۰	۵۴۶۶	۱۲۱	۱۶۴	۲۴۵	۱۰۹	۱۸۶	۰	۰	۰	۰	۰
۳۰	۱۲۴۱۰	۹۰	۱۱۶	۱۱۵	۱۳۱	۲۷۵	۰	۰	۰	۰	۰
۴۰	۲۲۳۰۸	۴۸۷	۹۴۳	۲۳۰	۸۶۱	۱۰۲۲	۰	۰	۰	۰	۰
۵۰	۳۵۸۱۲	۲۰۹۱	۱۷۲۴	۱۲۱۸	۱۹۱۴	۱۰۵۲	۱	۰	۱	۰	۰
۶۰	۵۱۶۵۸	۱۲۲	۱۴۲	۱۶۳	۱۹۱	۲۰۶	۰	۰	۰	۰	۰
۷۰	۷۰۰۳۰	۹۰۰	۱۵۵۴	۱۷۲۸	۱۱۵۲	۹۷۶	۴	۶	۲	۲	۳
۸۰	۹۱۵۵۶	۸۳۸	۳۶۳	۶۰۶	۱۱۴۶	۴۳۳	۳	۴	۲	۲	۱
۹۰	۱۱۵۰۳۲	۲۴۱	۲۳۶	۲۹۱	۲۹۶	۳۱۴	۰	۰	۰	۰	۰
۱۰۰	۱۴۴۷۱۰	۵۰۷	۴۴۲	۸۴۳	۵۳۸	۶۱۶	۲	۰	۴	۲	۶
۱۱۰	۱۷۴۷۲۴	۱۰۹۴	۳۲۸	۴۲۱	۵۳۰	۶۸۳	۵	۴	۴	۴	۴
۱۲۰	۲۰۸۲۶۶	۳۶۰	۲۹۷	۲۷۸	۶۴۶	۴۲۵	۰	۱	۰	۰	۰

جدول ۱۶: تناظر دو گراف با تصحیح خطا برای اغتشاش ۱۵٪ توسط Krylov-GI (K, N, n)

عمق اندازه گراف	خطا پس از همگرایی	متوسط تعداد تکرار الگوریتم					تعداد اجراهای همگرا نشده				
		۱	۳	۵	۷	۹	۱	۳	۵	۷	۹
۲۰	۵۵۱۴	۶۶	۷۲	۸۲	۹۳	۱۰۱	۰	۰	۰	۰	۰
۳۰	۱۲۶۳۴	۶۴۲	۳۶۶	۳۸۶	۶۱۸	۲۵۹	۰	۰	۰	۰	۰
۴۰	۲۲۳۳۰	۸۳۵	۹۰۶	۷۲۱	۳۹۹	۸۶۵	۰	۰	۰	۰	۰
۵۰	۳۵۴۶۸	۵۷۴	۱۰۰۳	۵۹۰	۳۰۳	۴۴۶	۰	۰	۰	۰	۰
۶۰	۵۱۸۱۲	۵۳۹	۳۸۵	۸۵۵	۳۲۵	۷۲۱	۰	۱	۲	۲	۱
۷۰	۶۹۷۹۶	۳۴۶	۶۷۱	۴۶۱	۵۰۸	۶۰۴	۲	۲	۱	۳	۰
۸۰	۹۳۲۵۲	۲۱۳	۸۶۵	۲۵۳	۲۸۲	۳۴۹	۱	۱	۳	۱	۰
۹۰	۱۱۶۰۲۰	۴۱۸	۱۲۵۹	۱۶۳۴	۲۷۷۷	۱۰۷۶	۳	۶	۳	۵	۴
۱۰۰	۱۴۴۷۰۲	۱۵۴	۱۹۲	۲۲۶	۲۶۰	۳۰۰	۰	۰	۰	۰	۰
۱۱۰	۱۷۴۶۸۶	۱۵۵	۱۹۱	۲۲۸	۲۷۶	۳۱۱	۰	۰	۰	۰	۰
۱۲۰	۲۰۸۰۲۶	۱۹۱	۲۳۳	۲۸۶	۳۳۹	۳۹۵	۰	۰	۰	۰	۰

تذکر ۱: تعداد تکرارهای حلقه repeat-until در الگوریتم های فوق دارای توزیع شکل زیر می باشد. شکل زیر توزیع تعداد تکرار های حلقه repeat-until را برای الگوریتم Tsetline-GI (80, 2, 80) را نشان میدهد. با استفاده از چنین توزیعی می توان احتمال اینکه الگوریتم مورد نظر در تعداد مشخصی تکرار همگرا شود را پیدا نمود.



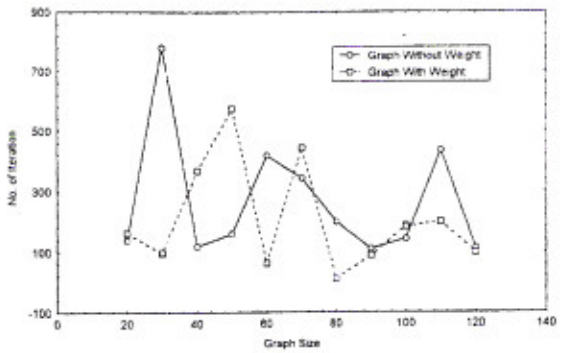
شکل ۱۳-توزیع تعداد تکرار های حلقه repeat-until برای الگوریتم Tsetline-GI (80, 2, 80)

تذکر ۲: در صورتیکه گراف ، یک گراف بدون وزن باشد. الگوریتم های فوق را میتوان بدون افزایش مرتبه پیچیدگی برای تناظر گراف استفاده نمود. الگوریتم Krylov-GI (K, 1, n) برای گراف های بدون وزن استفاده شده و مقدار متوسط تکرار حلقه repeat-until و تعداد اجراهایی که همگرا نشده اند در جدول ۱۷ آورده شده است. همانطور که دیده میشود تعداد تکرار های لازم در مقایسه با گراف های با وزن بیشتر میباشد. احتمالا این بدان علت است که وزنهای کمانها یکسان میباشند (وزن هر کدام از کمانها یک میباشد) و الگوریتم برای پیدا کردن تناظر گراف به فرصت بیشتری نیاز دارد.

جدول۱۷: متوسط تعدادتکرارلقه repeat-until درالگوریتم Krylov-GI (K,1, n) برای تناظر دو گراف بدون وزن برای ده اجرای مختلف

اندازه گراف	تعداد تکرار حلقه repeat-until	تعداد اجراهای همگرا نشده در ۲۰۰۰ تکرار
۲۰	۱۴۰	۴
۳۰	۷۸۰	۲
۴۰	۱۲۰	۰
۵۰	۱۶۰	۲
۶۰	۴۱۸	۵
۷۰	۳۴۴	۵
۸۰	۱۹۹	۰
۹۰	۱۱۰	۰
۱۰۰	۱۴۳	۱
۱۱۰	۴۳۵	۲
۱۲۰	۱۱۳	۲

در شکل ۱۴ متوسط تعداد تکرار حلقه repeat-until در الگوریتم Krylov-GI (K, 1, n) برای گراف های با وزن و بدون وزن نشان داده شده است.



شکل ۱۴ : متوسط تعداد تکرار حلقه repeat-until در الگوریتم Krylov-GI (K, 1, n) برای گراف های با وزن و بدون وزن

۵- نتیجه گیری

در این مقاله چهار اتوماتان مهاجرت اشیا برای مسئله تناظر دو گراف ارائه شده است. آزمایشهای مختلف نشان میدهند که برای گراف با اندازه n، این الگوریتم ها دارای پیچیدگی از مرتبه $O(n^3)$ میباشند. (توجه کنید که الگوریتم Backtracking دارای پیچیدگی زمانی از مرتبه O (n!) میباشد). با توجه به اینکه تشخیص صحت جواب نگاشت تولید شده نیاز به زمان $\theta(n^2)$ دارد در نتیجه میتوان گفت الگوریتم های پیشنهاد شده برای تناظر دو گراف دارای پیچیدگی زمانی از مرتبه O (n) میباشند که در مقایسه با الگوریتم Backtracking بهبود قابل ملاحظه ای را در مرتبه بزرگی[1] نشان میدهند. اگرگراف ها دارای اغتشاش باشند مسئله بسیار پیچیده تر میباشد ولی شبیه سازیها نشان میدهند که مرتبه متوسط زمان اجرای الگوریتم ها برای گراف های با اغتشاش و بدون اغتشاش تغییر نمیکند. بنظر میرسد در صورتیکه نگاشت اولیه σ توسط پیش پردازش انتخاب گردد، کارایی الگوریتم بهبود قابل ملاحظه ای میابد که در گزارش های بعدی به آن اشاره خواهد شد.

۶- مراجع

[1] Abadir, M.S. and Ferguson, J., "An Improved Layout Verification Algorithm", Proc. of IEEE European Int. Conf. on Design and Automation, pp. 391-395, 1990.

[2] Agusa, K., Fujita, S., Yamashita, M., and Ae, T., "On Neural Networks for Graph Isomorphism Problem", Proc. of RNNS/IEEE Int. Symposium on Neuroinformatics and Neurocomputers, pp. 1142-1148, 1992.

[3] Cinque, L., Yasuda, D., Shapiro, L. G., Tanimoto, S., and Allen, B., "An Improved Algorithm for Relational Distance Graph Matching", Pattern Recognition, Vol. 29, No. 2, pp. 349-359, 1996.

1- Order of Magnitude

[4] Depiero, F. Trived, M., and Serbin, S., "Graph Matching Using Direct Classification of Node Attendance", Pattern Recognition, Vol. 29, No. 6, pp. 1031-1048, 1996.

[5] Hashim, A. A., Amir, S., and Mars, p. , "Application of Learning Automata to Data Compression", In Adaptive and Learning Systems, K. S. Narendra (Ed.), New York: Plenum Press, pp. 229-234, 1986.

[6] Huang, K.-T.and Overhauser, D., "A Novel Graph Algorithm for Circuit Recognition", Proc. of IEEE Int. Symposium on Circuits and Systems, pp. 1695-1698, 1995.

[7] Kremar, M. and Dhawan, A. P., "Application of Genetic Algorithms in Graph Matching", Proc. of IEEE Int. Conf. on Conference on Neural Networks (ICNN'94), pp. 3872-3876, 1994.

[8] Mars, P., Chen, J. R., and Nambiar, R. , Learning Algorithms: Theory and Application in Signal Processing, Control, and Communications, CRC press, New York, 1996.

[9] Mars. P. and Narendra. K. S., and Chrystall, M. , "Learning Automata Control of Computer Communication Networks", Proc. of Third Yale Workshop on Applications of Adaptive Systems Theory, Yale University, 1983.

[10] Maurer, P.M. and Schapira, A.D., "A Logic-To-Logic Comparator for VLSAI Layout Verification", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 7, No. 8, pp. 897-907, 1988.

[11] Meybodi, M. R. and Beigy, H. , "New Class of Learning Automata Based Scheme for Adaptation of Backpropagation Algorithm Parameters", Proc. of EUFIT-98 , Sep. 7-10, Achen, Germany, pp. 339-344, 1998.

[12] Meybodi. M. R. and Lakshmivarhan, S. , "A Learning Approach to Priority Assignment in a Two Class M/M/1 Queuing System with Unknown Parameters", Proc. of Third Yale Workshop on Applications of Adaptive Systems Theory, Yale University, pp. 106-109, 1983.

[13] Narendra, K. S. and Thathachar, M. A. L. , Learning Automata : An Introduction, Prentice-hall, Englewood cliffs, 1989.

[14] Oommen, B. J. and Ma, D. C. Y. , "Deterministic Learning Automata Solutions to the Equipartitioning Problem" , IEEE Trans. on Computers, No. 37, No. 1, pp. 2-13, 1988.

[15] Oommen, B. J., Valiveti, R. S., and Zgierski, J. R. , "An Adaptive Learning Solution to the Keyboard Optimization Problem", IEEE Trans. on Systems, Man, and Cybernetics, Vol. 21, No. 6, pp. 1608-1618, 1991.

[16] Oommen, B. J. and Croix, E. V. de St. , "Graph Partioning Using Learning Automata", IEEE Trans. on Computers, No. 45, No. 2, pp. 195-208, 1996.

[17] Petrank, E.and Roth, R.M., "Is Code Equivalence easy to decide", IEEE Trans. on Information Theory, Vol. 43, No. 5, pp. 1602-1604, 1997.

[18] Schalkoff, R. J., Pattern Recognition: Statistical, Structural and Neural Approaches, John Whiley, New York, 1992.

[19] Thathachar, M. A. L. and Sastry, P. S., Learning Optimal Discriminant Functions Throught a Cooperative Game of Automata, IEEE Trans. Syst., Man and Cybern., Vol. SMC-27, pp.73-85, 1987.

[20] Wang, Y., Fan, K., and Horng, J., "Genetic-Based Search for Error-Correcting Graph Isomorphism", IEEE Trans. on Systems, Man, and Cybernetics-PartB: Cybernetics, Vol. 27, No. 4, pp. 588-597, 1997.

[21] Wilson, R. C. and Hancock, E. R., "Structural Matching by Discrete Relaxation", IEEE Trans. on Pattern Analysis and Machine Vision, Vol. 19, No. 6, pp. 634-648, 1997.

[22] Beigy, H. and Meybodi, M. R., "Randomized Las Vegas Algorithms for Graph Isomorphism", Proc. of ICEE-99, Vol. 3, pp. 1-8, Tehran, Iran , 1999.

[23] Beigy, H. and Meybodi, M. R. , "Optimization of Topology of neural Networks Using Learning Automata", Proc. of 3th Annual Int. Computer Society of Iran Computer Conf. CSICC-98, Tehran, Iran, pp. 417-428, 1999.