یازدهمین کنفرانس بین المللی کامپیوتر انجمن کامپیوتر ایران، پژوهشگاه دانشهای بنیادی، پژوهشکده علوم کامپیوتر، ۴ تا ۶ بهمن ۱۳۸۴، تهران، ایران.
11th International CSI Computer Conference (CSICC'2006), School of Computer Science, IPM, Jan. 24-26, 2006, Tehran, Iran.

بهبود الگوریتم ترکیبی (GA+LA) برای حل مساله تناظر گراف

مهدی رضاپور میرصالح محمد رضا میبدی

آزمایشگاه سیستم‌های نرم افزاری
دانشکده مهندسی کامپیوتر و فناوری اطلاعات
دانشگاه صنعتی امیرکبیر
تهران- ایران
(mirsaleh, meybodi)@ce.aut.ac.ir

چکیده: اتوماتاهای یادگیرو الگوریتم‌های ژنیتکی ابزارهای جستجوی قدرتمندی هستند که برای حل بسیاری از مسائل مشکل به‌کار برده شده‌اند. یکی از الگوریتم‌هایی که برای حل مساله تناظر گراف ارائه شده است الگوریتم ترکیبی (GA+LA) می‌باشدکه به‌طور همزمان از دو روش آتاماتاهای یادگیر و الگوریتم‌های ژنیتکی برای جستجو در فضای حالت استفاده می‌کند. با استفاده همزمان از اتوماهاتای یادگیر و الگوریتم ژنتیکی در فرآیند جستجو، سرعت رسیدن به جواب، افزایش چشم‌گیری پیدا می‌کند و از بدام افتادن الگوریتم در حداقل‌های محلی جلوگیری می‌شود. این الگوریتم ترکیبی برای گرافهای بدون وزن بسیار کند عمل میکند ولی در عوض برای گراف‌های وزن‌دار از کارایی بالایی برخوردار است. در این مقاله روشی برای وزن دهی راسها و یالهای یک گراف بدون وزن پیشنهاد میگردد. این روش وزن دهی راسها و یالهای گراف را بطریقی انجام میدهد که الگوریتم ترکیبی (GA+LA) بتواند با سرعت بالایی تناظر دو گراف بدون وزن را پیدا نماید. الگوریتم ترکیبی (GA+LA) که از این روش وزن دهی استفاده میکند بر روی گرافهای متعددی آزمایش گردید. نتایج آزمایشها حاکی از کارایی روش وزندهی پیشنهادی در این مقاله میباشد.

کلمات کلیدی: تناظر گراف، اتوماتاهای یادگیر، الگوریتم‌های ژنتیکی

۱ - مقدمه

گراف‌ها و بویژه گراف‌های برچسب دار ابزار قدرتمندی هستند که به طور گسترده در کاربردهای متعددی مورد استفاده قرار می‌گیرند. یکی از مسائل مهم در گراف مساله تناظر گراف (زیرگراف) می باشد. بسیاری از کاربردهای عمومی از جمله صحت مدارهای VLSI، صحت مدارهای PC-Board [۶]، معادل بودن دو برنامه [۷]، شناسایی الگو [۸] بینایی ماشین و فرمولهای شیمیایی [۹] [۱۰] را میتوان با تناظر گراف(زیر گراف) مدل کرد.

دو گراف G1 و G2 متناظر گفته می شود اگر یک نگاشت بین راس‌های G1 و G2 مشاهده شود بگونه ای که ساختار یال‌ها بوسیله این نگاشت حفظ شود. هرگاه چنین نگاشتی موجود باشد G1 و G2 متناظر گفته میشود. اگر بطور مثال G2 شامل تعداد راس‌های بیشتری نسبت به G1 باشد، بایستی یک تناظر زیر گراف از G1 به G2 جستجو شود. بدین معنا که یک زیرگراف S از G2 به گونه ای پیدا کنیم که G1 و S متناظر باشند.

در بسیاری از کاربردها مقادیر زیادی از اغتشاش در توصیف ساختاری وجود دارد. این اغتشاش‌ها که باعث خرابی شی می شوند، یافتن یک تناظر بین گراف ورودی و گرافهای موجود را غیر ممکن می سازد. از اینرو انطباق غیر دقیق یا اصطلاحا تناظر گراف با تصحیح خطا، تنها راه حل مساله در این حالت است[۳]. یک روش برای حل مساله تناظر گراف استفاده از الگوریتم Backtracking می باشد که در بدترین حالت برای گراف با اندازه n نیاز به بررسی O(n!) تناظر دارد[۱۶]. با در نظر گرفتن محدودیتهایی برای گراف می توان الگوریتم‌هایی با پیچیدگی چند جمله ای بدست آورد. بعنوان مثال، Luks و Hoffman الگوریتمی با پیچیدگی چند جمله ای برای تناظر گراف‌های با ظرفیت محدود ارائه دادند[۵]. برای گراف‌های سه ظرفیتی الگوریتم‌هایی با پیچیدگی $O(n^4)$ موجود می باشند. در [۱۵] الگوریتمی برای محاسبه تناظر گراف‌های مسطح ارائه شده است که پیچیدگی آن خطی نسبت به اندازه گراف میباشد. اگر چه از نقطه نظر تئوری این الگوریتم‌ها جالب هستند، اما بدلیل سربار زیاد کاربردی نیستند.

اتوماتاهای یادگیر و الگوریتم‌های ژنیتکی[۶]، هر دو ابزار جستجوی عمومی می باشند که برای حل بسیاری از مسائل مشکل از جمله افراز اشیاء[۱۳]، افراز گراف[۲۰]، بهینه سازی صفحه کلید[۲۱] و پیدا کردن ساختار بهینه شبکه عصبی[۴] بکار برده شده است. در[۲۲] یک الگوریتم ترکیبی به نام (GA+LA) برای حل مساله تناظر گراف ارایه گردید. این الگوریتم از دو روش الگوریتم‌های ژنیتکی و اتوماتاهای یادگیر بطور همزمان برای جستجوی در فضای حالت استفاده می‌نماید. نشان داده شده است که با استفاده همزمان اتوماتاهای یادگیر و

الگوریتم ژنتیکی در فرایند جستجو، سرعت رسیدن به جواب افزایش چشمگیری پیدا می نماید. نتایج آزمایش‌ها، برتری این الگوریتم را نسبت به الگوریتم مبتنی بر الگوریتمهای ژنیتکی و همچنین الگوریتمهای مبتنی بر آتوماتاهای یادگیر نشان می دهد. یکی از مشکلات الگوریتم ترکیبی (GA+LA) نداشتن کارایی بالا برای گرافهای بدون وزن میباشد. در این مقاله روشی برای وزن دهی گراف‌های بدون وزن پیشنهاد میگردد. این روش، وزن دهی گراف را بطریقی انجام میدهد که الگوریتم ترکیبی (GA+LA) بتواند با سرعت بالایی تناظر دو گراف بدون وزن را پیدا نماید. آزمایش الگوریتم ترکیبی (GA+LA) که از این روش وزن دهی استفاده میکند بر روی گرافهای متعددی حاکی از کارایی روش وزندهی پیشنهادی میباشد.

ادامه مقاله بدین صورت سازماندهی شده است. بخش دوم مساله تناظر گراف را تعریف میکند. توضیح مختصری در باره اتوماتاهای یادگیر و الگوریتمهای ژنتیکی در بخش ۳ آمده است. در بخش ۴، الگوریتم ترکیبی (GA+LA) شرح داده میشود. بخش ۶ به ارایه روش وزندهی پیشنهادی و نتایج آزمایشها میپردازد. بخش نهایی مقاله نتیجه گیری میباشد.

۲- تعریف مساله

یک گراف وزن‌دار به صورت سه‌تایی (V, E, α) نمایش داده می‌شود که در آن V مجموعه غیرتهی راس‌ها، $E \subset V \times V$ مجموعه یال‌ها و $\alpha : V \to R_v$ تابعی از V به R_v است که R_v مجموعه وزن یال‌ها است.

دو گراف G1=(V1,E1) و G2=(V2,E2) را متناظر می‌گویند اگر تابع یک به یک و پوشای f:V1→V2 وجود داشته باشد که برای هر یال (u,w) در گراف G1 یال (f(u),f(w)) در گراف G2 وجود داشته باشد. اگر ماتریس $M(G)$ $n \times n$، ماتریس همسایگی گراف G با n راس باشد جایگشت سطرها و متناظر با آن، ستون‌های ماتریس مجاورت ترتیب‌های مختلفی از راس‌ها را حاصل می کند. از اینرو می توان گفت دو گراف G و H متناظر هستند اگر و فقط اگر ماتریس‌های مجاورت آنها(M(G) و M(H)) تنها در جایگشت سطرها و ستون‌ها متفاوت باشند. رابطه زیر برای دو ماتریس M(G) و M(H) که توسط σ به یکدیگر نگاشت شده اند برقرار می باشد:

$$M(H) = P.M(G).P^T$$

که در آن P ماتریس جایگشت نگاشت σ است. اختلاف دو گراف به صورت $J(\sigma) = \| M(H) - P.M(G).P^T \|$ تعریف می شود که در آن $\| . \|$ نرم ماتریس (هر نوع نرم) می باشد. با توضیحات داده شده مشخص می شود که مساله تناظر گراف به کمینه کردن $J(\sigma)$ تبدیل می شود. برای گراف به اندازه n، محاسبه $J(\sigma)$ نیاز به $\theta(n^3)$ زمان دارد. می توان نشان داد که تحت نگاشت σ، رابطه زیر برقرار است.

$$[P.M(G).P^T]_{i,j} = [M(G)]_{\sigma(i),\sigma(j)}$$

که $[A]_{i,j}$ عضو سطر i و ستون j ماتریس A می باشد. اگر در رابطه $J(\sigma) = \| M(H) - P.M(G).P^T \|$ نرم $\| M \| = \sum_i \sum_j | m_{ij} |$) L1) استفاده کنیم شکل ساده تری برای محاسبه مقدار اختلاف بین دو گراف G و H بدست می آید. برای این منظور خطای نگاشت راس k از گراف G به راس $\sigma(k)$ از گراف H به صورت زیر تعریف می شود.

$$J_k(\sigma) = \sum_{m=1}^{n} | [M(H)]_{k,m} - [M(G)]_{\sigma(k),\sigma(m)} |$$

$$+ \sum_{m=1}^{n} | [M(H)]_{m,k} - [M(G)]_{\sigma(m),\sigma(k)} |$$

در صورتیکه گراف بدون جهت باشد خطای نگاشت راس k برابر است با $J_k(\sigma) = 2 \times \sum_{m=1}^{n} | [M(H)]_{k,m} - [M(G)]_{\sigma(k),\sigma(m)} |$ و در نتیجه خطای تناظر دو گراف برای نگاشت σ به صورت $J(\sigma) = \sum_{k=1}^{n} J_k(\sigma)$ تعریف می گردد. با توجه به معادلات فوق متوان گفت که محاسبه $J(\sigma)$ نیاز به $\theta(n^2)$ زمان دارد.

۳ - اتوماتاهای یادگیر و الگوریتمهای ژنتیکی

یادگیری در اتوماتاهای یادگیر، انتخاب یک اقدام بهینه از میان یک مجموعه از اقدام‌های مجاز اتوماتای یادگیرمی باشد. این اقدام روی یک محیط تصادفی اعمال می شود و محیط به این اقدام اتوماتای یادگیربوسیله یک پاسخ تصادفی از مجموعه پاسخ‌های مجاز جواب می دهد. پاسخ محیط بصورت آماری به اقدام اتوماتای یادگیر وابسته است. محیط شامل اجتماع تمام شرایط خارجی و تاثیرات آنها روی عملکرد اتوماتای یادگیر می باشد. اتوماتاهای یادگیر دارای کاربردهای فراوانی می باشد. بعضی از این کاربردها عبارتند از: مسیریابی در شبکه‌های ارتباطی[۷]، فشرده سازی تصاویر[۸]، شناسایی الگو[۹]، برنامه ریزی فرآیندها در یک شبکه کامپیوتری[۱۰]، تئوری صف[۱۱]، کنترل دسترسی در شبکه‌های انتقال ناهمزمان[۱۱]، کمک به آموزش شبکه‌های عصبی[۱۲]، دسته بندی و افراز اشیاء[۱۳] و پیدا کردن ساختار بهینه برای شبکه‌های عصبی[۱۴].برای اطلاعات بیشتر در باره اتوماتاهای یادگیر میتوان به [۱] مراجعه نمود.

برای یک گراف با اندازه n، !n نگاشت مختلف وجود دارد و در صورتیکه از اتوماتاهای یادگیر برای پیدا نمودن تناظر دو گراف استفاده شود، اتوماتای یادگیر باید !n اقدام داشته باشد که تعداد زیاد اقدامها سرعت همگرایی اتوماتای یادگیر را کم می کند بهمین جهت آتاماتای یادگیر مهاجرت اشیاء توسط اومن و ما برای این منظور پیشنهاد شده است[۱]. الگوریتم‌های ژنتیکی که برمبنای ایده تکامل در طبیعت عمل مینماید، برروی جمعیتی از راه‌حلهای بالقوه به جستجوی راه حل نهایی

می‌پردازد. در هر نسل، بهترینهای آن نسل انتخاب می‌شوند، و پس از زاد و ولد، مجموعهٔ جدیدی از فرزندان را تولید می‌کنند. در این فرایند افراد مناسبتر با احتمال بیشتری در نسلهای بعد باقی خواهند ماند[۲]. برای اطلاعات بیشتر در باره الگوریتمهای ژنتیکی میتوان به [۱۹-۱۷][۲] مراجعه نمود.

۴- الگوریتم ترکیبی(GA+LA) برای حل مساله تناظر گراف

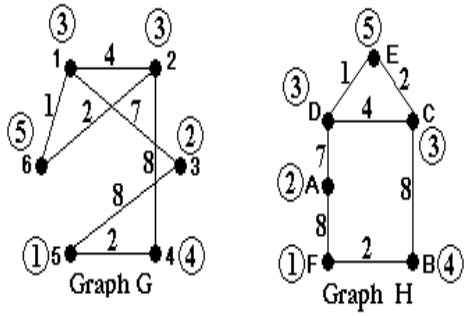
با ترکیب الگوریتم ژنتیکی و اتوماتای یادگیر و تلفیق مفاهیم ژن، کروموزوم، اقدام و عمق، میتوان سابقه تاریخی تکامل راه حلهای جزیی مساله را به صورت کارا استخراج نمود و در روند جستجوی راه حل نهایی مورد استفاده قرار داد. در ادامه ابتدا به مولفه های اصلی الگوریتم ترکیبی اشاره میشود و سپس الگوریتم ترکیب توضیح داده شده است[۲۲].

ژن و کروموزوم: در الگوریتم ترکیبی برخلاف الگوریتمهای ژنتیکی هر کروموزوم توسط یک اتوماتای یادگیر از نوع مهاجرت اشیا نشان داده میشود. بطوریکه هر کدام از ژنها در کروموزوم به یکی از اقدامهای آتاماتا نسبت داده میشود و در یک عمق مشخصی از آن اقدام قرار میگیرد.

در اتوماتای یادگیر از نوع مهاجرت اشیا $\underline{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_k\}$ مجموعه اقدامهای مجاز برای اتوماتای یادگیر است. این اتوماتای یادگیر K اقدام دارد (تعداد اقدامهای اتوماتای یادگیر با تعداد راسهای گراف برابر است). اگر راس u از گراف H در اقدام m قرار گرفته باشد، در این صورت راس u از گراف H با راس M_m از گراف G متناظر است. $\underline{\Phi} = \{\Phi_1, \Phi_2, ..., \Phi_{KN}\}$ مجموعه وضعیتها و N عمق حافظه برای اتوماتای یادگیر می باشد. مجموعه وضعیتهای این اتوماتای یادگیر به K زیر مجموعه افراز می شود و راسهای گراف بر اساس اینکه در کدام وضعیت قرار داشته باشند دسته بندی می گردند. اگر راس u از گراف H در مجموعه وضعیتهای $\{\Phi_{(j-1)N+1}, ..., \Phi_{jN}\}$ قرار داشته باشند در این صورت راس u از گراف H با راس M_j از گراف G متناظر است. در مجموعه وضعیتهای

اقدام j وضعیت $\Phi_{(j-1)N+1}$ وضعیت داخلی و وضعیت Φ_{jN} را وضعیت مرزی می نامیم. راسی که در وضعیت $\Phi_{(j-1)N+1}$ قرار دارد را راس با اهمیت بیشتر و راسی را که در وضعیت Φ_{jN} قرار دارد را با اهمیت کمتر نامیده می شوند.

در اثر پاداش دادن یا جریمه کردن یک اقدام، وضعیت راس وابسته به آن اقدام، تغییر می کند. اگر راسی در وضعیت مرزی یک اقدام قرار داشته باشد، جریمه شدن آن باعث تغییر اقدامی که راس به آن وابسته می‌باشد می‌شود و در نتیجه باعث ایجاد جایگشت جدیدی می‌گردد. به عنوان مثال دو گراف وزندار G و H که شکل زیر داده شده استرا در نظر بگیرید. هر کدام از این گرفها دارای ۶ راس می باشند. راسهای گراف G را با مجموعه $\{1,2,3,4,5,6\}$ و راسهای گراف H را با مجموعه $\{A,B,C,D,E,F\}$ نشان می دهیم. اعداد ذکر شده در دایره ها وزنهای راسها گراف میباشد



مجموعه $\{(1,D),(2,F),(3,A),(4,B),(5,E),(6,C)\}$ یک جایگشت از دو گراف را نشان می دهد. این جایگشت توسط یک اتوماتای یادگیر با اتصالهای مشابه آتاماتای Tsetline (شکل ۲) نشان داده شده است [۱]. این اتوماتای یادگیر دارای ۶ اقدام $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6\}$ (به تعداد راسها گرافG) و هر اقدام دارای عمق ۵ می باشد. مجموعه وضعیت‌های {۲۶و۲۱و۱۶و۱۱و۶و۱} وضعیت‌های داخلی و مجموعه وضعیت‌های {۳۰و۲۵و۲۰و۱۵و۱۰و۵} وضعیت‌های مرزی اتوماتای یادگیر هستند. در ابتدا هر یک از راس‌های گراف H در وضعیت مرزی اقدام مربوطه قرار دارد. در الگوریتم ترکیبی هر ژن از کروموزوم یک اقدام اتوماتای یادگیر می باشد و لذا در ادامه مقاله این دو واژه را به جای یکدیگر بکار میبریم. اتوماتای یادگیر(کروموزوم) شکل ۲ دارای ۶ اقدام (ژن) می باشد و هر اقدام (ژن) دارای ۵ وضعیت داخلی است.

جمعیت: با توجه به نتایج حاصله از آزمایش‌های انجام گرفته، هر چه تعداد اعضاء جمعیت در این الگوریتم به تعداد راس‌های گراف نزدیک باشد، الگوریتم سریعتر به جواب همگرا می شود و لذا تعداد اعضاء جمعیت برابر با تعداد راس‌های گراف در نظر گرفته می شود [۲۳]. برای تشکیل جمعیت اولیه بدین صورت عمل می شود. اگر گراف دارای n راس باشد، $n-1$ عضو جمعیت با ایجاد جایگشت تصادفی تولید می شوند. برای تولید آخرین عضو جمعیت، درجه راس‌های دو گراف مورد نظر را محاسبه، و سپس آنها را در دو آرایه مرتب می کنیم، جایگشتی که از راس‌های متناظر در دو آرایه مرتب شده حاصل می شود را به عنوان آخرین عضو جمعیت اولیه در نظر می گیریم. به این جایگشت، جایگشت تقریبی می گوییم. به عنوان مثال نحوه تشکیل جمعیت اولیه دو گراف G و H در ادامه توضیح داده شده است. پنج عضو اول جمعیت بوسیله پنج جایگشت تصادفی

$\{(1,B),(2,D),(3,E),(4,A),(5,F),(6,C)\}$

$\{(1,D),(2,E),(3,F),(4,B),(5,C),(6,A)\}$

$\{(1,E),(2,F),(3,B),(4,D),(5,A),(6,C)\}$

$\{(1,C),(2,F),(3,B),(4,E),(5,D),(6,A)\}$

$\{(1,B),(2,D),(3,C),(4,A),(5,E),(6,F)\}$ ایجاد می‌شوند. برای ایجاد جایگشت ششم با مرتب کردن درجه راس‌های دو گراف G و H جایگشت $\{(6,E),(5,F),(4,B),(1,D),(2,A),(3,C)\}$ بدست می آید. نحوه بدست آوردن این جایگشت در شکل ۳ نشان داده شده است. جمعیت اولیه دو گراف G وH در شکل ۴ نشان داده شده است.

تابع برازندگی: در مساله تناظر گراف هدف یافتن جایگشت σ (کروموزوم) میباشد که $J(\sigma)$ کمینه باشد. تابع برازندگی $f(\sigma)$ به صورت زیر تعریف می شود که در آن M(G) و M(H) ماتریسهای گراف‌های G و H و P ماتریس جایگشت نگاشت σ هستند.

$$f(\sigma) = C\max - J(\sigma) =$$
$$C\max - \| M(H) - P.M(G).P^T \|$$

Cmax ماکزیمم $J(\sigma)$ میباشد. اگر چه تابع برازندگی شامل ضرب ماتریسی $P.M(G).P^T$ می باشد که می تواند به صورت $[P.M(G).P^T]_{i,j} = [M(G)]_{\sigma(i),\sigma(j)}$ نوشته شود و لذا در زمان $O(n^2)$ قابل محاسبه است. اگر نرم $\|.\|$ با نرم L1 $(\| M \| = \sum\sum |m_{ij}|)$ جایگزین شود تابع برازندگی به صورت زیر خواهد بود.

$$f(\sigma) = C\max - \sum_{i=1}^{n}\sum_{j=1}^{n} | m_{ij} - m_{\sigma(i),\sigma(j)} |$$

عملگرها: با توجه به اینکه در الگوریتم ترکیبی هرکروموزوم بصورت یک آتاماتای یادگیر نمایش داده میشود عملگرهای جابجایی و جهش با مشابه این عملگرها در الگوریتمهای ژنتیکی متفاوت میباشند.

عملگر جابجایی: در این عملگر دو کروموزوم(اتوماتای یادگیر) والد انتخاب می شوند و به صورت تصادفی دو ژن i و j در یکی از دو کروموزوم والد انتخاب میشوند سپس همین دو ژن در کروموزوم دیگر انتخاب میشوند. مجموعه ژن‌های با شماره‌های بین i و j را مجموعه جابجایی مینامیم. سپس ژنهای هم شماره در دو مجموعه جابجایی با یکدیگر جابجا میشوند (مثلا ژن شماره i از مجموعه جابجایی اول با ژن شماره i از مجموعه جابجایی دوم، ژن شماره i+1 از مجموعه جابجایی اول با ژن شماره i+1 از مجموعه جابجایی دوم و ...). با این عمل دو کروموزوم(اتوماتای یادگیر) جدید حاصل می شوند که اصطلاحا فرزندان دو آتاماتای والد خوانده می شوند. با توجه به آزمایشهایی که انجام گرفته است نرخ اعمال این عملگر بایستی کوچک و در حدود 0.1 در نظر گرفته شود.

به عنوان مثال اتوماتاهای یادگیر LA_5 و LA_2 از جمعیت تشکیل شده قبلی به صورت تصادفی انتخاب می شوند. سپس با انتخاب تصادفی دو ژن α_3,α_2 مجموعه جابجایی $\{\alpha_2,\alpha_3\}$ برای دو کروموزوم حاصل می شود و نهایتا مطابق شکل ۴ با جابجایی ژنهای متناظر در مجموعه‌های جابجایی، دو کروموزوم جدید حاصل می شود.

عملگر جهش: در عملگر جهش که بر روی یک کروموزوم انجام می شود، دو ژن به صورت تصادفی انتخاب شده و جابجا می شوند. با توجه به آزمایشهایی که انجام گرفته است نرخ اعمال این عملگر بایستی در حدود 0.4 در نظر گرفته شود.

عملگر جریمه و پاداش: در هر یک از کروموزومها پس از بررسی میزان برازندگی یک ژن که به صورت تصادفی انتخاب می شود، آن ژن پاداش یا جریمه داده می شود. در اثر پاداش دادن یا به عنوان مثال در آتاماتای با اتصال‌های مشابه آتاماتای Tsetline اگر راس B در مجموعه وضعیت‌های {۲۰و۱۹و۱۸و۱۷و۱۶} قرار داشته باشد و برازندگی راس B از گراف H و راس 4 از گراف G از مقدار آستانه (مقدار آستانه به صورت تطبیقی مشخص می گردد و مقدار آن در هر لحظه برابر است با میانگین خطای نگاشت برای تمامی راس‌ها) کوچکتر باشد به این راس پاداش داده می شود و اهمیت راس افزایش میابد و به سمت وضعیت‌های داخلی تر این اقدام حرکت می کند. اگر راس B در وضعیت داخلی(۱۶) قرار داشته باشد و پاداش بگیرد در همان وضعیت باقی می ماند. این مورد در شکل ۵ نشان داده شده است. اگر میزان برازندگی یک راس از گراف از مقدار آستانه بزرگتر باشد در اینصورت تناظر برقرار شده مناسب نبوده و این راس جریمه می شود. نحوه حرکت چنین راسی برای دو حالت مختلف در زیر آمده است.

الف) راس در وضعیتی غیر از وضعیت مرزی قرار داشته باشد: جریمه نمودن این راس سبب کم اهمیت شدن این راس می شود. نحوه حرکت چنین راسی در شکل ۶ نشان داده شده است.

ب) راس در وضعیت مرزی قرار دارد. در این حالت راسی از گراف H را پیدا می کنیم بطوریکه اگر در نگاشت مربوطه جای دو راس عوض شوند بیشترین کاهش در مقدار خطا حاصل گردد. در اینصورت اگر راس پیدا شده در وضعیت مرزی قرار داشته باشد جای دو راس عوض می شود و در غیر اینصورت ابتدا راس مشخص شده به وضعیت مرزی اقدام خود منتقل و سپس جابجایی صورت می پذیرد. نحوه حرکت چنین راسی در شکل ۷ نشان داده شده است.

برای اطلاعات بیشتر در باره الگوریتم ترکیبی و نتایج آزمایشها میتوان به [۲۲][۲۳] مراجعه نمود.

۵- الگوریتم پیش پردازش

در یک گراف راس‌ها، یال‌ها، وزن یال‌ها، درجه راس‌ها و نحوه ارتباط بین راس‌ها از مشخصه‌های هستند که در یافتن تناظر بین دو گراف نقش مهمی بازی میکنند. در [۲۲] نشان داده شده است که الگوریتم ترکیبی عملکرد بهتری برای گراف‌های وزن دار در مقایسه با گراف بدون وزن از خود نشان می‌دهد. آزمایشها نشان داه است[۲۲][۲۳] که هر چه پراکندگی وزنی (یال‌ها و راس‌ها) در گراف بیش‌تر باشد الگوریتم ترکیبی، تناظر بین دو گراف را در مدت زمان کوتاه تری پیدا می‌کند. منظور از پراکندگی وزنی یال‌ها (راس‌ها)، تعداد وزن‌های مختلفی است

که به یال‌ها (راس‌ها) نسبت داده می‌شود. به طور مثال اگر گراف، وزن یال‌ها (راس‌ها) را از مجموعه {2,5,8} اختیار کند، پراکندگی وزنی یال‌های (راس‌های) این گراف ۳ خواهد بود. هرچه تعداد اعضاء این مجموعه بالاتر باشد پراکندگی وزنی یال‌ها (راس‌ها) بالاتر خواهد بود. اگر مجموعه وزن یال ها را با R_E و مجموعه وزن راس ها راR_V با نشان دهیم، پراکندگی وزن یال ها را با$|R_E|$ و پراکندگی وزن راس ها را با$|R_V|$ نمایش می دهیم.

برای گرافهای بدون وزن بدلیل اینکه وزن یال‌ها از مجموعه {0,1} انتخاب می‌شود (0 به معنای عدم وجود یال، و 1 به معنای وجود یال بین دو راس) پراکندگی وزنی یال‌ها ۲ می‌باشد. الگوریتم ترکیبی بر روی چنین گراف‌هایی خوب عمل نمیکند. جهت بهبود عملکرد الگوریتم ترکیبی برای گراف‌های بدون وزن بایستی گراف بدون وزن به طریقی وزندار شود(پراکندگی وزنی گراف را بالا برد) که خصوصیات اصلی گراف از دست نرود. پیش پردازشی که در ادامه به جزئیات آن خواهیم پرداخت، باعث افزایش پراکندگی وزن‌های گراف بدون اینکه مشخصه‌های اصلی دو گراف تغییر کند میشود.

وزن قدیم یال * (مجموع وزن دو راس یال) = وزن جدید یال

که در این رابطه وزن هر راس گراف از رابطه زیر محاسبه می‌شود.

مجموع وزن یال‌های واردشده به راس + مجموع وزنهای یال‌های خارج شده از راس = وزن راس

اگر گراف جهت‌دار باشد قبل از تعیین وزن جدید یال‌ها و راس‌ها، وزن دهی منفی نیز لازم است. وزن‌دهدی منفی بدین صورت انجام می‌گیرد که اگر از p به q یالی با وزن w وجود داشته باشد، یالی با وزن w- از q به p ایجاد می‌کنیم(p و q راس‌های گراف هستند). جزئیات پیش پردازش پیشنهادی در الگوریتم ۱ که در انتهای مقاله آمده است نشان داده شده است.

در این الگوریتم فوق G1_Old و G2_Old ماتریس‌های مجاورت دو گراف قبل از تغییر، و G1_New و G2_New ماتریس‌های مجاورت دو گراف بعد از وزن دهی میباشد. هر بار اجرای این پیش پردازش بر روی دو گراف، پراکندگی وزنی گراف را بیش‌تر میکند. آزمایشها نشان داده است که پس از اجرای تعداد مشخصی پیش پردازش، با وجود افزایش وزن راس‌ها و یال‌ها، تغییری در پراکندگی وزن‌ها ایجاد نمیشود. طبق نتایج آزمایشی که بر روی گراف‌های مختلف با اندازه‌های ۲۰ تا ۱۲۰ راس انجام گرفته است، حداکثر تعداد پیش پردازشها تا ثابت ماندن پراکندگی ۵ بوده است.

شکل ۸ مثالی از اجرای الگوریتم پیش پردازش بر روی یک گراف بدون وزن با ۳۶ راس را نشان می‌دهد که وزن‌ها با رنگ‌های مختلف نشان داده شده است. وزن راس‌ها در ریف اول و با علامت R_V نشان داده شده است و وزن یال‌ها در ردیف دوم و با علامت R_E نشان داده شده است.

در گراف (۱) وزن همه راس‌ها صفر و وزن یال ها صفر یا یک میباشد($|R_E|=2$, $|R_V|=1$). پس از انجام قدم اول از مرحله یک (محاسبه وزن جدید راس‌ها) وزن هر راس از گراف یکی از مقادیر از مجموعه {4,6,8} می‌باشد (پراکندگی وزنی راس‌ها به ۳ افزایش می‌یابد) و پراکندگی وزنی یال‌ها بدون تغییر می‌ماند , $|R_V|=3$) ($|R_E|=2$. وزن ۴ مربوط به راس‌های ۱، ۶، ۳۱ و ۳۶، وزن ۶ مربوط به راس‌های ۲، ۳، ۴، ۵، ۷، ۱۲، ۱۳، ۱۸، ۱۹، ۲۴، ۲۵، ۳۰، ۳۲، ۳۳، ۳۴ و ۳۵ و وزن ۸مربوط به راس‌های ۸، ۹، ۱۰، ۱۱، ۱۴، ۱۵، ۱۶، ۱۷، ۲۰، ۲۱، ۲۲، ۲۳، ۲۶، ۲۷، ۲۸ و ۲۹ هستند. به عنوان نمونه وزن جدید راس شماره یک که برابر۴ می‌باشد، از مجموع ۲+۲ (یال‌های خارج شده از راس شماره یک + یال‌های خارج شده از راس شماره یک) حاصل شده است و وزن جدید راس شماره چهارده که برابر۸ می‌باشد، از مجموع ۴+۴ (یال‌های خارج شده از راس شماره چهارده + یال‌های خارج شده از راس شماره چهارده) به‌دست آمده است و ... .

پس از انجام قدم دوم از مرحله یک (محاسبه وزن جدید یال‌ها) وزن هر یک از یالهای گراف یکی از مقادیر مجموعه {0,10,12,14,16} می‌باشد (پراکندگی وزنی یال‌ها به ۵ افزایش می‌یابد) و پراکندگی وزنی راس‌ها بدون تغییر می‌ماند , $|R_V|=5$) ($|R_E|=2$. وزن ۱۰ مربوط به یال‌های (۲و۱)، (۷و۱)، (۶و۵)، (۱۲و۶)، (۳۱و۲۵)، (۳۲و۳۱)، (۳۶و۳۵) و (۳۶و۳۰)، وزن 12 مربوط به یال‌های (۳و۲)، (۴و۳)، (۵و۴)، (۱۳و۷)، (۱۹و۱۳)، (۲۵و۱۹)، (۱۸و۱۲)، (۲۴و۱۸)، (۳۰و۲۴)، (۳۳و۳۲)، (۳۴و۳۳)، (۳۵و۳۴)، وزن 14 مربوط به یال‌های (۸و۲)، (۹و۳)، (۱۰و۴)، (۱۱و۵)، (۸و۷)، (۱۴و۱۳)، (۲۰و۱۹)، (۲۶و۲۵)، (۳۲و۲۶)، (۳۳و۲۷)، (۳۴و۲۸)، (۳۵و۲۹)، (۱۲و۱۱)، (۱۸و۱۷)، (۲۴و۲۳)، (۳۰و۲۹) و وزن ۱۶ مربوط به یال‌های (۹و۸)، (۱۰و۹)، (۱۱و۱۰)، (۱۵و۱۴)، (۱۶و۱۵)، (۱۷و۱۶)، (۱۴و۸)، (۱۵و۹)، (۱۶و۱۰)، (۱۷و۱۱)، (۲۱و۲۰)، (۲۲و۲۱)، (۲۳و۲۲)، (۲۰و۱۴)، (۲۱و۱۵)، (۲۲و۱۶)، (۲۳و۱۷)، (۲۷و۲۶)، (۲۸و۲۷)، (۲۹و۲۸)، (۲۶و۲۰)، (۲۷و۲۱)، (۲۸و۲۲) و (۲۹و۲۳) هستند. به عنوان نمونه وزن جدید یال (۲و۱) که برابر10 می‌باشد، از حاصل‌ضرب (4+6)*1 ((وزن راس شماره دو + وزن راس شماره یک)* وزن قدیم یال (۲و۱)) حاصل شده است و وزن جدید یال (۱۱و۵) که برابر ۱۴ می‌باشد، از حاصل‌ضرب (8+6)*1 ((وزن راس شماره یازده + وزن راس شماره پنج)* وزن قدیم یال (۱۱و۵)) به‌دست آمده است و ... .

نتیجه آزمایشها: جدول ۱ تعداد تکرار مورد نیاز برای الگوریتم‌های ترکیبی مبتنی بر آتاماتاهای اومن، کرینسکی، کرایلو و ستلین در حالت بدون پیش پردازش(Pre0)، با انجام یک، دو و یا سه بار پیش پردازش (Pre1، Pre2 و Pre3) و مقایسه آن با الگوریتم‌های آتاماتای یادگیر (LA) مبتنی بر اتوماتاهای یادگیر اومن، کرینسکی، کرایلو و ستلین و الگوریتم ژنتیکی(GA) را نشان می‌دهد. همانگونه که مشاهده می‌شود پس از هربار اجرای پیش پردازش، تعداد تکرار الگوریتم ترکیبی کمتر

می‌شود و نهایتا پس از سه بار انجام پیش پردازش، (زمانی که پراکندگی وزن‌ها ثابت می‌ماند) تعداد تکرار الگوریتم به حداقل خود می‌رسد. نمودارهای ۱ الی ۴ مقایسه تعداد تکرار الگوریتم‌های ترکیبی مبتنی بر اتوماتاهای یادگیر اومن، کرینسکی، کرایلو و ستلین در برای بدون انجام پیش پردازش، یک و یا دو بار انجام پیش پردازش را نشان می‌دهد. ملاحظه می‌شود که پس از هر بار انجام پیش پردازش، متوسط تعداد تکرار الگوریتم کاهش می‌یابد. همچنین مشاهده می‌شود که میزان کاهش تعداد تکرارهای الگوریتم ترکیبی پس از انجام پیش پردازش اول (Pre1) بیشتر از اجرای پیش پردازش‌های دوم و سوم است. برای اطلاعات بیشتر در باره الگوریتم ترکیبی و پیش پردازش پیشنهادی در این مقاله و آزمایشهای بیشتر میتوان به [۲۲][۲۳] مراجعه نمود.

۶- نتیجه گیری

در این مقاله ابتدا الگوریتم ترکیبی (GA+LA) برای حل مساله تناظر گراف مختصرا شرح داده شد و سپس روشی برای وزن دهی گراف‌های بدون وزن پیشنهاد گردید. این روش وزن دهی، راسها و یا یالهای یک گراف بدون وزن را بطریقی وزن دهی میکند که الگوریتم ترکیبی (GA+LA) بتواند با سرعت بالایی تناظر دو گراف بدون وزن را پیدا نماید. الگوریتم ترکیبی (GA+LA) که از این روش وزن دهی استفاده میکند بر روی گرافهای متعددی آزمایش گردید. نتایج آزمایشها حاکی از کارایی روش وزندهی پیشنهادی در این مقاله بوده است.

مراجع

[1] Beigy, H. and Meybodi, M. R. "Randomized Las Vegas Algorithm for Graph Isomorphism", Proceedings of Third International Conference on Intelligent Data Engineering and Automated Learning, Manchester, UK, August. 12-14, 2002.

[2]Wang, Y. "Fan Genetic-Basic Search for Error-Correcting Graph Isomorphism", IEEE Transaction on Systems, Man. And Cybernetics-Par`t B: Cybernetics, Vol. 27, No. 4, August 1997.

[3]Messmer, B. T. "Efficient Graph Matching Algorithms for Preprocessed Model Graphs", Ph.D. Thesis, Inst. of Comp. Sci. and Applied Mathematics, University of Bern, 1996.

[4] 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.

[5]Hoffman, C. M. Group-theoretic Algorithms and Graph Isomorphism. Springer Verlag, 1982.

[6] Goldberg, D. E. "Genetic Algorithms in Search, Optimization and Machine Learning", Reading, MA:.Addition-Wesley, 1989.

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

[8] Hashim, 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.

[9]Thathachar, M. A. L. and Sastry, P. S. "Learning Optimal Discriminant Functions Through a Cooperative Game of Automata", IEEE Trans. Syst., Man and Cybern., Vol. 27, No. 4, pp. 588-597, 1997.

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

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

[12] 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.

[13] Oommen, B. J. and Ma,, D. C. Y. "Deterministic Learning Automata Solution to the Keyboard Optimization Problem", IEEE Trans. On Computers, Vol. 37, No. 1, pp. 2-3, 1988.

[14] 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.

[15] Hopcroft, J. E. and Wong, J. K. "Linear time algorithm for isomorphism of planar graphs", In Annual ACM Symposium on Theory of Computing, pages 172--184, 1974.

[16] Ullman, J. R. "An Algorithm for Subgraph Isomorphism", Journal of the Association for Computing Machinery, Vol. 23, No. 1, pp. 31--42, 1976.

[17] Falkenhainer, B., Forbus, K. D. and Gentner, D. "The Structure-mapping Engine: Algorithms and Examples", Artificial Intelligence, No 41, pp. 1--63, 1989/90.

[18] Bryant, K. Genetic Algorithms and the Traveling Salesman Problem , Thesis, Harvey Mudd College, Dept. of Mathematics, 2000.

[19] Cantu-Paz, E. "A Survey of Parallel Gentic Algorithms", IlliGAL Report, No. 97003, May 1997.

[20] 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.

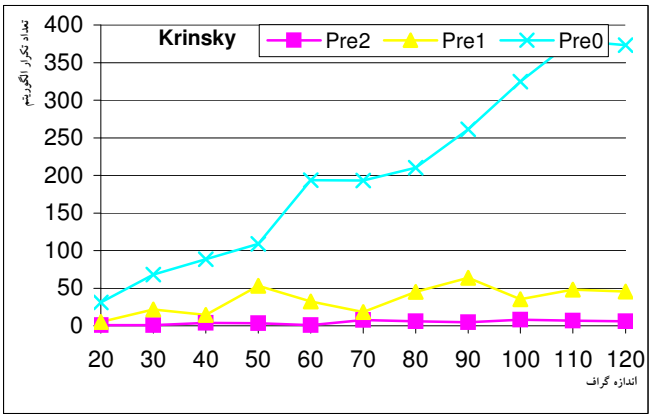
[21] Oommen, B. J. and Croix, E. V. "Graph Partitioning Using Learning Automata", IEEE Trans. On Computers, Vol. 45, No. 2, pp. 195-208, 1996.

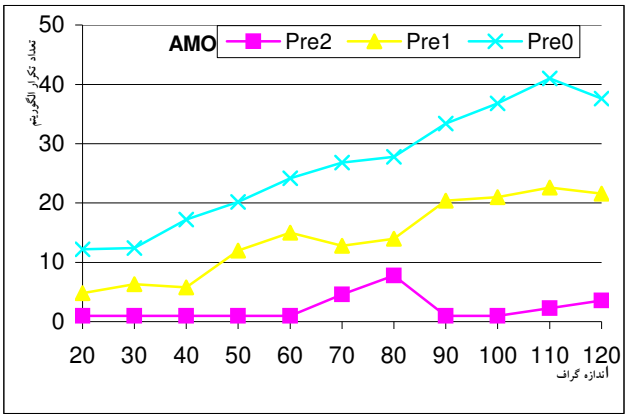
[22] Rezapourmirsaleh, R. and Meybodi, M. R. "A Hybrid Algorithm (GA+LA) for Graph Isomorphism", Proceedings of Information and Knowledge Technology (IKT2005), Tehran, Iran, May 2005.

[23] Rezapourmirsaleh, R., Solving Graph Isomorphism problem Using Learning Automata, M. S. Thesis, Computer and Information Technology Department, Amirkabir University, Teharn, Iran, 2004.

GA	LA				LA+GA (Tsetline)				LA+GA (Krylov)				LA+GA (Krinsky)				LA+GA (AMO)				الگوریتم
	Tsetline	Krylov	Krinsky	AMO	Pre 3	Pre 2	Pre 1	Pre 0	Pre 3	Pre 2	Pre 1	Pre 0	Pre 3	Pre 2	Pre 1	Pre 0	Pre 3	Pre 2	Pre 1	Pre 0	اندازه گراف
۹۰	۵۸۳,۲	۵۰۴,۸	۶۵۲	۵۶۹,۶	۱	۱	۵,۳	۹۰,۲	۱	۱	۵,۳	۳۶	۱	۱	۵,۴	۳۱,۲	۱	۱	۴,۸	۱۲,۲	۲۰
۲۶۲	۵۵۲,۸	۶۸۷,۴	۷۰۵,۴	۵۵۱	۱	۱	۴۵,۲	۵۷,۶	۱	۱	۳۴,۵	۸۸,۲	۱	۱	۲۲	۶۸,۲	۱	۱	۶,۳	۱۲,۴	۳۰
۲۷۳	۷۳۸	۵۰۷,۸	۶۹۷	۵۶۶	۱	۳,۱	۱۹,۸	۸۰,۶	۱	۳	۱۵,۸	۸۳,۴	۱	۴,۱	۱۴,۶	۸۸,۶	۱	۱	۵,۸	۱۷,۲	۴۰
۳۸۸	۶۶۳,۶	۷۳۰,۲	۵۳۲,۶	۵۳۲,۶	۱	۲,۸	۵۴,۶	۱۵۴,۶	۱	۶,۳	۵۲	۱۱۰	۱	۳,۵	۵۳,۴	۱۰۹,۲	۱	۱	۱۲	۲۰,۲	۵۰
۵۱۴	۵۹۹,۶	۴۹۰,۴	۸۵۹,۸	۵۵۸,۸	۱	۱	۴۵,۹	۱۵۱	۱	۱	۴۲,۶	۱۵۳,۸	۱	۱	۳۲,۵	۱۹۳,۸	۱	۱	۱۵	۲۴,۲	۶۰
۶۱۱	۷۱۵,۲	۵۰۸,۲	۵۱۹,۴	۵۶۰,۸	۱	۵,۱	۳۲,۵	۲۱۶,۶	۱	۵,۶	۲۸,۵	۲۰۹,۴	۱	۷,۹	۱۸,۲	۱۹۳,۲	۱	۴,۶	۱۲,۸	۲۶,۸	۷۰
۷۳۸	۵۱۶,۸	۸۱۶	۶۴۲,۸	۵۴۳	۱	۷,۲	۴۹,۷	۲۱۰,۴	۱	۸,۴	۳۵	۲۵۰,۴	۱	۶,۲	۴۵,۳	۲۱۰,۴	۱	۷,۸	۱۴	۲۷,۸	۸۰
۱۰۰۹	۸۸۶,۴	۵۰۸	۵۸۴	۵۵۶,۶	۱	۵	۵۴,۴	۲۵۹,۴	۱	۵,۲	۴۵,۲	۲۴۸	۱	۴,۸	۶۴	۲۶۱,۴	۱	۱	۲۰,۴	۳۳,۴	۹۰
۱۱۴۲	۴۱۱,۴	۸۲۰,۸	۸۹۴,۸	۵۴۲,۴	۱	۳	۲۹,۷	۳۱۳,۲	۱	۶,۸	۳۴	۳۰۳,۸	۱	۸,۴	۳۵,۳	۳۲۵	۱	۱	۲۱	۳۶,۸	۱۰۰
۱۴۰۷	۵۱۴,۸	۷۳۹,۶	۷۲۸,۶	۵۴۹,۴	۱	۵,۳	۴۰,۷	۳۷۴,۲	۱	۴,۸	۵۶,۷	۳۲۰	۱	۶,۸	۴۸	۳۷۹,۲	۱	۲,۳	۲۲,۶	۴۱	۱۱۰
۱۴۰۴	۶۴۳	۶۵۲,۲	۱۲۷۱	۴۷۰,۶	۱	۶	۶۲,۶	۴۴۷,۲	۱	۶,۶	۵۸	۳۹۶,۸	۱	۶,۲	۴۵,۸	۳۷۳,۴	۱	۳,۶	۲۱,۶	۳۷,۶	۱۲۰
۷۱۲,۵	۶۲۰,۴	۶۳۳,۲	۷۳۵,۲	۵۴۵,۵	۱	۳,۵	۴۰,۰۴	۲۱۴	۱	۳,۵	۳۷,۰۵	۱۹۹,۹	۱	۳,۴	۳۴,۹۵	۲۰۳	۱	۱,۴	۱۴,۲۱	۲۶,۳	میانگین

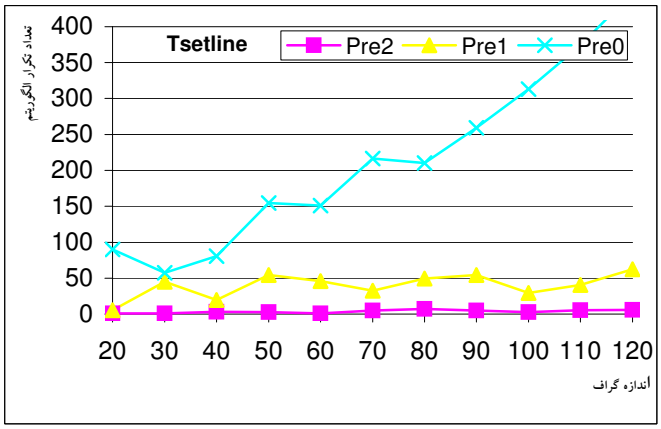
جدول ۱: تعداد تکرار مورد نیاز برای الگوریتم ترکیبی (GA+LA) مبتنی بر آتاماتاهای اومن، کرینسکی، کرایلو و ستلین در حالت بدون پیش پردازش(Pre0)، انجام یک، دو و سه بار پیش پردازش (Pre1، Pre2 و Pre3) و مقایسه آن با الگوریتم‌های آتاماتای یادگیر(LA) مبتنی بر آتاماتاهای اومن، کرینسکی، کرایلو و ستلین و الگوریتم ژنتیکی(GA)

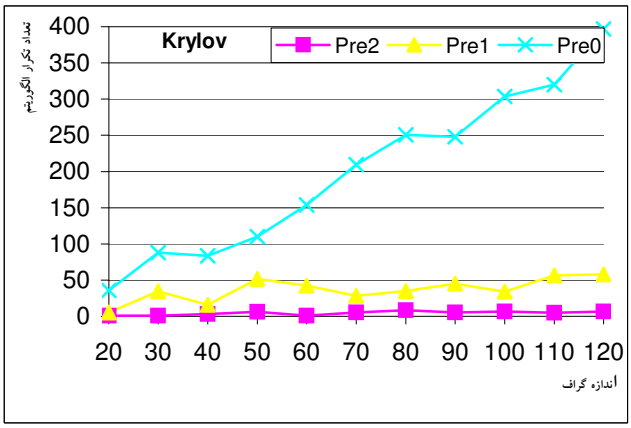




نمودار شماره۱: مقایسه تعداد تکرار الگوریتم‌های GA+LA مبتنی بر آتاماتای اومن در حالت بدون پیش پردازش(Pre0) و انجام یک و دو بار پیش پردازش(Pre2,Pre1)

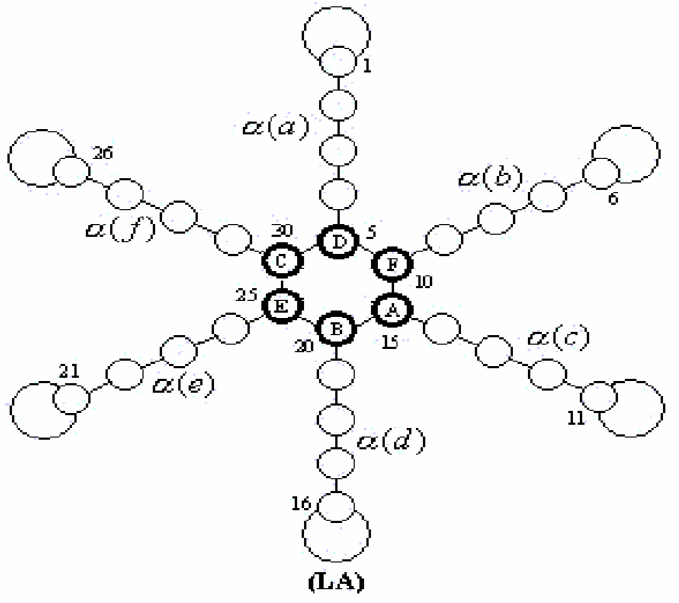
نمودار شماره۲: مقایسه تعداد تکرار الگوریتم‌های GA+LA مبتنی بر آتاماتای کرینسکی در حالت بدون پیش پردازش(Pre0) و انجام یک و دو بار پیش پردازش(Pre2,Pre1)





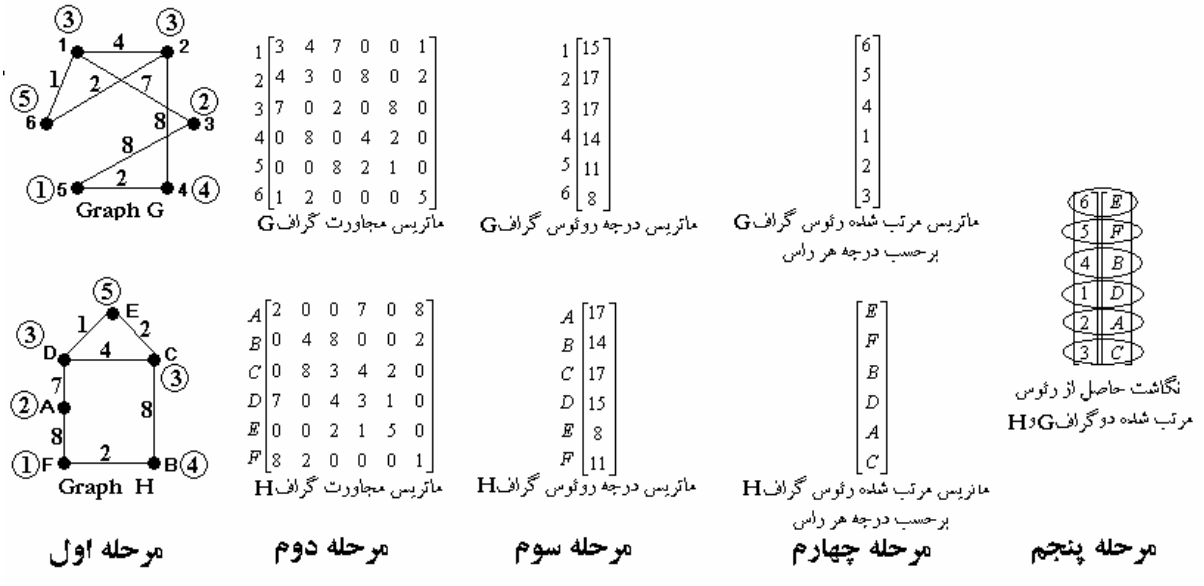
نمودار شماره۳: مقایسه تعداد تکرار الگوریتم‌های GA+LA مبتنی بر آتاماتای کرایلو در حالت بدون پیش پردازش(Pre0) و انجام یک و دو بار پیش پردازش(Pre2,Pre1)

نمودار شماره۴: مقایسه تعداد تکرار الگوریتم‌های GA+LA مبتنی بر آتاماتای ستلین در حالت بدون پیش پردازش(Pre0) و انجام یک و دو بار پیش پردازش(Pre2,Pre1)

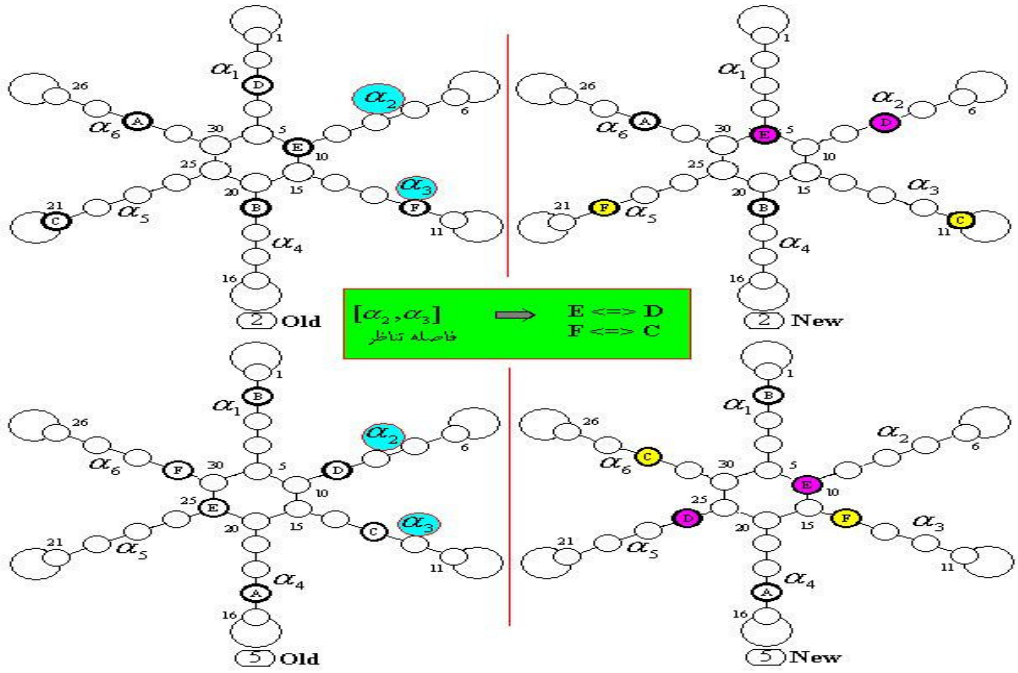


شکل ۱ - نمایش جایگشت

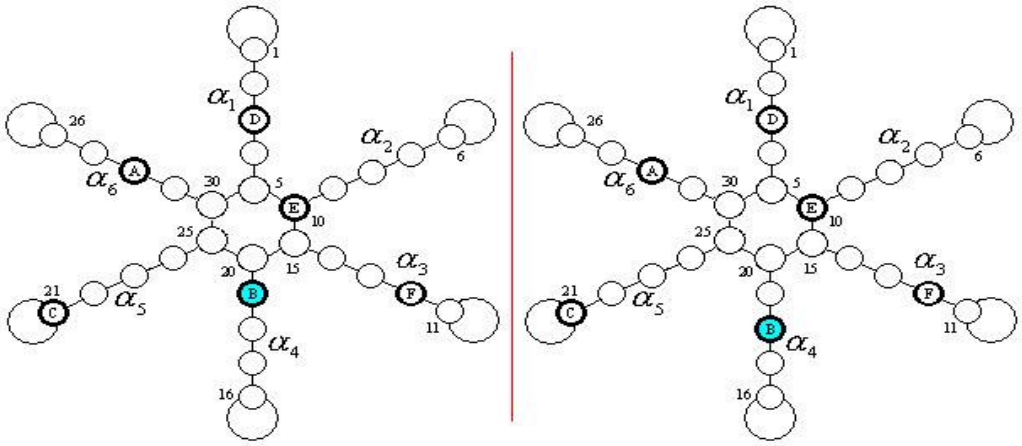
بوسیله آتاماتای یادگیر $\{(1,D),(2,F),(3,A),(4,B),(5,E),(6,C)\}$



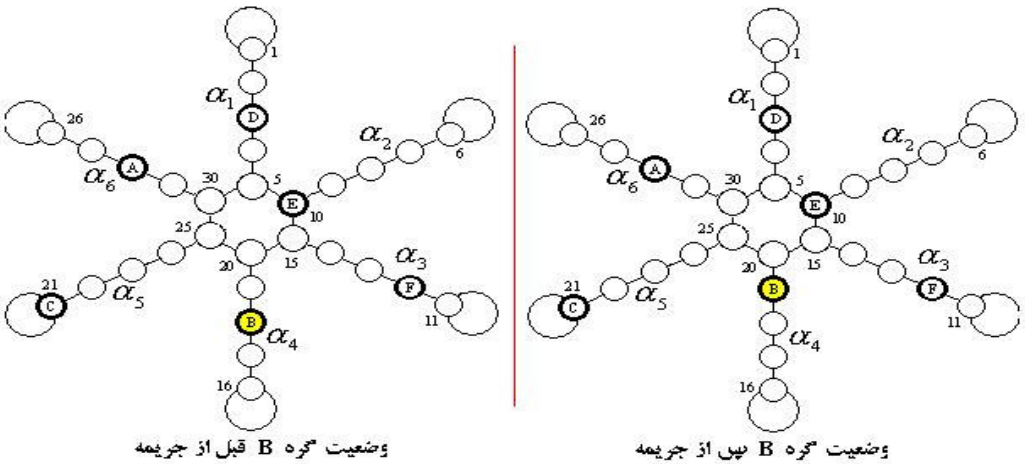
شکل ۲- نحوه بدست آوردن جایگشت تقریبی



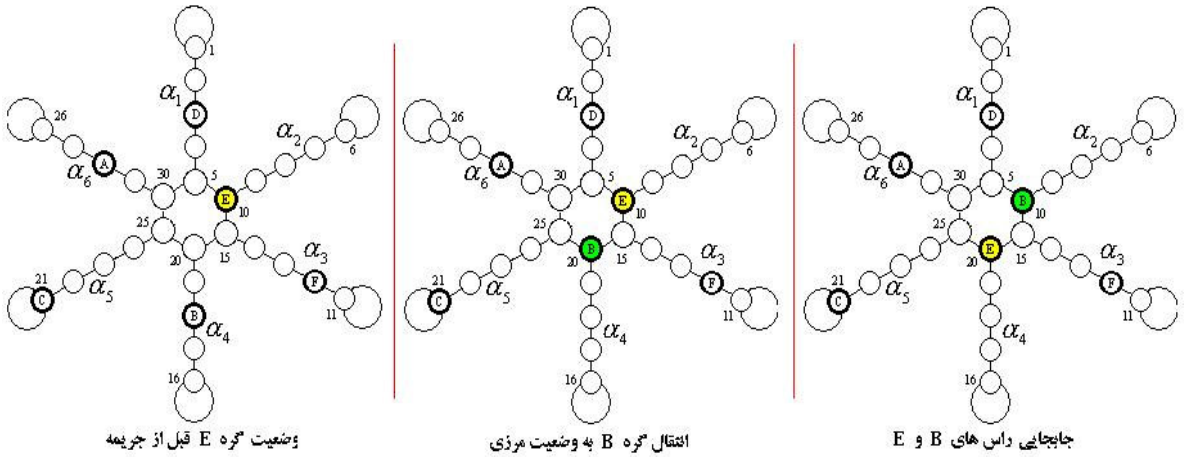
شکل۴: نحوه انجام عملگر جابجایی



شکل۵ : نحوه پاداش راس B



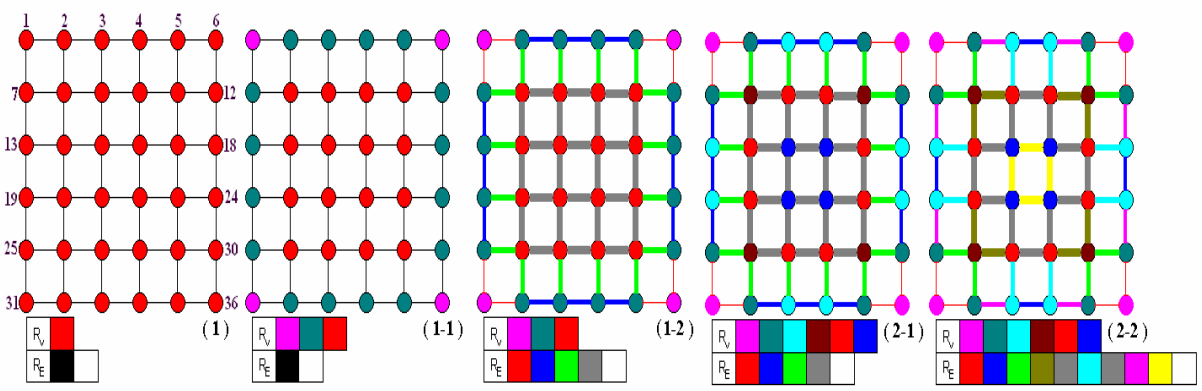
شکل ۶ : نحوه جریمه کردن یک راس که در وضعیتی غیر از وضعیت مرزی قرار داشته باشد



شکل۷ : نحوه جریمه کردن یک راس که در وضعیت مرزی قرار داشته باشد

```
Procedure  WeightingPreProcess(G1,G2)
Begin
For p = 0 to SizeOfGraph do
Begin
    For q = 0 to SizeOfGraph do
    Begin
        G1_New[p][q] = 0; G2_New[p][q] = 0;
            For k = 0 to SizeOfGraph do
            Begin
                  If( p=q ) then
                Begin
                  G1_New[p][q] = G1_New[p][q] + (G1_Old[k][p] + G1_Old[k][q] +
                                | G1_Old[p][k] + G1_Old[q][k] | );
                  G2_New[p][q] = G2_New[p][q]  + (G2_Old[k][p] + G2_Old[k][q] +
                                | G2_Old[p][k] + G2_Old[q][k] | );
                End
                Else
                  G1_New[p][q] = G1_New[p][q] + (G1_Old[k][p] + G1_Old[k][q] +
                                 G1_Old[p][k] + G1_Old[q][k] | )*G1_Old[p][q];
                  G2_New[p][q] = G2_New[p][q]  + (G2_Old[k][p] + G2_Old[k][q] +
                                | G2_Old[p][k] + G2_Old[q][k] | )*G2_Old[p][q];
                End if
          EndFor
    EndFor
EndFor
End  WeightingPreProcess
```

الگوریتم پیش پردازش



شکل ۸