ICEE2006 14th Iranian Conference on Electrical Engineering

# یک الگوریتم ترکیبی (الگوریتم ژنتیک+ اتوماتاهای یادگیر) برای حل مساله درخت اشتاینر

سمیرا نوفرستی
دانشکده مهندسی کامپیوتر و فناوری اطلاعات
دانشگاه صنعتی امیرکبیر
Samira_noferesty@yahoo.com

محمدرضا میبدی
دانشکده مهندسی کامپیوتر و فناوری اطلاعات
دانشگاه صنعتی امیرکبیر
mmeybodi@aut.ac.ir

**چکیده- مساله درخت اشتاینر یکی از مسایل NP-complete می‌باشد و بهمین دلیل الگوریتمهای تقریبی متعددی برای حل آن گزارش شده است. در این مقاله ابتدا یک الگوریتم تقریبی که از ترکیب الگوریتم ژنتیکی و اتوماتاهای یادگیر حاصل شده است برای حل مساله اشتاینر ایستا پیشنهاد می‌شود و سپس با استفاده از این الگوریتم، الگوریتمی برای حل مساله درخت اشتاینر پویا ارایه می‌گردد. به منظور نشان دادن کارایی الگوریتمهای پیشنهادی، این الگوریتمها بر روی گرافهای مجموعه B مسائل بیسلی اجرا و سپس نتایج بدست آمده با نتایج بدست آمده برای تعدادی از الگوریتمهای گزارش شده مقایسه گردیده است. نتایج ازمایشها کارایی الگوریتم پیشنهادی را نشان می‌دهد.**

کلیدواژه: اتوماتاهای یادگیر، الگوریتم ژنتیک، درخت اشتاینر

## ۱- مقدمه

مساله اشتاینر در گراف $G=(V,E)$ عبارت است از یافتن درختی با حداقل هزینه که زیرمجموعه $T$ از رئوس گراف به نام ترمینالها را به هم متصل کند. این درخت می‌تواند شامل رئوس دیگری به نام نقاط اشتاینر نیز باشد.

در بعضی از کاربردها (مانند شبکه‌های سیار سلولی)، بدلیل اضافه و یا حذف شدن گره‌ها ساختار گراف در طی زمان تغییر می‌کند. تغییرات می‌تواند در قسمتهای مختلف گراف بطور همزمان ایجاد شود. در این شرایط مساله درخت اشتاینر پویا مطرح می‌شود. مساله درخت اشتاینر پویا (DST)[1] شامل یک گراف به همراه یک تابع هزینه و یک رشته از درخواستهاست. هر درخواست می‌تواند متقاضی حذف یا افزودن یک گره در گراف باشد. هدف یافتن درخت اشتاینر بر روی گراف داده شده با توجه به درخواستهای حذف و یا اضافه می‌باشد.

مساله درخت اشتاینر در گراف NP-complete است و بنابراین اکثر الگوریتمهای گزارش شده برای حل آن الگوریتمهای تقریبی هستند که در بین آنها می‌توان به الگوریتم حریصانه [۱]، الگوریتم حریصانه پویا (DGA) [۲] و الگوریتم حریصانه وزن‌دار (WGA) [۳] اشاره کرد. از دیگر روشهای تقریبی کارا برای حل مساله اشتاینر الگوریتمهای تصادفی تکراری[2] هستند. اگرچه این الگوریتمها رسیدن به یک پاسخ بهینه را تضمین نمی‌کنند اما در اغلب موارد جوابهای تقریبی مناسبی را تولید می‌کنند. از جمله الگوریتمهای تصادفی تکراری می‌توان به الگوریتم ژنتیکی [۴،۵] و الگوریتم کلونی مورچه‌ها [۶] اشاره کرد.

اتوماتای یادگیر و الگوریتم ژنتیک دو روش کارا برای حل مسائل بهینه‌سازی می‌باشند. این دو روش دارای دو ویژگی مهم می‌باشند که عبارتند از: ۱- هر دو با استفاده از عملگرهای احتمالی سعی در یافتن ناحیه‌ای از فضای جستجو دارند که احتمال یافتن جواب بهینه در آن بسیار بالاست، ۲- هر دو برای هدایت فرایند جستجو تنها نیاز به ارزیابی تابع هدف دارند. در این مقاله ابتدا یک الگوریتم ترکیبی که از ترکیب الگوریتم ژنتیکی و اتوماتاهای یادگیر حاصل شده است برای حل مساله اشتاینر ایستا پیشنهاد

[1] Dynamic Steiner Tree

[2] iterative

می‌شود. سپس با استفاده از این الگوریتم، الگوریتمی برای حل مساله درخت اشتاینر پویا ارایه می‌گردد. نتایج آزمایشات انجام شده کارایی الگوریتمهای پیشنهادی را نشان می‌دهد.

ادامه مقاله بدین صورت سازماندهی شده است. در بخشهای ۲ و ۳ به ترتیب الگوریتم ژنتیکی و اتوماتاهای یادگیر به اختصار شرح داده می‌شود. در بخش ۴ الگوریتم پیشنهادی برای حل مساله اشتاینر ایستا و نتایج شبیه‌سازی‌ها ارائه می‌شود. در بخش ۵ ابتدا چگونگی استفاده از الگوریتم پیشنهادی برای حل مساله درخت اشتاینر پویا شرح داده می‌شود و سپس نتایج شبیه‌سازیها ارایه می‌گردد. بخش پایانی مقاله نتیجه‌گیری می‌باشد.

## ۲- الگوریتم ژنتیک

الگوریتمهای ژنتیک برای حل مسائل، بقاء افراد شایسته‌تر را شبیه‌سازی می‌کنند. هر نسل متشکل از تعدادی فرد است. هر فرد یک نقطه در فضای جستجو و در واقع یک راه‌حل از مساله را نشان می‌دهد. هر فرد با یک کروموزوم متناظر می‌گردد. کروموزوم یک رشته بیتی با طول ثابت است و بیتها در کروموزوم ژن نامیده می‌شوند. الگوریتم با یک مجموعه از راه‌حل‌های ارائه شده توسط کروموزومها شروع می‌شود که به آن نسل اول می‌گویند. از افراد نسل اول برای تولید نسلهای بعدی استفاده می‌شود. با تولید نسلهای جدید فرآیند حل مساله به سمت جواب بهینه همگرا می‌گردد. تولید افراد جدید به کمک عملگرهای ژنتیکی انجام می‌شود. راه‌حل‌هایی که برای تشکیل افراد جدید انتخاب می‌شوند باید نسبت به دیگر راه‌حل‌ها دارای شایستگی بیشتری باشند. در واقع کروموزومهای شایسته‌تر شانس بیشتری برای تولید نسل دارند. این روند تا زمانی که بعضی از شروط (از قبیل تعداد تکرارها یا بهبود بهترین راه‌حل) رضایتبخش باشد ادامه پیدا می‌کند. برای کسب اطلاعات بیشتر درباره الگوریتمهای ژنتیکی می‌توان به [۷،۸،۹] مراجعه کرد.

## ۳-
## توماتاهای یادگیر

اتوماتای یادگیر [۱۰] یک مدل انتزاعی است که می‌تواند تعداد محدودی عمل را انجام دهد. هر عمل انتخاب شده توسط محیطی تصادفی ارزیابی شده و پاسخی به اتوماتای یادگیر داده می‌شود. اتوماتای یادگیر از این پاسخ استفاده نموده و عمل خود را برای مرحله بعد انتخاب می‌کند.

**محیط**: محیط تصادفی را می‌توان توسط سه‌تایی $E \equiv \{\alpha, \beta, c\}$ تعریف نمود که $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه ورودی‌ها، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_m\}$ مجموعه خروجی‌ها و $c \equiv \{c_1, c_2, ... c_r\}$ مجموعه احتمال‌های جریمه شدن می‌باشند. هرگاه $\beta_i$ دو مقداری باشد $\beta_i = 1$ به عنوان جریمه و $\beta_i = 0$ به عنوان پاداش در نظر گرفته می‌شود. $c_i$ احتمال اینکه عمل $\alpha_i$ نتیجه نامطلوب داشته باشد می‌باشد.

اتوماتاهای یادگیر به دو گروه اتوماتای یادگیر با ساختار ثابت[۳] و اتوماتای یادگیر با ساختار متغیر[۴] تقسیم می‌شوند. در ادامه به شرح مختصری در باره اتوماتاهای یادگیر با ساختار متغیر که در این مقاله استفاده شده است می‌پردازیم.

**اتوماتاهای یادگیر با ساختار متغیر:** اتوماتای یادگیر با ساختار متغیر توسط ۵ تایی $LA \equiv \{\alpha, \beta, p, T, c\}$ نشان داده می‌شود که $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه عمل‌های اتوماتای یادگیر، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_m\}$ مجموعه ورودی‌های اتوماتای یادگیر، $p \equiv \{p_1, p_2, ..., p_r\}$ بردار احتمال انتخاب عملها، $T \equiv p(n+1) = T[\alpha(n), \beta(n), p(n)]$ الگوریتم یادگیری و $c \equiv \{c_1, c_2, ... c_r\}$ جریمه هر عمل می‌باشند. اگر اتوماتای یادگیر عمل $\alpha_i$ در مرحله $n$ام انتخاب شود و پاسخ مطلوب از محیط دریافت نماید، احتمال $p_i(n)$ افزایش یافته و سایر احتمال‌ها کاهش می‌یابند و برای پاسخ نامطلوب احتمال $p_i(n)$ کاهش یافته و سایر احتمال‌ها افزایش می‌یابند. تغییرات به گونه‌ای صورت می‌گیرد تا حاصلجمع $p_i(n)$ ها همواره ثابت و مساوی یک باقی بماند.

الف- پاسخ مطلوب

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

ب- پاسخ نامطلوب

ا

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

در روابط فوق، $a$ پارامتر پاداش و $b$ پارامتر جریمه می‌باشد. با توجه به مقادیر $a$ و $b$ سه حالت مختلف را می‌توان در نظر

---

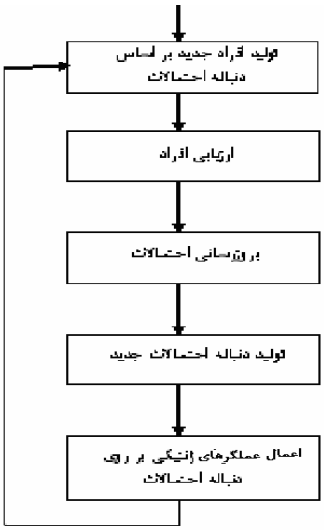
[۳] Fixed Structure
[۴] Variable Structure

گرفت. زمانیکه $a$ و $b$ با هم برابر باشند، الگوریتم را[5] $L_{RP}$، زمانیکه $b$ از $a$ خیلی کوچکتر باشد، الگوریتم را $L_{R\varepsilon P}$ [6] و زمانیکه $b$ مساوی صفر باشد، الگوریتم را $L_{RI}$ [7] می‌نامند.

## ۴- الگوریتم پیشنهادی برای حل مساله اشتاینر ایستا

الگوریتم پیشنهادی (LGA ) از ترکیب الگوریتمهای ژنتیکی و اتوماتاهای یادگیر بدست می‌آید. وظیفه اتوماتاهای یادگیر تعیین مقدار ژنها در کروموزومها می‌باشد. هر کروموزوم با یک بردار دودویی با اندازه تعداد گره‌های گراف نمایش داده می‌شود که مقدار ژن $i$ام حضور و یا عدم حضور گره $i$ام در درخت اشتاینر را مشخص می‌کند. هر ژن از یک کروموزوم به یک اتوماتای یادگیر با ساختار متغیر با دو عمل مجهز شده است. دو عمل اتوماتای یادگیر یک ژن، انتخاب و عدم انتخاب گره متناظر با آن ژن برای حضور در درخت اشتاینر می‌باشند. اگر فرض شود که $p_{i1}$ و $p_{i2}$ به ترتیب احتمالهای اعمال اول و دوم اتوماتای یادگیر متناظر با ژن $i$ام باشند در این صورت هر کروموزوم با بردار $(p_{i1}, p_{12}, p_{i3},\ldots,p_{in})$ نشان داده می‌شود. عضو $i$ام این بردار احتمال یک بودن ژن متناظر با این عضو می‌باشد. در واقع عضو $i$ام این بردار احتمال انتخاب اقدام اول ("انتخاب گره") اتوماتای یادگیر متناظر با این عضو می‌باشد. در شروع الگوریتم مقادیر این احتمالات یکسان و برابر ۰/۵می‌باشند. بروز کردن احتمالات اعمال اتوماتاهای یادگیر باعث بروز در آمدن بردار احتمالات ژنها می‌گردد. افراد یک نسل با توجه به بردار احتمالات بهنگام شده افراد نسل قبل انتخاب می‌شوند. پس از تولید افراد یک نسل شایستگی آنها ارزیابی شده و بر اساس این شایستگی بردارهای احتمالات اعمال اتوماتاهای یادگیر بروز می‌شود. برای تولید یک نسل جدید (دنباله احتمالات جدید) از عملگرهای ژنتیکی انتخاب[8]، دورگه شدن[9] و تمایز[10] استفاده می‌شود.

چرخه الگوریتم پیشنهادی مطابق شکل ۱ است. الگوریتم با یک مجموعه از راه‌حل‌های ارائه شده توسط کروموزومها شروع می‌شود که به آن نسل اول می‌گوییم. بردار احتمالات کلیه افراد نسل اول یکسان می‌باشد. بدین صورت که در هر کروموزوم احتمال انتخاب گره‌های غیرترمینال برابر ۰.۵ و احتمال انتخاب ترمینالها برابر ۱ قرار داده می‌شود. این کار باعث می‌شود افراد نسل اول به صورت تصادفی انتخاب شوند. برای ایجاد یک فرد از نسل اول، اتوماتاهای یادگیر کروموزوم متناظر با آن به طور همزمان فعال شده و هر کدام بر اساس بردار احتمال اقدامهای متناظر با آنها یک عمل را انتخاب می‌کنند. سپس با استفاده از الگوریتم پریم، بر روی گره‌هایی که اتوماتاهای یادگیر آنها عمل حضور در درخت را انتخاب کرده‌اند یک درخت پوشای کمینه ایجاد می‌شود. در پایان درخت حاصل با حذف گره‌های غیرترمینالی که در برگهای درخت واقع شده‌اند هرس می‌شود.




شکل ۱ : چرخه الگوریتم ژنتیک یادگیر

برای تولید افراد نسلهای بعدی، در ابتدا مانند نسل اول بر اساس بردار احتمالات فرد $i$ام(کروموزوم $i$ ام) نسل فعلی، یک فرد (کروموزوم) جدید ایجاد می‌شود. اگر شایستگی فرد جدید نسل فعلی از شایستگی فرد $i$ام نسل قبلی کمتر باشد فرد $i$ام نسل قبلی در نسل جدید باقی می‌ماند و در غیر این صورت فرد جدید جایگزین فرد $i$ام نسل فعلی می‌گردد. برای تولید افراد شایسته‌تر از عملگرهای ژنتیکی استفاده می‌شود. عملگرهای ژنتیکی این الگوریتم به شرح زیر است.

**انتخاب:** اگر جمعیت یک نسل $p$ باشد، فرد $i$ام و $i+1$ام هر نسل که $i = 1,2,\ldots,\frac{p+1}{2}$ و نیز فرد $j$ام و $p+1-j$ام که $j = 1,2,\ldots,\frac{p}{2}$ است برای عمل دورگه شدن انتخاب می‌شوند.

---

[5] Linear Reward Penalty
[6] Linear Reward Epsilon Penalty
[7] Linear Reward Inaction
[8] selection
[9] crossover
[10] differentiation

**دورگه شدن:** دو فرد $A$ و $B$ به عنوان والدین انتخاب می‌شوند. فرض می‌شود $A$ شایسته‌تر از $B$ است. یک گره مشترک در $A$ و $B$ انتخاب می‌شود. سپس آن گره از طریق کوتاهترین مسیر به گره‌های انتخاب شده در $A$ و $B$ متصل می‌گردد. اگر درخت حاصل از درخت راه‌حل $A$ بهتر بود، یک فرزند ایجاد می‌شود که مقدار هر بیت در آن برابر میانگین احتمال بیتهای متناظر در والدین است. این فرزند بجای راه‌حل A قرار می‌گیرد.

**دگردیسی:** دو نوع دگردیسی به نامهای تغییر و تعویض در نظر گرفته شده است که احتمال انتخاب آنها یکسان می‌باشد. در تغییر یک ژن صفر فرد به تصادف انتخاب و مقدار آن برابر ۱ قرار داده می‌شود. در تعویض به صورت تصادفی مقدار یک ژن غیرترمینال با دیگری تعویض می‌شود. احتمال انتخاب عملگر دگردیسی ۳۰ درصد است.

**تمایز افراد مشابه:** اگر در یک نسل افراد تکراری باشد، عمل دگردیسی آنقدر تکرار می‌شود تا افراد متمایز شوند.

بعد از ایجاد یک نسل، شایستگی افراد آن ارزیابی می‌شود. شایستگی هر فرد برابر معکوس هزینه درخت اشتاینر بدست آمده در نظر گرفته شده است. بر اساس این شایستگی، بردار احتمالات کروموزوم بروز می‌شود. بروزرسانی احتمالات افراد یک نسل بر اساس الگوریتم یادگیری اتوماتای یادگیر انجام می‌گیرد. اگر الگوریتم یادگیری $L_{RI}$ باشد بردار احتمالات یک کروموزوم بر اساس روابط زیر انجام می‌پذیرد.

$$P_i(n+1) = P_i(n) + a\beta(n)(1 - P_i(n)) \quad (۱)$$

اگر بیت موقعیت $i$ در نسل $n$ام برابر یک باشد

$$P_i(n+1) = P_i(n) - b\beta(n)P_i(n)$$

اگر بیت موقعیت $i$ در نسل $n$ام برابر صفر باشد.

که $a$ پارامتر پاداش و $b$ پارامتر جریمه می‌باشد و $\beta(n)$ با توجه به شایستگی فرد تعیین می‌شود. مقدار $\beta(n)$ برای فرد $j$ام در نسل $n$ام با توجه به فرمول زیر تعیین می‌شود.

$$Bj(n) = \frac{fj(n) - \min(f)}{\max(f) - \min(f)} \quad (۲)$$

که $min(f)$ و $max(f)$ به ترتیب کمترین و بیشترین شایستگی افراد در نسل فعلی را نشان می‌دهند. $f_j(n)$ نیز شایستگی فرد $j$ام نسل فعلی است. فرد $i$ام یک نسل در صورتی پاداش می‌گیرد که شایستگی آن از شایستگی فرد $i$ام نسل قبل بهتر باشد. پس از ارزیابی، افراد یک نسل بر اساس شایستگی مرتب می‌شوند به طوری که اولین فرد شایسته‌ترین فرد باشد. برای تولید بردار احتمالات جدید از عملگر ژنتیکی دگردیسی استفاده می‌شود. در این عمل که با احتمال پایین انتخاب می‌شود یک بیت غیرترمینال به تصادف انتخاب و مقدار آن برابر ۰.۵ قرار می‌گیرد. برای بهبود نتایج، بر روی افراد نسل آخر عمل کاهش انجام می‌گیرد. این عمل برای هر یال از درخت به صورت زیر انجام می‌شود. با حذف یال از درخت، دو زیردرخت مجزا حاصل می‌شود. با اجرای الگوریتم جستجوی عمق اول[11] بر روی هر زیردرخت نقاط اشتاینری که اضافه هستند، حذف می‌شوند. با استفاده از الگوریتم دایکسترا کوتاهترین مسیر بین گرههای دو زیردرخت پیدا می‌شود. اگر مجموع هزینه‌های دو زیردرخت به علاوه هزینه مسیر یافت شده کمتر از هزینه درخت قبلی باشد دو زیردرخت از طریق کوتاهترین مسیر متصل شده و درخت حاصل به عنوان درخت فعلی در نظر گرفته می‌شود.

**نتایج شبیه‌سازی**: برای بررسی کارایی روش پیشنهادی، الگوریتم ژنتیک (GA) گزارش شده در [۴] و الگوریتم ژنتیک یادگیر (LGA) در حل مساله اشتاینر بر روی گرافهای مجموعه B از دسته مسائل بیسلی[12][۱۱] آزمایش شده‌اند. در آزمایش اول الگوریتم ژنتیک یادگیر با سه الگوریتم یادگیری $L_{RP}$، $L_{R\&P}$ و $L_{RI}$ پیاده‌سازی شده است. در هر سه روش مقدار $a$ در رابطه (۱) برابر ۰/۰۵ و در الگوریتم $L_{R\&P}$ مقدار $b$ برابر ۰/۰۱ در نظر گرفته شده است. میانگین ۱۰ بار اجرای این الگوریتمها بر روی گرافهای مجموعه B مسائل بیسلی نشان می‌دهد که اتوماتای یادگیر $L_{RP}$ به نتایج بهتری دست یافته است.

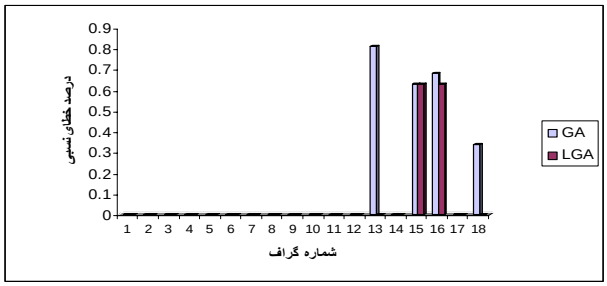
در آزمایش دوم الگوریتم ژنتیک یادگیر که از الگوریتم یادگیری $L_{RP}$ استفاده می‌کند با الگوریتم ژنتیک مقایسه می‌گردد. میانگین درصد خطای نسبی این الگوریتمها بر روی ۱۰ اجرا در شکل ۲ ارائه شده است. برای محاسبه خطای نسبی از رابطه زیر استفاده می‌شود:

$$relative-error = \frac{C - C_{opt}}{C_{opt}} \quad (۳)$$

که $C$ هزینه درخت اشتاینر حاصل از الگوریتم و $C_{opt}$ هزینه درخت اشتاینر بهینه می‌باشد. تعداد تکرارها در هر بار اجرای الگوریتمها برابر تعداد گره‌های گراف مورد بررسی در نظر

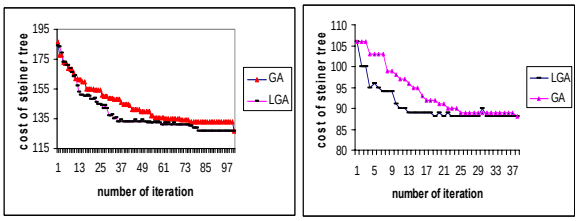
[11] Depth First Search
[12] Beasly

گرفته شده است. همانطور که مشاهده می‌شود الگوریتم ژنتیک یادگیر به نتایج بهتری دست یافته است.




شکل ۲: مقایسه درصد خطای نسبی الگوریتمهای GA و LGA

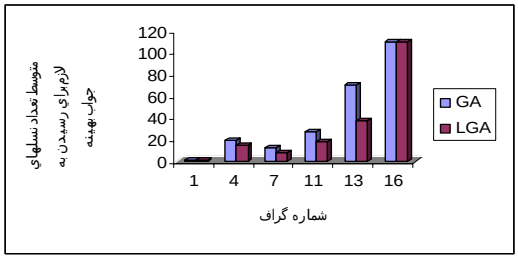
روند همگرایی به پاسخ بهینه در الگوریتم ژنتیک و الگوریتم ژنتیک یادگیر برای گراف شماره ۱۱ و گراف شماره ۱۶ از مجموعه B مسائل بیسلی در شکل ۳ نشان داده شده است. دو الگوریتم بارها اجرا شده‌اند تا هزینه بهترین فرد نسل اول در آنها یکسان شود. همان طور که مشاهده می‌شود الگوریتم ژنتیک یادگیر از سرعت همگرایی بهتری برخوردار است.




الف) گراف شماره ۱۱ ب) گراف شماره ۱۶

شکل ۳: مقایسه روند همگرایی به جواب در الگوریتم GA و LGA

آزمایش دیگری به منظور مقایسه زمان اجرای الگوریتم GA و الگوریتم LGA انجام گردید. این دو الگوریتم بر روی تعدادی از گرافهای مجموعه B مسائل بیسلی اجرا شدند و متوسط تعداد نسلهای لازم برای رسیدن به جواب بهینه محاسبه گردید. نتیجه این آزمایش در شکل ۴ آمده است. همانطور که مشاهده می‌شود الگوریتم LGA برای رسیدن به جواب بهینه نیاز به تعداد تکرارهای کمتری دارد.




شکل ۴: مقایسه متوسط تعداد نسلهای لازم برای رسیدن به جواب بهینه

در جدول ۱ درصد خطای نسبی روش پیشنهادی با درصد خطای نسبی الگوریتم کلونی مورچه‌ها [۶] و تعدادی از الگوریتمهای حل مساله اشتاینر که در [۴] معرفی شده‌اند، مقایسه شده است. جزئیات الگوریتمهای ADH [13] و SPH [14] در مرجع [۱۲] آمده است. تعداد تکرار الگوریتم LGA برابر تعداد یالهای گراف در نظر گرفته شده است.

جدول ۱: مقایسه الگوریتم LGA و تعدادی از الگوریتمهای گزارش شده برای حل مساله اشتاینر

| شماره گراف | هزینه بهینه | SDG | SPH | ADH | ACS | LGA |
|---|---|---|---|---|---|---|
| ۱ | ۸۲ | ۰ | ۰ | ۰ | ۰ | ۰ |
| ۲ | ۸۳ | ۸۴/۳ | ۰ | ۰ | ۰ | ۰ |
| ۳ | ۱۳۸ | ۱/۴۵ | ۰ | ۰ | ۰ | ۰ |
| ۴ | ۵۹ | ۸/۴۳ | ۵/۰۸ | ۵/۰۸ | ۲/۷ | ۰ |
| ۵ | ۶۱ | ۴/۹۲ | ۰ | ۰ | ۰ | ۰ |
| ۶ | ۱۲۲ | ۴/۹۲ | ۳/۲۸ | ۱/۶۴ | ۱/۸ | ۰ |
| ۷ | ۱۱۱ | ۰ | ۰ | ۰ | ۰ | ۰ |
| ۸ | ۱۰۴ | ۰ | ۰ | ۰ | ۰ | ۰ |
| ۹ | ۲۲۰ | ۲۷/۲ | ۰ | ۰ | ۰/۰۵ | ۰ |
| ۱۰ | ۸۶ | ۱۳/۹۵ | ۴/۶۵ | ۴/۶۵ | ۳/۸۴ | ۰ |
| ۱۱ | ۸۸ | ۲/۲۷ | ۲/۲۷ | ۲/۲۷ | ۱/۵۹ | ۰ |
| ۱۲ | ۱۷۴ | ۰ | ۰ | ۰ | ۰/۱۱ | ۰ |
| ۱۳ | ۱۶۵ | ۶/۰۶ | ۷/۸۸ | ۲/۲۴ | ۰ | ۰ |
| ۱۴ | ۲۳۵ | ۱/۲۸ | ۲/۵۵ | ۰/۴۳ | ۰/۰۹ | ۰ |
| ۱۵ | ۳۱۸ | ۲/۲۰ | ۰ | ۰ | ۰ | ۰ |
| ۱۶ | ۱۲۷ | ۷/۸۷ | ۳/۱۵ | ۰ | ۰/۱۶ | ۰ |
| ۱۷ | ۱۳۱ | ۳۴/۵ | ۳/۸۲ | ۳/۰۵ | ۱/۲۲ | ۰ |
| ۱۸ | ۲۱۸ | ۴/۵۹ | ۱/۸۳ | ۰ | ۱/۴۲ | ۰ |

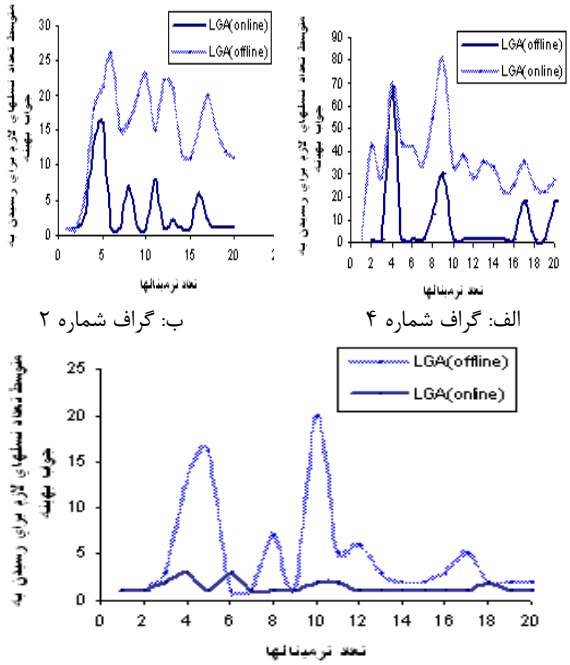
## ۵- الگوریتم ژنتیک یادگیر پویا

مساله درخت اشتاینر پویا شامل یک گراف و یک رشته از درخواستها است که هر درخواست می‌تواند متقاضی حذف یا افزودن یک گره در گراف و یا تغییر وزن یک یا چند یال باشد. هدف یافتن درخت اشتاینر بر روی گراف داده شده با توجه به تغییرات درخواستی می‌باشد. برای بدست آوردن درخت اشتاینر بعد از تغییرات، الگوریتم ژنتیک یادگیر پس از هر تغییر در گراف به این صورت عمل می‌کند که نسل تولید شده قبل از تغییرات به عنوان اولین نسل ساخت درخت جدید در نظر گرفته می‌شود و سپس درخت اشتاینر جدید با شروع از این نسل و در نظر گرفتن تغییرات ایجاد شده محاسبه می‌شود. اگر با توجه به تغییرات داده شده در

[13] Average Distance Heuristic
[14] Shortest Path Heuristic

آخرین نسل نمونه قبلی افراد تکراری وجود داشت آنها را حذف می‌کنیم. اگر به دلیل حذف افراد تکراری تعداد افراد نسل از حد مشخص شده کمتر بود تعدادی فرد تصادفی نیز تولید می‌شود. نشان داده شده است که تعداد تکرارهای مورد نیاز بسیار کمتر از موقعی می‌باشد که الگوریتم ژنتیک یادگیر از ابتدا بر روی گراف جدید اجرا شود. برای نشان دادن این امر گرافهای شماره ۱، ۳، ۹، ۱۵ و ۱۸ از مجموعه B مسائل بیسلی مورد بررسی قرار گرفته‌اند. در هر گراف تغییراتی به میزان ۲٪، ۵٪ و ۱۰٪ تعداد گره‌های گراف ایجاد شده است. سپس الگوریتم ژنتیک یادگیر پویا برای هر درخواست اجرا شده است. نتایج نشان می‌دهد بعد از ۱۰ تکرار برای هر درخواست می‌توان به نتایج حاصل از اجرای مجدد الگوریتم ایستا با تکرارهای زیاد دست یافت.

برای مقایسه زمان اجرای الگوریتم ژنتیک یادگیر ایستا و پویا نمودار متوسط تعداد نسلهای لازم برای رسیدن به جواب بهینه برحسب تعداد ترمینالها برای گرافهای شماره ۲، ۴ و ۱۱ از مجموعه B مسائل بیسلی در شکل ۵ نشان داده شده است. همانطور که مشاهده می‌شود الگوریتم LGA پویا در تعداد تکرارهای بسیار کمتری نسبت به حالت ایستا به جواب می‌رسد. برای آزمایشهای بیشتر می‌توان به [۱۳] مراجعه کرد.




الف: گراف شماره ۴ ب: گراف شماره ۲

ج: گراف شماره ۱۱

شکل ۵: متوسط تعداد نسلهای لازم برای رسیدن به بهترین جواب در الگوریتم LGA ایستا و پویا

## ۶- نتیجه گیری

در این مقاله ابتدا یک الگوریتم ترکیبی به نام الگوریتم ژنتیک یادگیرکه از ترکیب الگوریتم ژنتیکی و اتوماتاهای یادگیر حاصل شده است برای حل مساله اشتاینر ایستا پیشنهاد گردید. سپس با استفاده از این الگوریتم، الگوریتم دیگری برای حل مساله درخت اشتاینر پویا ارایه شد. به منظور نشان دادن کارایی الگوریتمهای پیشنهادی، این الگوریتمها بر روی گرافهای مجموعه B مسائل بیسلی اجرا گردید و نتایج بدست آمده با نتایج بدست آمده از تعدادی از الگوریتمهای گزارش شده مقایسه گردید. نتایج آزمایشها نشان از کارایی بالای الگویتمهای پیشنهادی داشت.

## مراجع


[1] Tsai,Y. T., Tang, Ch., and Chen, Y. Y. An Average Case Analysis of a Greedy Algorithm for the On-Line Steiner Tree Problem. *Computre Math. Applic.*, Vol. 31, No. 11, 121-131, 1996.

[2] Chiu, L. K. Comp 670K (Online Algorithms) Final Report On Steiner trees. www.cs.ust.hk/faculty/rudolf/Courses/Online03/chiu_report.pdf, December 2003.

[3] Waxman, B. M. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, Vo. 6, No. 9, 1611–1622, December 1988.

[4] Ding, S. and Ishii, N. An Online Genetic Algorithm for Dynamic Steiner Tree Problem. *Symposium on Computational Geometry*, 337-343, 1995.

[5] Esbensen, H. Computing Near-Optimal Solutions to the Steiner Problem in a Graph Using a Genetic Algorithm. *Networks*, Vol. 26, 173-185, 1995.

[۶] م. تشکری هاشمی، پ. ادیبی، ع. جهانیان، و ع. نوراله، حل مساله درخت اشتاینر پویا به کمک سیستم کولونی مورچه‌ها ، **در مجموعه مقالات نهمین کنفرانس سالانه انجمن کامپیوتر ایران**، ۱۳۸۲.

[7] Genetic Algorithms Archive, *Repository for GA Related Informatio*, http://www.aic.nrl.navy.mil/galist.

[8] Genetic Algorithms Tutorials, http://geneticalgorithms.ai-depot.com/Tutorials.html, July 2002.

[9]Genetic Programming Notebook, http://www.geneticprogramming.com.

[10] Sutton, R. S., and Barto, A. G. Reinforcement Learning: An Introduction. Cambridge, *MIT Press*, 1998.

[11] Beasley, J. E. OR-Library: Distributing Test Problems by Electronic Mail. *Operational Research. SOC.* , Vol. 41, No. 11, 1096-1072, 1990.

[12] Rayward-smith, V. J., and Clare, A.. On Finding Steiner vertices. *Networks*, Vol. 16, 283-294, 1986.

[۱۳] س. نوفرستی، م. ر. میبدی، **الگوریتمهای ترکیبی برای حل مساله درخت اشتاینر**، گزارش فنی، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران، ایران، ۱۳۸۴.