علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۶، شماره ۳ (الف)، ۱۳۸۷
صفحات ۵۳-۵۹
مقاله عادی

یک الگوریتم جدید بر مبنای اتوماتاهای یادگیر برای حل مسئله درخت اشتاینر

سمیرا نوفرستی[۱] محمدرضا میبدی[۲]

[۱]دانشکده مهندسی برق و کامپیوتر، دانشگاه سیستان و بلوچستان، زاهدان، ایران
[۲]دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران، ایران

چکیده

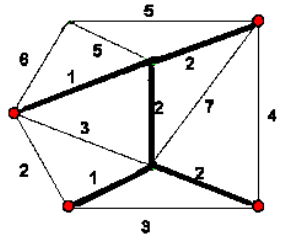
مسئله درخت اشتاینر در گراف عبارت است از پیدا کردن کم هزینه‌ترین درختی که شامل تعدادی گره خاص به نام ترمینال باشد. این مسئله از جمله مسائل NP-hard است و به همین دلیل الگوریتم‌های تقریبی متعددی برای حل آن پیشنهاد شده است. اغلب این الگوریتم‌ها درخت‌های اشتاینر نزدیک به بهینه تولید می‌کنند اما از سرعت همگرایی مناسبی برخوردار نیستند. در این مقاله یک الگوریتم تکرار شونده مبتنی بر اتوماتاهای یادگیر برای حل مسئله اشتاینر ایستا و پویا پیشنهاد می‌شود. نتایج شبیه‌سازی‌های انجام گرفته کارایی الگوریتم پیشنهادی را هم از لحاظ کیفیت جواب‌های تولید شده و هم از لحاظ سرعت همگرایی به جواب در مقایسه با الگوریتم‌های گزارش شده نشان می‌دهد.

کلمات کلیدی: درخت اشتاینر ایستا، درخت اشتاینر پویا، اتوماتاهای یادگیر، مسایل مشکل.

۱- مقدمه

برای پیاده‌سازی انتقال چندپخشی در شبکه‌های کامپیوتری ایستا و شبکه‌های کامپیوتری سیار نیاز به ساخت درخت‌های چندپخشی می‌باشد. یکی از متداولترین راههای ساخت درخت‌های چندپخشی استفاده از الگوریتم‌های ساخت درخت اشتاینر[۱] در گراف است.

مسئله درخت اشتاینر در گراف به صورت زیر تعریف می‌شود: گراف $G(V,E)$ را در نظر بگیرید که V مجموعه گره‌ها و $E \subseteq V*V$ مجموعه یال‌های گراف را نشان می‌دهد. به هر یال (i,j) در گراف یک هزینه $c(i,j)$ نسبت داده می‌شود. یک زیرمجموعه از گره‌ها به نام T نیز به عنوان مجموعه ترمینال‌ها تعریف می‌شود. هدف مسئله اشتاینر یافتن درختی در گراف است که مجموعه ترمینال‌ها را در برگیرد و هزینه آن حداقل باشد. این درخت، درخت اشتاینر کمینه نامیده می‌شود. اگر درخت حاصل فقط شامل ترمینال‌ها باشد، درخت پوشای کمینه[۲] نامیده می‌شود و الگوریتم‌های چندجمله‌ای متعددی مانند الگوریتم پریم[۳] برای حل آن وجود دارد. با این وجود در حالت کلی برای کاهش هزینه درخت از گره‌های غیرترمینال نیز استفاده می‌شود. این گره‌ها نقاط اشتاینر نامیده می‌شوند. مثالی از درخت اشتاینر در یک گراف نمونه در شکل ۱ نشان داده شده است. در این شکل گره های ترمینال و یال‌های درخت اشتاینر به صورت پررنگ رسم شده‌اند.



شکل ۱- درخت اشتاینر در گراف

در شبکه‌های سیار، توپولوژی شبکه در طی زمان به دلیل اضافه و یا حذف شدن گره‌ها تغییر می‌کند که در این شرایط مسئله درخت اشتاینر پویا مطرح می‌شود. در این موارد هدف یافتن یک درخت اشتاینر بعد از تغییرات انجام گرفته

در گراف می‌باشد. ساخت درخت‌های چندپخشی در شبکه‌های سیار به مراتب دشوارتر از ساخت این درخت‌ها در شبکه‌های ایستا می‌باشد. یک الگوریتم کارا برای ساخت درخت‌های چندپخشی در شبکه‌های سیار بایستی بتواند درخت‌های مناسب را سریع تولید کند حتی اگر چند تغییر همزمان در گروه چندپخشی رخ دهد.

به دلیل اهمیت مسئله درخت اشتاینر تاکنون الگوریتم‌های متعددی برای حل آن گزارش شده است. الگوریتم‌های دقیق که معمولاً از روشهای برنامه‌نویسی پویا و برش‌وانشعاب استفاده می‌کنند [۱، ۲، ۳] در بدترین حالت دارای پیچیدگی نمایی هستند و برای استفاده در کاربردهای عملی مناسب نمی‌باشند. به همین دلیل الگوریتمهای تقریبی متعددی برای حل مسئله اشتاینر در گراف گزارش شده است [۴، ۵، ۶].

تلاشهای زیادی نیز در جهت حل تقریبی مسئله اشتاینر پویا انجام گرفته است [۷، ۸]. در سال‌های اخیر نیز الگوریتم‌های تکرار شونده مانند الگوریتم ژنتیکی [۹] و الگوریتم کلونی مورچه‌ها [۱۰] برای حل مسئله درخت اشتاینر ایستا و پویا پیشنهاد شده است. در این روش‌ها رسیدن به یک پاسخ بهینه تضمین نمی‌شود اما در اغلب موارد جواب‌های تقریبی قابل قبولی تولید می‌کنند.

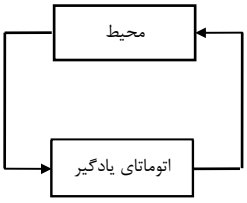
در این مقاله ابتدا یک الگوریتم تکرارشونده مبتنی بر اتوماتاهای یادگیر برای حل مسئله اشتاینر ایستا پیشنهاد می‌شود و سپس با استفاده از این الگوریتم، الگوریتم دیگری برای حل مسئله اشتاینر پویا ارایه می‌گردد. در الگوریتم حل مسئله اشتاینر ایستا، هر گره غیرترمینال از گراف به یک اتوماتای یادگیر مجهز است. هر اتوماتای یادگیر دارای دو عمل می‌باشد که حضور یا عدم حضور گره متناظر در درخت اشتاینر را مشخص می‌کند. در هر تکرار از الگوریتم اتوماتاهای یادگیر به صورت همزمان فعال شده و یک عمل را انتخاب می‌کنند. سپس بر روی گره‌هایی که اتوماتاهای یادگیر آنها عمل حضور در درخت اشتاینر را انتخاب کرده‌اند یک درخت پوشای کمینه ایجاد می‌شود. با توجه به هزینه درخت بدست آمده از هر تکرار، بردار احتمالات اتوماتاهای یادگیر بروز می‌شود. این روند به دفعات تکرار می‌گردد و در پایان بهترین درخت بدست آمده به عنوان جواب نهایی انتخاب می‌شود. نتایج شبیه‌سازی‌های انجام گرفته کارایی الگوریتم پیشنهادی را هم از لحاظ کیفیت جوابهای تولید شده و هم از لحاظ سرعت همگرایی به جواب در مقایسه با تعدادی از الگوریتم‌های گزارش شده نشان می‌دهد.

ادامه مقاله بدین صورت سازماندهی شده است. ابتدا در بخش ۲ اتوماتاهای یادگیر به صورت اجمالی معرفی می‌گردد. در بخش ۳ الگوریتم پیشنهادی برای حل مسئله درخت اشتاینر ایستا و نتایج شبیه‌سازی‌های انجام گرفته ارائه می‌شود. در بخش ۴ الگوریتم پیشنهادی برای حل به مسئله درخت اشتاینر پویا و نتایج شبیه‌سازی‌ها ارائه می‌شود. بخش پایانی مقاله نتیجه‌گیری می‌باشد.

۲- اتوماتاهای یادگیر

اتوماتای یادگیر یک مدل انتزاعی است که می‌تواند تعداد محدودی عمل را انجام دهد. هر عمل انتخاب شده توسط محیطی تصادفی ارزیابی می‌گردد و پاسخی به اتوماتای یادگیر داده می‌شود. اتوماتای یادگیر از این پاسخ استفاده نموده و عمل خود را برای مرحله بعد انتخاب می‌کند. محیط تصادفی را می‌توان با 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 می‌باشد [۱۱].

هرگاه β_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 α_i نتیجه نامطلوب داشته باشد است. در محیط پایدار مقادیر c_i بدون تغییر باقی می‌مانند، حال آنکه در محیط ناپایدار این مقادیر در طی زمان تغییر می‌کنند. شکل ۲ ارتباط بین اتوماتای یادگیر و محیط را نشان می‌دهد.



مجموعه اعمال $\{\alpha\}$ مجموعه ورودی $\{\beta\}$

شکل ۲- ارتباط بین اتوماتای یادگیر و محیط

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

۲-۱- اتوماتای 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 عمل‌ها، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_m\}$ 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 $c \equiv \{c_1, c_2, ...c_r\}$ احتمال جریمه شدن هر عمل می‌باشد. اگر در اتوماتای یادگیر عمل α_i در مرحله nام 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. در هر حال، تغییرات به گونه‌ای صورت می‌گیرد تا حاصل جمع $p_i(n)$ ها همواره ثابت و مساوی یک باقی بماند.

رابطه (۱) نحوه بروزرسانی بردار احتمالات یک اتوماتای یادگیر هنگامی که این اتوماتا پاسخ مطلوب از محیط دریافت می‌کند را نشان می‌دهد.

$$\begin{aligned} p_i(n+1) &= p_i(n) + a[1 - p_i(n)] \qquad \forall j \qquad j \neq i \\ p_j(n+1) &= (1-a)\, p_j(n) \end{aligned} \tag{۱}$$

در رابطه (۲) بروزرسانی بردار احتمالات اتوماتای یادگیر هنگامی که پاسخ نامطلوب از محیط دریافت می کند، نشان داده شده است.

$$\begin{aligned} p_i(n+1) &= (1-b)\, p_i(n) \\ p_j(n+1) &= \frac{b}{r-1} + (1-b)\, p_j(n) \qquad \forall j \qquad j \neq i \end{aligned} \tag{۲}$$

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 گرفت. زمانیکه a و b با هم برابر باشند، الگوریتم را L_{RP}[۴] می‌نامند. زمانیکه b از a خیلی کوچکتر باشد، الگوریتم را $L_{R\varepsilon P}$[۵] و زمانیکه b مساوی صفر باشد، الگوریتم را L_{RI}[۶] می‌نامند. برای مطالعه بیشتر در رابطه با اتوماتاهای یادگیر با ساختار ثابت و متغیر می‌توان به [۱۲، ۱۳] مراجعه نمود.

۳- حل مسئله اشتاینر ایستا با استفاده از اتوماتاهای یادگیر

در الگوریتم پیشنهادی که آنرا NLA[۷] می‌نامیم، گراف مسئله با یک شبکه از اتوماتاهای یادگیر مدل می‌شود. هر گره غیر ترمینال از گراف به یک اتوماتای یادگیر با دو عمل حضور گره در درخت اشتاینر و عدم حضور گره در درخت اشتاینر مجهز است. در ابتدای الگوریتم، احتمال انتخاب این اعمال مساوی و برابر ۰/۵ می‌باشد.

در هر تکرار از الگوریتم همه اتوماتاهای یادگیر به صورت همزمان فعال شده و هر یک از آنها طبق بردار احتمالات انتخاب اعمال خود یکی از دو عمل خود را انتخاب می‌کند. سپس با استفاده از الگوریتم پریم یک درخت پوشای کمینه بر روی گره‌هایی که انتخاب شده‌اند ایجاد می‌شود. در پایان درخت حاصل هرس می‌شود به این صورت که گره‌های غیرترمینالی که در برگ‌های درخت واقع شده‌اند، حذف می‌گردند. در این روش با احتمال 1-q هر اتوماتای یادگیر گره j را با احتمال p(j) انتخاب و با احتمال q گره j را به صورت حریصانه انتخاب می‌کند. برای تعیین مقدار پارامتر q از یک اتوماتای یادگیر دیگر استفاده می‌شود [۱۵].

در هر تکرار از هزینه درخت بدست آمده برای ارزیابی عمل انتخابی اتوماتاها استفاده می‌شود. اگر هزینه درخت فعلی از هزینه بهترین درخت بدست آمده تا این مرحله کمتر باشد، اعمال انتخاب شده توسط اتوماتاهای یادگیر از طریق افزایش احتمال انتخاب آنها طبق رابطه زیر پاداش داده می‌شود. فرمول یادگیری برای پاداش عمل انتخابی بصورت زیر می‌باشد:

$$p(t+1) = p(t) + \theta.a.(1-p(t)) \qquad (۳)$$
$$\theta = \left| \frac{f(c) - c_{\min}}{f(c)} \right|$$

در رابطه فوق $f(c)$ هزینه درختی است که در تکرار tام تولید شده و C_{min} هزینه بهترین درخت بدست آمده تا این مرحله است. a پارامتر پاداش نامیده می‌شود.

اگر هزینه درخت فعلی از هزینه بهترین درخت اشتاینر بدست آمده تاکنون بیشتر باشد اعمال انتخاب شده توسط اتوماتاهای یادگیر گره‌های شرکت کننده در بهترین درخت حاصل شده تا این مرحله پاداش و اعمال انتخاب شده توسط اتوماتاهای یادگیر دیگر گره‌ها جریمه می‌شوند. برای جریمه عمل انتخابی از فرمول زیر استفاده می‌شود.

$$p(t+1) = p(t) - \theta.b.p_i(n) \qquad (۴)$$

که b پارامتر جریمه نامیده می‌شود. برای بهبود نتایج حاصل شده، در بعضی از تکرارها بر روی درخت‌های بدست آمده عمل کاهش انجام می‌گیرد. در این عمل برای هر یال از درخت به صورت زیر عمل می‌شود:

۱. با حذف یال از درخت، دو زیردرخت مجزا حاصل می‌شود.

۲. با اجرای الگوریتم جستجوی عمق اول نقاط اشتاینری که در برگ‌های دو زیردرخت واقع شده‌اند، حذف می‌شوند.

۳. با استفاده از الگوریتم دایکسترا کوتاهترین مسیر بین گره‌های دو زیردرخت مشخص می‌شود.

۴. اگر مجموع هزینه‌های دو زیردرخت و 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.

از آنجا که عمل کاهش وقت‌گیر است و منجر به افزایش زمان‌اجرای الگوریتم می‌شود، این عمل می‌تواند بر روی تعداد کمی از درخت‌های بدست آمده انجام شود.

۳-۱- نتایج شبیه‌سازی

الگوریتم پیشنهادی بر روی گرافهای مجموعه B از OR-Library آزمایش شده است. این مجموعه شامل ۱۸ گراف است که برای هر گراف تعداد گره ها، تعداد یالها و هزینه درخت اشتاینر بهینه در گراف مشخص است [۱۴].

در آزمایش اول برای پیدا کردن کاراترین الگوریتم یادگیری، الگوریتم پیشنهادی با الگوریتم‌های یادگیری L_{RP} و $L_{R\varepsilon P}$ پیاده‌سازی شده است. در L_{RP} مقدار پارامترهای a و b برابر ۰/۳ و در $L_{R\varepsilon P}$ مقدار پارامتر a برابر ۰/۳ و مقدار پارامتر b برابر ۰/۰۱ در نظر گرفته شده است. نتایج میانگین ۲۰ بار اجرای این الگوریتم‌ها با ۵۰۰ تکرار در هر اجرا در جدول ۱ ارائه شده است. ستون اول جدول شماره گراف در مجموعه B مسائل بیسلی، ستون دوم هزینه درخت اشتاینر بهینه برای هر گراف که در [۱۴] گزارش شده است و ستونهای بعدی درصد خطای نسبی حاصل از اجرای الگوریتم پیشنهادی با الگوریتم‌های یادگیر مختلف را نشان می‌دهد. برای محاسبه خطای نسبی از رابطه زیر استفاده می‌شود:

$$relative-error = \frac{C - C_{opt}}{C_{opt}} \qquad (۵)$$

که C هزینه درخت اشتاینر حاصل از الگوریتم و C_{opt} هزینه درخت اشتاینر بهینه در گراف است. عمل کاهش تنها در ۳۰ تکرار انتهایی انجام گرفته است. همان‌طور که از جدول ۱ مشخص است استفاده از اتوماتای یادگیر $L_{R\varepsilon P}$ منجر به نتایج بهتری نسبت به L_{RP} می‌شود. الگوریتم یادگیری $L_{R\varepsilon P}$ برای ۱۰ گراف اول مجموعه B بیسلی به جواب بهینه دست می یابد. به همین دلیل در آزمایشات بعدی از این روش یادگیری استفاده شده است.

جدول ۱- نتایج الگوریتم NLA با الگوریتم‌های یادگیری L_{RP} و $L_{R\varepsilon P}$

شماره گراف	هزینه بهینه	L_{RP}	$L_{R\varepsilon P}$
۱	۸۲	۰	۰
۲	۸۳	۰	۰
۳	۱۳۸	۰	۰
۴	۵۹	۲/۵۴	۰
۵	۶۱	۱/۶۴	۰
۶	۱۲۲	۰	۰
۷	۱۱۱	۵/۴۹	۰
۸	۱۰۴	۲/۸۸	۰
۹	۲۲۰	۰	۰
۱۰	۸۶	۲/۰۹	۰

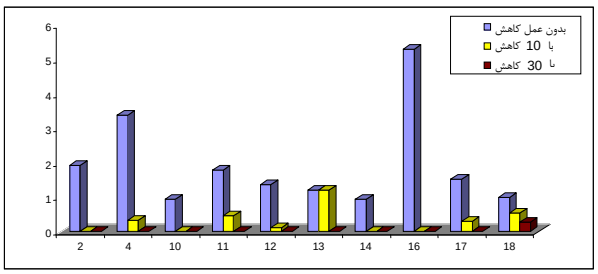
در جدول ۲ کیفیت جواب‌های بدست آمده در روش پیشنهادی با تعدادی از الگوریتم‌های حل مسئله اشتاینر مقایسه شده است. ستون اول جدول شماره گراف در مجموعه B بیسلی، ستون دوم هزینه درخت اشتاینر بهینه و سایر ستون‌ها درصد خطای نسبی تعدادی از الگوریتم‌های گزارش شده برای حل مسئله درخت اشتاینر شامل SDG، SPH، ADH و ACS را نشان می‌دهد. الگوریتم SPH برای ساخت درخت اشتاینر یک گره ترمینال را به تصادف انتخاب می‌کند. سپس تا قرار

گرفتن کلیه گره‌های ترمینال در درخت، در هر مرحله یال با کمترین هزینه را به درخت اضافه می‌کند [۴]. الگوریتم ADH از گره‌های ترمینال شروع می‌کند. هر گره ترمینال نزدیکترین گره به خود را انتخاب می‌کند. با این کار مجموعه‌ای از جنگلها ایجاد می‌گردد که در پایان الگوریتم از ادغام آنها یک درخت ایجاد می‌شود [۵]. الگوریتم ACS ابتدا بر روی هر گره ترمینال یک مورچه قرار می‌دهد. در هر تکرار از الگوریتم کلیه مورچه‌ها با توجه به غلظت فرمون یال‌های مجاور یک یال را انتخاب و به سمت آن حرکت می‌کنند. در صورت برخورد یک مورچه به مسیر مورچه دیگر درخت‌های آنها ادغام می‌شود. این مراحل آنقدر تکرار می شود تا نهایتا یک درخت باقی بماند [۱۰]. همان طور که در جدول ۲ نشان داده شده است الگوریتم پیشنهادی برای ۱۴ گراف از ۱۸ گراف مجموعه B بیسلی به جواب بهینه دست یافته است. در مورد ۴ گراف دیگر نیز در صد خطای نسبی الگوریتم NLA کمتر از درصد خطای نسبی الگوریتم‌های SDG، SPH و ADH است.

جدول ۲- مقایسه درصد خطای نسبی الگوریتم‌های حل مسئله اشتاینر

شماره گراف	هزینه بهینه	SDG %	SPH %	ADH %	ACS %	NLA %
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	۱/۸
۱۲	۱۷۴	۰	۰	۰	۰/۱۱	۰
۱۳	۱۶۵	۶/۰۶	۷/۸۸	۲/۲۴	۰	۴/۲
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	۱/۲
۱۷	۱۳۱	۳۴/۵	۳/۸۲	۳/۰۵	۱/۲۲	۱/۳
۱۸	۲۱۸	۴/۵۹	۱/۸۳	۰	۱/۴۲	۰

برای بررسی تأثیر عمل کاهش، الگوریتم پیشنهادی یکبار بدون عمل کاهش و بار دیگر با اجرای عمل کاهش بر روی تکرارهای نهایی الگوریتم (به تعداد ۱۰ و ۳۰) اجرا شده است. شکل ۳ درصد خطای نسبی الگوریتم NLA با تعداد عمل‌های کاهش متفاوت را نشان می‌دهد. همان طور که مشاهده می‌شود عمل کاهش باعث بهبود نتایج حاصل از الگوریتم می‌شود.



شکل ۳- درصد خطای نسبی الگوریتم NLA با عمل‌های کاهش متفاوت

در آزمایش پایانی برای بررسی کارایی الگوریتمهای پیشنهادی در شبکه‌های واقعی از شبیه‌ساز شبکه $ns2^{8}$ استفاده شده است. با استفاده از اصلاح شده روش واکسمن [۱۵] تعدادی توپولوژی شبکه تولید شده است. توپولوژی شبکه با توزیع تصادفی n گره در ناحیه مستطیل شکل با طول و عرض ۳۰۰ متر ایجاد شده است. برای تولید این گراف‌ها مقدار پارامتر α در روش واکسمن برابر ۰/۵ و مقدار پارامتر β برابر ۰/۱ در نظر گرفته شده است. مشخصات این گراف‌ها در جدول ۳ نشان داده شده است.

جدول ۳- مشخصات شبکه‌های ایجاد شده

شماره گراف	تعداد گره‌ها	تعداد یال‌ها	تعداد ترمینال‌ها
۱	۱۰	۴۳	۳
۲	۲۰	۵۶	۶
۳	۳۰	۱۰۷	۸
۴	۴۰	۱۹۳	۱۱
۵	۵۰	۲۶۷	۱۳
۶	۶۰	۴۶۴	۱۶
۷	۷۰	۵۸۹	۱۸
۸	۸۰	۱۶۸	۲۵
۹	۹۰	۲۰۵	۲۵

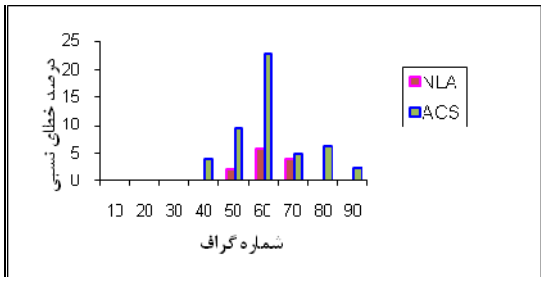
در شکل ۴ نتیجه اجرای الگوریتم پیشنهادی بر روی شبکه‌های جدول ۳ با الگوریتم‌های تکرار شونده کلونی مورچه‌ها [۱۰] و الگوریتم ژنتیکی [۹] مقایسه شده است. این الگوریتم‌ها بر روی یک کامپیوتر محلی با مشخصات اینتل پنتیوم۴، ۳.۰۶ گیگا هرتز ، ۴۸۰ مگابایت حافظه و ویندوز XP سرویس پک۲ اجرا شده‌اند. این نتایج حاصل میانگین‌گیری از ۲۰ بار اجرای الگوریتم‌هاست. تعداد تکرار الگوریتم NLA و ACS در هر اجرا برابر ۵۰۰ در نظر گرفته شده است.

در شکل ۴- الف درصد خطای نسبی الگوریتم‌های NLA و ACS با یکدیگر مقایسه شده است. از آنجا که هزینه درخت اشتاینر بهینه در گراف‌های تولید شده در دسترس نیست برای مقایسه الگوریتم‌ها با یکدیگر، در معیار خطای نسبی بجای هزینه بهینه از هزینه حاصل از اجرای الگوریتم ژنتیک با تعداد تکرارهای زیاد استفاده شده است.

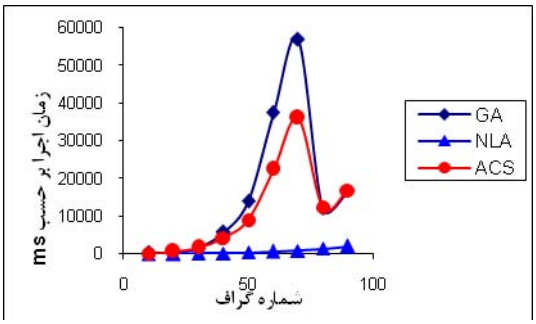
در شکل ۴- ب زمان اجرای الگوریتم‌های NLA، GA و ACS نشان شده است. همان‌طور که در شکل ۴ مشاهده می‌شود، الگوریتم GA در تعداد تکرارهای زیاد جوابهای بهتری نسبت به دیگر الگوریتم‌ها تولید می‌کند با این وجود زمان اجرای این الگوریتم در مقایسه با دیگر الگوریتم‌ها به مراتب بیشتر است. الگوریتم پیشنهادی در مقایسه با الگوریتم کلونی مورچه‌ها به نتایج بهتری رسیده است. به علاوه این الگوریتم زمان محاسباتی کمتری نیاز دارد.

۴- حل مسئله درخت اشتاینر پویا

مسئله درخت اشتاینر پویا به صورت زیر تعریف می شود: گراف همبند $G(V,E)$ که V مجموعه متناهی از گره‌های گراف و E مجموعه یال‌ها (اتصالات بین گره‌ها) است و یک دنباله از درخواست‌های تغییر در گراف را در نظر بگیرید. هدف از مسئله اشتاینر پویا یافتن درختی با حداقل هزینه است که مجموعه ترمینال‌ها را در برگیرد و به هر درخواست رسیده نیز بدون اطلاع از درخواست‌های بعدی پاسخ دهد. درخواست‌های در نظر گرفته شده عبارتند از افزودن یا حذف یک گره ترمینال، افزودن یا حذف یک گره غیرترمینال، افزودن یا حذف یک یال و تغییر وزن یک یال.



الف) مقایسه درصد خطای نسبی الگوریتم‌های NLA و ACS



ب) مقایسه زمان اجرای الگوریتم‌های NLA، ACS و GA

شکل ۴- مقایسه الگوریتم‌های NLA، ACS و GA

در بخش قبل با استفاده از شبکه‌ای از اتوماتاهای یادگیر همکار روشی برای حل مسئله اشتاینر ایستا پیشنهاد شد. در الگوریتم حل مسئله اشتاینر پویا در ابتدا الگوریتم ایستا بر روی گراف قبل از تغییرات اجرا و بردارهای احتمال‌های اقدام‌های اتوماتاهای یادگیر محاسبه می‌شود. بعد از تغییرات (حذف و یا اضافه کردن گره به گراف)، الگوریتم درخت اشتاینر ایستا با در نظر گرفتن آخرین مقادیر بردارهای احتمال‌های اقدام‌های اتوماتاهای یادگیر اجرا می‌شود. از این طریق با تعداد تکرارهای کمتری درخت اشتاینر مورد نظر تعیین می‌شود.

۴-۱- نتایج شبیه سازی

جدول ۴ نتیجه الگوریتم NLA را بر روی گراف‌های ۱، ۳، ۹، ۱۵ و ۱۸ از مجموعه B از دسته مسائل بیسلی با تعداد تکرارهای ۱۰ و ۲۰ برای هر درخواست را نشان می‌دهد. در این آزمایش‌ها، ابتدا الگوریتم ایستا بر روی گراف‌های اصلی با تعداد تکرار ۵۰۰ اجرا شده است. سپس هر یک از این گرافها به میزان ۲٪، ۵٪ و ۱۰٪ تعداد گره‌های گراف تغییر داده شده و برای هر درخواست الگوریتم حل مسئله اشتاینر ایستا اجرا شده است. برای محاسبه درصد خطای نسبی، الگوریتم ایستا با تعداد تکرارهای زیاد بر روی گراف تغییر یافته اعمال و کم‌هزینه‌ترین درخت بدست آمده به عنوان جواب در نظر گرفته شده است.

جدول ۴- الف نتیجه اجرای الگوریتم NLA با تعداد تکرار ۲۰ برای هر درخواست را نشان می‌دهد. در این جدول متوسط درصد افت کیفیت (بدتر شدن هزینه کل درخت اشتاینر) ۰/۰۹ درصد ثبت شده است. جدول ۴- ب نتیجه اجرای الگوریتم را با تعداد تکرار ۱۰ برای هر درخواست نشان می‌دهد. در این مرحله کیفیت درخت اشتاینر به میزان کمی بدتر شده است. متوسط خطای نسبی بدست آمده ۰/۱۳ درصد است. در این مرحله نیز متوسط خطا بیشتر شده است ولی نتایج همچنان قابل مقایسه می‌باشد و اختلاف قابل توجهی دیده نمی‌شود. بنابراین جدول ۴ نشان می‌دهد که پس از هر تغییر یا تغییرات، می‌توان با استفاده از الگوریتم حل مسئله درخت اشتاینر ایستا به شرطی که مقادیر بردار احتمالات اقدام‌ها قبل از تغییرات در نظر گرفته شود، با تعداد تکرارهای بسیار کمتر از تعداد تکرارهای حالتی که مقادیر بردار احتمالات اقدام‌ها به طور تصادفی انتخاب شده باشند به جواب مطلوب و نزدیک به بهینه رسید. ماهیت تصادفی الگوریتم باعث می‌شود نسبت به نوع تغییر حساسیت زیادی نداشته باشد.

جدول ۴- درصد خطای نسبی وزن درخت اشتاینر پویا با تعداد تکرارهای مختلف

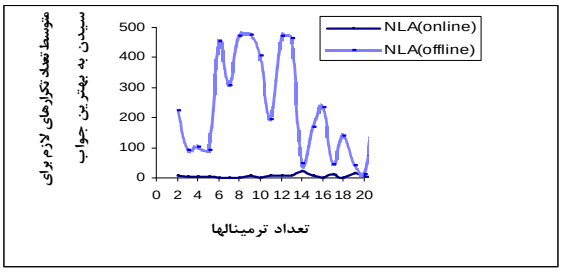
الف) درصد خطای نسبی وزن درخت اشتاینر پویا با تعداد تکرار ۲۰

شماره گراف	تعداد گره	تغییر ۲٪	تغییر ۵٪	تغییر ۱۰٪
۱	۵۰	۰	۰	۰
۳	۵۰	۰	۰	۰
۹	۷۵	۰	۰	۰
۱۵	۱۰۰	۰	۰	۰
۱۸	۱۰۰	۰/۴۴	۰	۰/۸۸

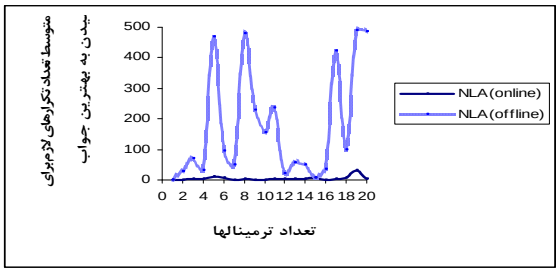
ب) درصد خطای نسبی وزن درخت اشتاینر پویا با تعداد تکرار ۱۰

شماره گراف	تعداد گره	٪۲	٪۵	٪۱۰
۱	۵۰	۰	۰	۰
۳	۵۰	۰	۰	۰
۹	۷۵	۰	۰	۰
۱۵	۱۰۰	۰	۰	۰/۰۶
۱۸	۱۰۰	۰/۸۸	۰	۰/۹۷

برای مقایسه زمان اجرای الگوریتم ایستا و پویا نمودار متوسط تعداد تکرارهای لازم برای رسیدن به بهترین جواب برحسب تعداد ترمینال‌ها، برای گراف‌های شماره ۲ و ۱۱ از مجموعه B مسائل بیسلی در شکل ۵ نشان داده شده است. منظور از بهترین جواب، کمترین هزینه بدست آمده از اجرای الگوریتم ایستا در تکرارهای بالاست. همان طور که در شکل دیده می‌شود در اکثر موارد الگوریتم NLA پویا نیاز به تعداد تکرارهای بسیار کمتری دارد.



الف- گراف شماره ۲



ب- گراف شماره ۱۱

شکل ۵- متوسط تعداد تکرارهای لازم در الگوریتم NLA ایستا و پویا برای رسیدن به جواب بر حسب تعداد ترمینال‌ها

۵- نتیجه‌گیری

در این مقاله ابتدا یک الگوریتم مبتنی بر اتوماتاهای یادگیر برای حل مسئله اشتاینر ایستا پیشنهاد شد و سپس این الگوریتم برای حل مسئله اشتاینر پویا استفاده گردید. نتایج بدست آمده از آزمایش‌ها نشان داد که الگوریتم پیشنهاد شده در این مقاله برای حل هر دو مسئله درخت اشتاینر ایستا و درخت اشتاینر پویا از کارایی بالایی برخوردار است. نتایج شبیه‌سازی‌های انجام گرفته کارایی الگوریتم پیشنهادی را هم از لحاظ کیفیت جواب‌های تولید شده و هم از لحاظ سرعت همگرایی به جواب در مقایسه با تعدادی از الگوریتم‌های گزارش شده نشان می‌دهد. به علاوه از آنجا که زمان اجرای این روش متناسب با تعداد گره‌های گراف است برای گراف‌های شلوغ (گراف با تعداد یال‌های زیاد) سرعت همگرایی بسیار مناسبی دارد.

مراجع

[1] S. E. Dreyfuss, and R. A. Wagner, "The Steiner Problem in Graphs," *Networks*, Vol. 1, pp. 195-207, 1971.

[2] S. Chopra, E. R. Gorres, and M. R. Rao, "Solving the Steiner tree problem on a graph using branch and cut," *ORSA Journal on Computing*, Vol. 4, pp. 320-335, 1992.

[3] S. Milan, and V. Mirko, "An Exact Algorithm for Steiner Tree Problem on Graphs," *International Journal of Computers, Communications & Control*, Vol. I, No. 1, pp. 41-46, 2006.

[4] H. Takahashi, and A. Matsuyama, "An Approximate Solution for the Steiner Problem in Graphs," *Mathematica Japonica*, Vol. 24, No. 6, pp. 573-577, 1980.

[5] V. J. Rayward-Smith, and A. Clare, "On Finding Steiner Vertices," *Networks*, Vol. 16, pp. 283-294, 1986.

[6] L. Kou, G. Markowsky, and L. Berman, "A Fast Algorithm for Steiner Trees," *Acta Informatica*, Vol. 15, pp. 141-145, 1981.

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

[8] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, Vol. 6, No. 9, pp. 1611–1622, 1998.

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

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

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

[12] M. A. L. Thathachar, and P. S. Sastry, "Varieties of Learning Automata: An Overview," *IEEE Trans. Systems, Man and Cybernetics – Part B: Cybernetics*, Vol. 32, No. 6, 2002.

[۱۳] م. ر. شیرازی، و م. ر. میبدی، "به کارگیری اتوماتای یادگیر در سیستمهای جندعامله همکار،" *مجموعه مقالات اولین کنفرانس بین المللی فناوری اطلاعات و دانش*، ص. ۳۳۸-۳۴۹، تهران، ۱۳۸۲.

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

[15] B. M. Waxman, "Routing of multipoint connections," *IEEE J. Select. Areas Commun.* Vol. 6, No. 9, pp. 1617-1622, 1988.

سمیرا نوفرستی در حال حاضر عضو هیات علمی گروه فناوری اطلاعات، دانشکده مهندسی برق و کامپیوتر دانشگاه سیستان و بلوچستان است. وی مدرک کارشناسی خود را در رشته کامپیوتر گرایش نرم‌افزار از دانشگاه صنعتی شریف در سال ۱۳۸۲ و مدرک کارشناسی‌ارشد را در همین رشته از دانشگاه صنعتی امیرکبیر در سال ۱۳۸۴ اخذ کرده است. پس از فارغ‌التحصیلی به مدت یک سال عضو هیات علمی دانشگاه آزاد اسلامی واحد بیرجند بوده و از آذر ماه ۱۳۸۵ به عنوان هیات علمی دانشگاه سیستان و بلوچستان مشغول به کار شده است. هم‌اکنون نیز سمت معاونت کتابخانه مرکزی و مرکز اسناد دانشگاه را دارا است. زمینه‌های تحقیقاتی وی عبارتند از سیستم‌های یادگیر، مهندسی نرم‌افزار و هوش مصنوعی.

آدرس پست‌الکترونیکی ایشان عبارت است از:

snoferesti@ece.usb.ac.ir

محمدرضا میبدی در سال ۱۳۳۱ در اراک بدنیا آمد. وی تحصیلات دانشگاهی خود را در دانشگاه شهید بهشتی در رشته اقتصاد آغاز و تا سطح کارشناسی‌ارشد در آنجا به تحصیل پرداخت و در سال ۱۳۵۶ موفق به اخذ مدرک کارشناسی‌ارشد شد. وی در سال ۱۳۶۲ موفق به اخذ دکتری رشته علوم کامپیوتر از دانشگاه اوکلاهما ی آمریکا شد و هم‌اکنون عضو هیئت علمی و استاد دانشگاه صنعتی امیرکبیر می‌باشد. مشاغل و سمت‌های اداری و مدیریتی دکتر محمدرضا میبدی به ترتیب زیر است: رئیس و عضو کمیته مهندسی کامپیوتر در وزارت علوم تحقیقات و فناوری از سال ۱۳۷۵ تا کنون، عضو کمیته داوری برای چهاردهمین جشنواره خوارزمی در وزارت علوم تحقیقات و فناوری از سال ۱۳۷۵ تا کنون، عضو کمیته علمی در مرکز تحقیقات ریاضیات و فیزیک نظری وزارت علوم تحقیقات و فناوری از سال ۱۳۷۷ تا سال ۱۳۷۹، عضو کمیته چند رشته‌ای وزارت علوم تحقیقات و فناوری از سال ۱۳۷۵ تا کنون، عضو کمیته پژوهشی وزارت علوم تحقیقات و فناوری از سال ۱۳۸۲ تا سال ۱۳۸۳، عضو کمیته علمی در مرکز تحقیقات علوم پایه وزارت علوم تحقیقات و فناوری از سال ۱۳۸۱ تا کنون، عضو کمیته انتخاب بهترین کتاب ایرانی وزارت فرهنگ و ارشاد اسلامی از سال ۱۳۷۵ تا کنون.

فعالیت‌های آموزشی و عنوان دروس ارائه شده توسط محمدرضا میبدی به قرار زیر است: نظریه زبان‌ها و ماشین‌ها، تحلیل و طراحی الگوریتم‌های موازی، ساختمان داده، تحلیل و طراحی الگوریتم، محاسبات موازی.

آدرس پست‌الکترونیکی ایشان عبارت است از:

mmeybodi@aut.ac.ir

اطلاعات بررسی مقاله:

تاریخ ارسال: ۸۶/۰۷/۲۴
تاریخ اصلاح: ۸۹/۰۳/۰۹
تاریخ قبول شدن: ۸۹/۰۳/۲۳
نویسنده مرتبط: سمیرا نوفرستی، دانشکده مهندسی برق و کامپیوتر، دانشگاه سیستان و بلوچستان، زاهدان، ایران.

[1] Steiner Tree
[2] Minimum Spanning Tree
[3] Prim
[4] Linear Reward Penalty
[5] Linear Reward Epsilon Penalty
[6] Linear Reward Inaction
[7] Network of Learning Automata
[8] Network Simulator