حل مسئله درخت اشتاینر کمینه با استفاده از اتوماتاي يادگير توزيع شده

علی نوراله[۱] محمد رضا میبدی[۲]

چکیده

مسئله پیدا کردن درخت اشتاینر کمینه در یک گراف وزندار عبارت است از پیدا کردن یک درخت با کمترین هزینه بر روی گراف که شامل تعدادی از گره های خاص به نام ترمینال باشد. این مسئله از جمله مسائل NP-Complete می‌باشد و بهمین دلیل الگوریتمهای تقریبی متعددی مانند الگوریتمهای ژنتیک و کلونی مورچه‌ها برای آن گزارش شده است. در این مقاله الگوریتمی مبتنی بر آتاماتای یادگیر توزیع شده برای حل مسئله درخت اشتاینر کمینه پیشنهاد می‌گردد. نتایج حاصل از آزمایشها نشان میدهد که الگوریتم پیشنهادی در مقایسه با روشهای گزارش شده مانند الگوریتمهای ژنتیکی و کلونی مورچه ها از کارایی بالاتری بر خوردار است.

کلمات کليدي: مسئله درخت اشتاینر کمینه، اتوماتاي يادگير ، اتوماتاي يادگير توزيع شده

Solving Minimum Steiner Tree Problem Using Generalized Distributed Learning Automata

Ali Nourollah Mohammad Reza Meybodi

Abstract

Minimum Steiner tree problem in a weighted graph is to find a least cost sub tree in the graph such that it contains special vertices called terminal points. Since this problem belongs to NP-Complete problems, many approximate algorithms including genetic algorithms and ant colony strategies have been designed for solving it. In this paper, we propose an algorithm based on distributed learning automata for solving Minimum Steiner tree problem. Experimental results show that the proposed algorithm is superior to other existing algorithms.

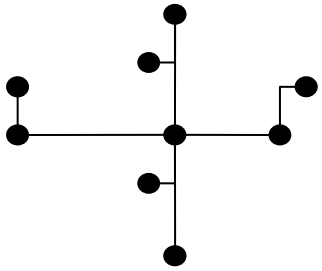
Keywords: Minimum Steiner Tree Problem, Learning Automata, Distributed Learning Automata

۱- مقدمه

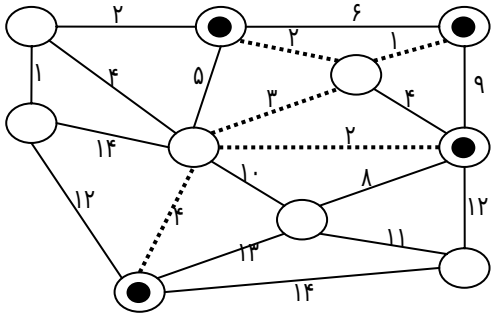
مسئله پیدا کردن درخت اشتاینر کمینه در یک گراف وزندار که از جمله مسائل NP-Complete میباشد عبارت است از پیدا کردن یک درخت با کمترین هزینه که شامل تعدادی از گره های خاص به نام ترمینال باشد. این مساله انواع مختلفی دارد. در یک نوع آن تعدادی نقطه بر روی صفحه دو بعدی به عنوان ورودی ارائه میگردد و هدف، تولید درختی بر روی صفحه اقلیدسی است که دارای کمترین هزینه بوده و همه نقاط را در بر بگیرد[۱]. همین مسئله به صورت عمود بر هم[۱] نیز مطرح میشود که در آن باید خطوط درخت فقط به صورت عمودی یا افقی رسم شوند که این مسئله در طراحی فیزیکی مدارات VLSI کاربرد فراوانی دارد. شکل (۱) نمونه‌ای از یک درخت اشتاینر کمینه عمودی- افقی را نشان می-دهد. نوع دیگر مسئله درخت اشتاینر در گراف است که در این مقاله نیز ما همین مسئله را مورد بررسی قرار داده‌ایم. در این مسئله یک گراف وزن‌دار $G=(V,E)$ و یک زیرمجموعه از رئوس گراف $(T\subseteq V)$ که مجموعه ترمینالها نام دارد ارائه می‌گردد و هدف پیدا کردن درخت $GT=(V',E')$ با کمترین هزینه است که $T\subseteq V'\subseteq V$ و $E'\subseteq E$. مسئله درخت اشتاینر دارای کاربردهای فراوانی نظیر مسیریابی یک به چند در شبکه‌های

[1]آزمایشگاه تحقیق و توسعه نرم افزار، دانشکده مهندسي کامپيوتر و فن آوري اطلاعات، دانشگاه صنعتي امير کبير، nourollah@ce.aut.ac.ir
[2]آزمایشگاه محاسبات نرم، دانشکده مهندسي کامپيوتر و فن آوري اطلاعات، دانشگاه صنعتي امير کبير، mmeybodi@ aut.ac.ir

کامپیوتری[۲]، سیستمهای حمل و نقل، پلیس امداد، ایستگاههای پستی و کاربردهای دیگر میباشد[۲،۳]. شکل (۲) نمونه‌ای از یک درخت اشتاینر کمینه بر روی گراف را نشان میدهد. خطوط خطچین درخت مورد نظر میباشند و رئوس توپر ترمینال هستند.



شکل (۱): درخت اشتاینر کمینه عمودي - افقي



شکل (۲): درخت اشتاینر کمینه بر روي گراف

با توجه به اینکه مساله پیدا کردن درخت اشتاینر کمینه یک مساله NP-Complete میباشد الگوریتمهای تقریبی متعددی از جمله الگوریتمهای ژنتیکی[۴] و کلونی مورچه ها[۵،۶] برای آن گزارش شده است.

در روش ADH[۳] از گره‌های ترمینال شروع میکنیم. هر گره ترمینال یال با کمترین هزینه مجاور خود را انتخاب میکند و با این کار جنگلی از درختان تشکیل میشود که این جنگل در انتهای الگوریتم شامل یک درخت خواهد بود[۳]. در روش SPH[4] از یک گره اولیه شروع کرده و هر بار یال با کمترین هزینه را به درخت موجود اضافه میکنیم. این عملیات برای همه رئوس گراف تکرار میشود و از بین درختان تولید شده کم هزینه ترین درخت انتخاب میشود[۷]. در روش ACS[5] که مبتنی بر سیستم کولونی مورچه‌ها است روی هر گره ترمینال یک مورچه قرار داده و هر مورچه به صورت احتمالی یالهای اطراف خود را انتخاب کرده و حرکت میکند. در صورت برخورد یک مورچه با مسیر مورچه دیگر آن دو مسیر با هم پیوند خورده و کم کم جنگل حاصل از درختانی که مسیر مورچه‌ها می‌باشد پیوند خورده و یک درخت باقی می‌ماند[۵،۶].

در این مقاله الگوریتمی مبتنی بر اتوماتای یادگیر توزیع شده برای حل مساله درخت اشتاینر کمینه پیشنهاد میکردد. نشان داده میشود که الگوریتم پیشنهادی در مقایسه با روشهای گزارش شده مانندریتمهای ژنتیکی، کلونی مورچه ها، روش ADH و روش SPH از کارایی بالاتری بر خوردار است. آزمایشهای بر روی یک مجموعه مسائل استاندارد که در [۸] ارائه شده است انجام گرفته است..

ادامه مقاله بدین صورت سازماندهی شده است. در بخش ۲ ابتدا اتوماتاهای یادگیر و اتوماتای یادگیر توزیع شده به طور مختصر شرح داده میشود و سپس اتوماتای یادگیر توزیع شده تعمیم یافته ارایه میگردد. در بخش ۳ الگوریتم پیشنهادی شرح داده میشود و در بخش ۴ نتایج آزمایشها ارایه میگردد. بخش نهایی مقاله نتیجه گیری میباشد.

۲- اتوماتای یادگیر توزیع شده

در این بخش ابتدا اتوماتاهای یادگیر و اتوماتای یادگیر توزیع شده و سپس اتوماتای یادگیر توزیع شده تعمیم یافته به طور مختصر شرح داده میشود

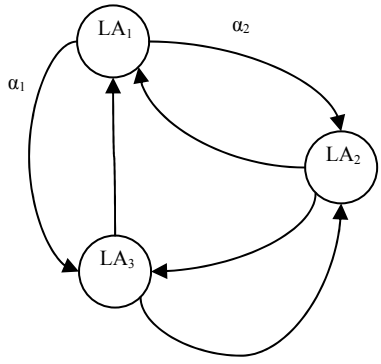
اتوماتای یادگیر: یک اتوماتای یادگیر یک مدل انتزاعی است که میتواند تعداد محدودی عمل یا اقدام[۶] را انجام دهد. هدف آتاماتای یادگیر آن است که بتواند از بین یک سری اعمال مجاز، عمل بهینه را انتخاب کند. عمل بهینه عملی است که احتمال دریافت پاداش از محیط را به حداکثر برساند. آتاماتا با یک محیط در تعامل است و هر عمل که توسط اتوماتای یادگیر انتخاب میشود به محیط اعمال شده و نتیجه آن عمل در محیط ارزیابی گردیده و به اتوماتای یادگیر ارائه میگردد. اتوماتای یادگیر از این پاسخ استفاده نموده و عمل بعدی خود را بهبود می بخشد. شکل (۳) این تعامل بین محیط و اتوماتای یادگیر را نشان میدهد.

شکل (۳): ارتباط آتاماتاي يادگير و محيط

محيط: محيط را ميتوان توسط سه تايي $E=(\alpha, \beta, c)$ نشان داد که در آن $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه ورودیها و $\beta = \{\beta_1, \beta_2, ..., \beta_m\}$ مجموعه خروجيها و $c = \{c_1, c_2, ..., c_r\}$ مجموعه احتمالهای جريمه ميباشد. هرگاه β مجموعه دو عضوی باشد، محيط از نوع P ميباشد. در چنين محيطی $\beta_1=1$ به عنوان جريمه و $\beta_2=0$ به عنوان پاداش محسوب ميشود. در محيط از نوع Q، $\beta(n)$ ميتواند به طور گسسته يک مقدار از مقادير محدود در فاصله [0,1] و در محيط از نوع S ، $\beta(n)$ متغير تصادفی در فاصله [0,1] است. c_i احتمال آن است که عمل α_i نتيجه نامطلوب داشته باشد. در محيط ايستا مقادير c_i بدون تغيير ميمانند در حاليکه در محيط غير ايستا اين مقادير در طی زمان تغيير ميکنند. آتاماتاهای يادگير به دو گروه ساختار ثابت و ساختار متغير تقسيم بندی ميگردند که به دليل استفاده ما از آتاماتای با ساختار متغير در ادامه آن را به طور مختصر شرح ميدهيم.

اتوماتاهای يادگير با ساختار متغير: اين اتوماتای يادگير توسط چهارتايی (α, β, p, T) نشان داده ميشود که در آن $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه عملهای اتوماتا، $\beta = \{\beta_1, \beta_2, ..., \beta_m\}$ مجموعه ورودیهای اتوماتا، $p = \{p_1, p_2, ..., p_r\}$ بردار احتمال انتخاب هر يک از عملها و $p(n+1)=T[\alpha(n), \beta(n), p(n)]$ الگوريتم يادگيری ميباشد. در اين نوع از اتوماتاهای يادگير اگر عمل α_i در مرحله n-ام انتخاب شود و پاسخ مطلوب از محيط دريافت نمايد، احتمال $p_i(n)$ افزايش يافته و ساير احتمالها کاهش مييابند و برای پاسخ نامطلوب احتمال $p_i(n)$ کاهش يافته و ساير احتمالها افزايش مييابند. در هر حال تغييرات به گونهای صورت ميگيرد که حاصل جمع $p_i(n)$ ها همواره برابر يک باقی بماند.

اتوماتای يادگير توزيع شده(DLA): اين آتاماتا شبکهای از اتوماتاهای يادگير است که برای حل يک مسئله با يکديگر همکاری ميکنند. تعداد اقدامهای يک اتوماتای يادگير در DLA برابر با تعداد آتاماتاهای متصل به آتاماتای مورد نظر ميباشد. انتخاب يک اقدام توسط آتاماتا در شبکه، اتوماتای متناظر با اين اقدام را فعال ميسازد. به عنوان مثال در شکل (۴) هر اتوماتا دارای دو اقدام ميباشد. انتخاب اقدام α_1 توسط LA_1 ، اتوماتای يادگير LA_3 را فعال خواهد کرد. LA_3 به نوبه خود يکی از اقدامهای خود را انتخاب ميکند که در نتيجه آن يکی از اتوماتاهای يادگير متصل به آن اتوماتا که متناظر با اقدام انتخاب شده ميباشد فعال ميشود. در هر زمان فقط يک اتوماتای يادگير در شبکه فعال ميباشد. بطور رسمی DLA را ميتوان توسط گراف $DLA=(V,E)$ که $V=\{LA_1, LA_2, ..., LA_n\}$ مجموعه اتوماتاهای يادگير و n تعداد آتاماتاها در DLA و $E \subseteq V \times V$ مجموعه يالهای گراف ميباشد، تعريف کرد. يال (i,j) اقدام j آتاماتا LA_i را نشان ميدهد. LA_j زمانی فعال خواهد شد که اقدام j آتاماتون LA_i انتخاب شود. تعداد اقدامهای آتوماتای LA_k $(k=1,\ldots, n)$ برابر درجه خروجی آن گره ميباشد[۹].



شکل (۴): آتاماتاي يادگير توزيع شده

اتوماتای یادگیر توزیع شده تعمیم یافته (GDLA[7]): یک اتوماتای یادگیر توزیع شده تعمیم یافته متناظر با یک گراف $G=(V,E)$ است که در آن $V=\{v_1,v_2,...,v_n\}$ مجموعه گره های گراف و E مجموعه یالهای گراف و n تعداد گره های گراف میباشد. هر زیر مجموعه از V میتواند یک آتاماتای یادگیر محسوب شود که در هر زمان تعدادی از این اتوماتونهای یادگیر فعال و تعدادی غیر فعال هستند. هر آتاماتون را با T و مجموعه آتاماتونهای فعال را با مجموعه S نمایش میدهیم. اگر تعداد آتاماتونهای فعال T_i ، m باشد در این صورت $S=\{T_1,T_2,...,T_m\}$ که برای هر $T_i \in S$ خواهیم داشت $T_i=\{v_{j_1},v_{j_2},...,v_{j_{k_i}}\}; for\ i=1,2,...,m$.

هر T_i دارای k_i عضو است به طوریکه

$$for\ all\ T_i\ ,\ T_j \in S\ (i\neq j)\ ,\ \ T_i \cap T_j = \varnothing$$

برای هر اتوماتای یادگیر Ti به تعداد یالهای خروجی از مجموعه T_i اقدام وجود دارد. مجموعه اقدامهای هر آتاماتای یادگیر T_i بصورت $A_i=\{(u,v)|u\in T_i, v\notin T_i, (u,v)\in E\}$ تعریف میشود.

در طول زمان هر یک از اتوماتونهای فعال Ti (عناصر مجموعه S) یکی از اقدامهای خود را انتخاب و سپس اعمال مینماید که در اثر آن ممکن است یکی از دو حالت زیر رخ دهد.

- اعمال اقدام منجر به انتخاب راسی مانند v از گراف شود که آن راس داخل هیچ کدام از T_j های فعال قرار نداشته باشد$(j\neq i)$ و یا به بیان دیگر $\forall T_j \in S, v\notin T_j$ که در این حالت اتوماتای یادگیر T_i غیر فعال شده و آتاماتایی که معادل $T_i \cup \{v\}$ است فعال میشود که در نتیجه تعداد عناصر مجموعه S تغییر نمیکند.
- اعمال اقدام منجر به انتخاب راسی مانند v از گراف شود که آن راس در داخل عناصر یک آتاماتای فعال دیگر مانند T_j قرار دارد و یا به بیان دیگر $\exists T_j \in S\ ,\ v\in T_j$ که در این حالت دو آتاماتای T_i و T_j غیر فعال شده و آتاماتای معادل با $T_i \cup T_j$ فعال میشود که در نتیجه تعداد عناصر مجموعه S یک واحد کاهش می‌یابد.

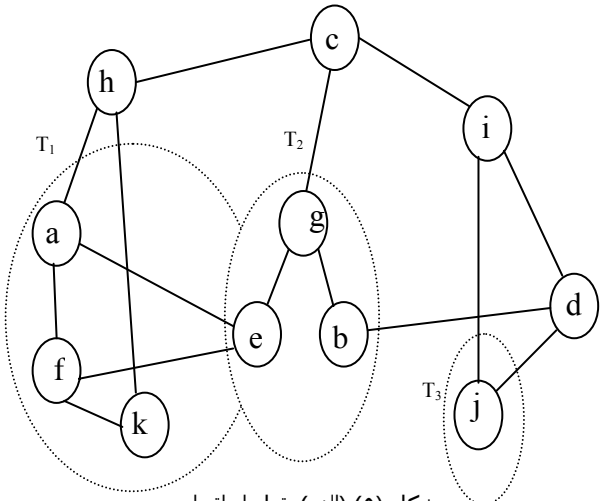
بدیهی است که اعضای S در طول زمان تغییر میکنند و به مرور تعداد آنها کاهش می‌یابد تا بالاخره یک آتاماتای فعال در S باقی می‌ماند. حال بسته به مناسب بودن اقدامهای انجام شده آتاماتاها پاداش گرفته و یا جریمه میشوند. این عملیات آنقدر تکرار شده تا آتاماتاها به سمت انتخاب اقدام بهینه همگرا شوند.

شکل (۵) یک اتوماتای یادگیر توزیع شده تعمیم یافته را نشان میدهد. این سیستم میتواند دارای 2^{11} اتوماتای یادگیر باشد که از بین آنها سه اتوماتای یادگیر T_1 ، T_2 و T_3 فعال هستند و لازم است که ذخیره شوند. شکل (۵) (الف) سیستم را قبلا از انتخاب اقدام توسط اتوماتاهای یادگیر فعال و شکل (۵) (ب) مجموعه S را بعد از انتخاب و اعمال اقدام توسط اتوماتاهای یادگیر فعال نشان میدهد. قبل از انتخاب اقدام، مجموعه S به قرار زیر است

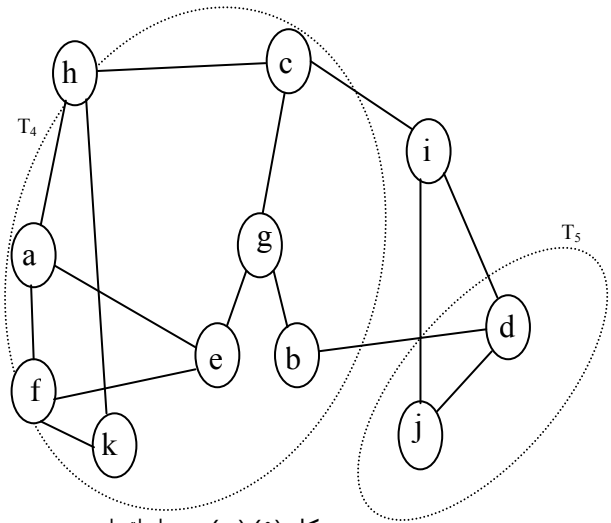
$$S = \{T_1, T_2, T_3\},\ T_1 = \{a, f, k\}$$
$$,\ T_2 = \{b, e, g\}, T_3 = \{j\}$$

بعد از انتخاب و اعمال اقدام (f, e) توسط اتوماتای یادگیر T_1 ، انتخاب و اعمال اقدام (g, c) توسط اتوماتای یادگیر T_2 و انتخاب و اعمال اقدام (j, d) توسط اتوماتای یادگیر T_3 مجموعه S به صورت زیر خواهد بود.

$$S = \{T_4, T_5\}.\ T_4 = \{a, b, c, e, f, k, g\},\ T_5 = \{j, d\}$$



شکل (۵) (الف). قبل از اقدام



شکل (۵) (ب). بعد از اقدام

۳- الگوریتم پیشنهادی برای حل مساله درخت اشتاینر بهینه

الگوریتم پیشنهادی از یک GDLA متناظر با گراف ورودی استفاده میکند. در ابتدای هر تکرار از الگوریتم مجموعه S GDLA دارای n اتوماتای یادگیر که هر کدام از آنها متناظر با یکی از روس گراف مساله است میباشد. سپس GDLA شروع به کار میکند که در مرحله آن اتوماتاهای فعال اقدام به انتخاب و اعمال یکی از اقدامهای خود مینماید که در نتیجه آن یا تعداد عناصر مجموعه S تغییر نمیکند و یا اینکه تعداد اعضای مجموعه S یک واحد کاهش می‌یابد. درخت ایجاد شده زمانی که $|S| = 1$ میباشد همه گره‌های ترمینال را در بر دارد. پس از حذف شاخه‌هایی که به هیچ ترمینالی ختم نمیشوند از درخت بدست آمده درختی بدست می‌آید که تمام برگهای آن ترمینال هستند. اگر هزینه‌ای این درخت از هزینه درخت جاری کمتر باشد در این صورت احتمال انتخاب اقدامهای اتوماتاهای تولید شده در طی این تکرار افزایش میابد و اگر هزینه‌ای این درخت از هزینه درخت جاری بیشتر باشد در این صورت احتمال انتخاب یادگیر اقدامهای اتوماتاهای تولید شده در طی این تکرار کاهش میابد. تعداد تکرارهای الگوریتم بایستی

الگوریتم مورد نظر در شکل (۶) نمایش داده شده است.

فرض کنید تعداد اقدامهای آتاماتای T_i برابر با r_i و بردار احتمال انتخاب اقدامهای آن $P^i = \left[p_1^i, p_2^i, \ldots, p_{r_i}^i \right]$ باشد. با توجه به اینکه اتوماتاهای یادگیر دارای اقدامهای مشترک هستند بروز رسانی بردارهای احتمال انتخاب اقدامها ممکن است به شرایطی بیانجامد که مجموع در

GDLA احتمالات بردار P^i یک نباشد. از این رو باید مقدار احتمالات را نرمالیزه کرد تا جمع آنها دوباره یک گردد. اگر احتمال فعلی اقدام jدر آتاماتای T_i را با $p_j^{i\,old}$ نمایش دهیم آنگاه رابطه (۱) چگونگی نرمالیزه کردن احتمالات را نشان می‌دهد.

$$p_j^i = \frac{p_j^{i\,old}}{\sum_{k=1}^{r_i} p_k^{i\,old}} \; ; \; for \;\; j = 1, \ldots, r_i \qquad (۱)$$

آزمایشها نشان داده است که دخالت هزینه یالهای گراف در انتخاب اقدام توسط آتاماتاهای باعث افزایش کیفیت جوابهای تولید شده توسط الگوریتم میگردد. برای این منظور بردار احتمال انتخاب اقدام برای هر T_i را بصورت موقتی به بردار P'^i تغییر می‌دهیم و پس از انتخاب اقدام بردار را دوباره به مقادیر قبلی خود بازمی‌گردانیم. P'^i برای هر T_i $(T_i \in S)$ بصورت رابطه (۲) محاسبه می‌شود.

$$P'^i = \left[p_1'^i, p_2'^i, \ldots, p_{r_i}'^i \right]$$

$$p_j'^i = \frac{\left(p_j^i\right)^\alpha . \left(\eta_j^{(i)}\right)^\beta}{\sum_{k=1}^{r_i} \left(p_k^i\right)^\alpha . \left(\eta_k^{(i)}\right)^\beta} \; ; \; for \;\; j = 1, 2, \ldots, r_i \qquad (۲)$$

که α و β پارامترهای ثابت هستند و

$$\eta_j^{(i)} = \frac{1}{\min\{c(k,j);\, k \in T_i\} + \gamma \psi_j^{(i)}} \; ; \; j \notin T_i \qquad (۳)$$

که γ یک پارامتر ثابت و $c(k,j)$ هزینه یال (k,j) است و

$$\psi_j^{(i)} = \min_k \{d(j,k)\} \; ; \; k \in \bigcup_{p \neq i} T_p \, , \, j \notin T_i \qquad (۴)$$

که T_p یک آتاماتای یادگیر غیراز T_i بوده و $d(j,k)$ کوتاهترین مسیر از گره j به گره k در گراف ورودی می‌باشد.

بعد از تمام یک تکرار و تولید یک درخت اشتاینر، هزینه آن محاسبه شده و با هزینه درخت اشتاینر جاری *(min_cost)* مقایسه میگردد. اگر هزینه درخت تولید شده از هزینه جاری *(min_cost)* کمتر باشد در این صورت اقدامهایی که انتخاب آنها منجر به انتخاب یالی شده است که در درخت موجود است، پاداش میگیرد و در صورتیکه هزینه درخت تولید شده از هزینه جاری بیشتر باشد در این صورت اقدامهایی که انتخاب آنها منجر به انتخاب یالی شده است که در درخت موجود نیست جریمه میشود. الگوریتم بروزرسانی بردار احتمالات بر اساس روش LRP میباشد. قوانین بروزرسانی احتمالات در صورت انتخاب اقدام j توسط آتاماتای T_i و گرفتن پاداش بر طبق روابط (۵) و (۶) میباشد.

$$p_j^{i\,new} = p_j^{i\,old} + a.\left(1 - p_j^{i\,old}\right) \qquad (۵)$$

$$p_k^{i\,new} = (1-a).p_k^{i\,old} \; ; \; \forall k \;\; k \neq j \qquad (۶)$$

قوانین بروزرسانی احتمالات در صورت انتخاب اقدام j توسط آتاماتای T_i و گرفتن جریمه بر طبق روابط (۷) و (۸) میباشد.

$$p_j^{i(new)} = (1-b).p_j^{i(old)} \qquad (۷)$$

$$p_k^{i(new)} = \frac{b}{r_i - 1} + (1-b).p_k^{i(old)} ; \forall k \ k \neq j \qquad (۸)$$

که در آن b پارامتر ثابت جریمه، a پارامتر تشویق و $p_j^{i(old)}$ احتمال فعلی اقدام j در آتاماتای T_i می‌باشد. پارامتر a طبق رابطه (۹) بروز میگردد.

$$a = \frac{current\ itration}{\max\ iteration}.\left(1 - \frac{c}{c_{first}}\right) \qquad (۹)$$

که در آن c هزینه درخت بدست آمده در مرحله جاری و c_{first} هزینه درخت بدست آمده در اولین مرحله است. در این بروزرسانی میزان پاداش بستگی به میزان خوب بودن جواب فعلی دارد به این معنی که هر چه جواب بهتری بدست آید پاداش بیشتری داده می‌شود.

۴- نتایج آزمایشها

آزمایشها برروی مجموعه B از دسته مسائل *Beasly* [۸] انجام گرفته است. در جدول (۱)، درصد خطای حاصل از روش پیشنهادی بهمراه درصد خطای روشهای [۲] و[۴] ارائه شده است.

برای روش پیشنهادی خود در یک ستون *(% GDLA avg.)* میانگین درصد خطا برای میانگین ۱۰ بار اجرا در ستونی دیگر *(%GDLA best)* بهترین درصد خطا در بین ۱۰ بار اجرا آمده است. در جدول (۲) نتایج روش پیشنهادی با نتایج گزارش شده در [۳-۷] مقایسه شده است. همانطور که مشاهده می‌شود که در اکثر موارد روش پیشنهادی جواب بهتری در مقایسه با سایر روشها تولید کرده است. مقادیر پارامترهای برای انجام آزمایشها عبارتند از

$$b = 0.01,\ \beta = 0.3,\ \alpha = 0.05$$
$$\gamma = 1,\ total\ \#\ of\ iterations = 500 \text{ ،}$$

قابل ذکر است که چون بعضی از اقدامها بین بعضی از آتاماتونها مشترک میباشند پس میتوان تعداد کل اقدامهای آتاماتونها را برابر با تعداد یالهای گراف در نظر گرفت. از این رو در پیاده‌سازی الگوریتم میتوان مقدار احتمال انتخاب هر اقدام را به هر یال از گراف وابسته کرد و در نتیجه با وجودیکه از نظر تئوری تعداد کل آتاماتونها در بدترین حالت برابر با تعداد زیرمجموعه های رئوس گراف میباشد ولی در عمل چنین چیزی اتفاق نمی‌افتد و با پیاده سازی مورد نظر حافظه مصرفی برای ذخیره مقدار احتمالات برای اقدامها برابر با $O(|E|)$ می‌باشد.

۵- نتیجه‌گیری

در این مقاله الگوریتمی مبتنی بر آتاماتای یادگیر توزیع شده برای حل مسئله درخت اشتاینر کمینه پیشنهاد گردید. آزمایشها نشان دادکه الگوریتم پیشنهادی در مقایسه با روشهای گزارش شده مانند الگوریتمهای ژنتیکی و کلونی مورچه‌ها از کارایی بالاتری بر خوردار است.

Construct a Generalized distributed learning automata *GDLA = (V, E, S)* // Initialization step
Set the probability vector for each automata as randomly
min_cost ← - ∞
For *i* ← *1* **to** *Total_No_of_Iterations* **do** // Iteration Step
S = {Tj | v∈Tj and v is a terminal node and |Tj| = 1} // *S* is the set of active automata
While *|S| > 1* **do**
For each automaton *Ti ∈ S* do
Select the best action *(u,v)* whose normalized probability is maximum
If *∀ Tj ∈ S; v ∉ Tj* **then**
Deactivate T_i and activate the automaton corresponding to $T_i \cup \{v\}$
Else *//∃ Tj ∈ S; v ∈ Tj*
Deactivate T_i and T_j and activate the automaton corresponding to $T_i \cup T_j$
End if
End for
End while
Delete extra edges which don't reach to any terminal node from the remaining unique automaton $T_j \in S$ and call it tree *T*
If cost of *T* < *min_cost* **then**
min_cost ← cost of T
Reward all of the automata according to the L_{RP} algorithm // Reward Step
Else // cost of *T ≥ min_cost*
Penalize all of the automata according to the L_{RP} algorithm // Penalty Step
End if
End for
Steiner tree is the best tree during the above loop whose cost is *min_cost*

شکل (۶): الگوریتم تولید درخت اشتاینر با کمک آتاماتای یادگیر توزیع شده تعمیم یافته

جدول (۱): مقایسه خطای روش پیشنهادی (GDLA) با خطای روشهای ارائه شده در[۴] و [۶]

شماره گراف	تعداد گره	تعداد کمان	تعداد ترمینال	هزینه بهینه	SDG %	SPH %	ADH %	ACS avg %	ACS best %	GDLA avg %	GDLA best %
۱	۵۰	۶۳	۹	۸۲	۰	۰	۰	۰	۰	۰	۰
۲	۵۰	۶۳	۱۳	۸۳	۸/۴۳	۰	۰	۰	۰	۰	۰
۳	۵۰	۶۳	۲۵	۱۳۸	۱/۴۵	۰	۰	۰	۰	۰	۰
۴	۵۰	۱۰۰	۹	۵۹	۸/۴۳	۵/۰۸	۵/۰۸	۲/۷	۰	۰/۶۷	۰
۵	۵۰	۱۰۰	۱۳	۶۱	۴/۹۲	۰	۰	۰	۰	۰	۰
۶	۵۰	۱۰۰	۲۵	۱۲۲	۴/۹۲	۳/۲۸	۱/۶۴	۱/۸	۰/۸۲	۱/۴۷	۰
۷	۷۵	۹۴	۱۳	۱۱۱	۰	۰	۰	۰	۰	۰	۰
۸	۷۵	۹۴	۱۹	۱۰۴	۰	۰	۰	۰	۰	۰	۰
۹	۷۵	۹۴	۳۸	۲۲۰	۲/۲۷	۰	۰	۰/۰۵	۰	۰/۰۴	۰
۱۰	۷۵	۱۵۰	۱۳	۸۶	۱۳/۹۵	۴/۶۵	۴/۶۵	۳/۸۴	۰	۳/۳	۰
۱۱	۷۵	۱۵۰	۱۹	۸۸	۲/۲۷	۲/۲۷	۲/۲۷	۱/۵۹	۰	۰/۶۸	۰
۱۲	۷۵	۱۵۰	۳۸	۱۷۴	۰	۰	۰	۰/۱۱	۰	۰	۰
۱۳	۱۰۰	۱۲۵	۱۷	۱۶۵	۶/۰۶	۷/۸۸	۴/۲۴	۰	۰	۰	۰
۱۴	۱۰۰	۱۲۵	۲۵	۲۳۵	۱/۲۸	۲/۵۵	۰/۴۳	۰/۰۹	۰	۰/۴۲	۰
۱۵	۱۰۰	۱۲۵	۵۰	۳۱۸	۲/۲۰	۰	۰	۰	۰	۰	۰
۱۶	۱۰۰	۲۰۰	۱۷	۱۲۷	۷/۸۷	۳/۱۵	۰	۰/۱۶	۰	۰	۰
۱۷	۱۰۰	۲۰۰	۲۵	۱۳۱	۵/۳۴	۳/۸۲	۳/۰۵	۱/۲۲	۰	۰/۸۳	۰
۱۸	۱۰۰	۲۰۰	۵۰	۲۱۸	۴/۵۹	۱/۸۳	۰	۱/۴۲	۰/۹۲	۰/۶۴	۰

جدول (۲): مقایسه نتایج الگوریتم پیشنهادی با الگوریتمهای گزارش شده در [۵] و [۶]

شماره گراف	تعداد گره	تعداد کمان	تعداد ترمینال	هزینه بهینه	حالت متوسط [۳]	بهترین حالت [۳]	حالت متوسط [۴]	بهترین حالت [۴]	حالت متوسط GDLA	بهترین حالت GDLA
۵	۵۰	۱۰۰	۱۳	۶۱	۶۱	۶۱	۶۱	۶۱	۶۱	۶۱
۱	۵۰	۶۳	۹	۸۲	۸۲	۸۲	۸۲	۸۲	۸۲	۸۲
۳	۵۰	۶۳	۲۵	۱۳۸	۱۳۸	۱۳۸	۱۳۸	۱۳۸	۱۳۸	۱۳۸
۱۱	۷۵	۱۵۰	۱۹	۸۸	۸۹	۸۸	۸۹/۴	۸۸	۸۸/۶	۸۸
۱۴	۱۰۰	۱۲۵	۲۵	۲۳۵	۲۳۵/۳	۲۳۵	۲۳۵/۲	۲۳۵	۲۳۶	۲۳۵
۱۸	۱۰۰	۲۰۰	۵۰	۲۱۸	۲۲۵/۵	۲۲۴	۲۲۱/۱	۲۲۰	۲۱۹/۴	۲۱۸

مراجع

[1] Noga Alon and Yossi Azar, "*On-line Steiner Trees Euclidean Plane*", Symposium on Computational Geometry, 1992.

[2] Plesnik J., "*Worst Case Relative Performances of Heuristics for the Steiner Problems in Graphs*", Acta Math, Vol. 2, pp. 269-284, 1991.

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

[4] S. Ding and N. Ishii, "*An Online Genetic Algorithm for Dynamic Steiner Tree Problem*", Symposium on Computational Geometry, pp. 337-343, 2000.

[5] S. Das, S. V. Gosavi, W. H. Hsu and S. A. Vaze, "*An Ant Colony Approach for The Steiner Tree Problem*", Proc. Of the Genetic and Evolutionary Computation Conference, 2002.

[6] M. Hashemi, P. Adibi, A. Jahanian and A. Nourollah, "*Dynamic Steiner Tree Problem by Using Ant Colony System*", *9th Annual CSICC*, pp. 472-465. Feb. 2004

[7] F. Bauer and A. Verma, "*Degree-Constrained Multicasting in Point-to-Point Networks*", *Proc. IEEE INFOCOM '95*, 1995.

[8] J. E. Beasley, "*OR-Library: Distributing Test Problems by Electronic Mail*", Operational Research. Soc., vol. 41, no. 11, pp. 1069-1072, 1990.

[9] M. R. Meybodi and H. Beigy, "*Solving Stochastic Shortest Path Problem Using Distributed Learning Automata*", Proceedings 7th Annual CSI Computer Conference, University of Isfahan's Computer Enginnering Department, Isfahan, Iran, 2001.

زیرنویس‌ها

[1] Orthogonal
[2] Multi Cast Routing
[3] the Average Distance Heuristic
[4] the Shortest Path Heuristic
[5] the Ant Colony System
[6] Action
[7] Generalized Distributed Learning Automata