تعیین درخت پوشای کمینه در گرافهای تصادفی بکمک اتوماتای یادگیر توزیع شده

جواد اکبری
دانشکدهٔ مهندسی کامپیوتر
دانشگاه آزاد اسلامی
اراک ایران

محمد رضا میبدی
دانشکدهٔ مهندسی کامپیوتر و فن آوری اطلاعات
دانشگاه صنعتی امیر کبیر
تهران ایران

چکیده: در این مقاله یک الگوریتم مبتنی بر اتوماتای یادگیر توزیع شده بمنظور تعیین درخت پوشای کمینه در یک گراف تصادفی که در آن تابع توزیع وزنهای یالهای گراف از قبل شناخته شده نمیباشند پیشنهاد می گردد. در این الگوریتم، به کمک یک اتوماتای یادگیر توزیع شده سعی میشود با حداقل تعداد نمونه گیری از یالهای گراف تصادفی درخت پوشای کمینه شناسایی شود. با انتخاب مناسب پارامترهای اتوماتای یادگیر توزیع شده الگوریتم پیشنهادی قادر است درخت پوشای کمینه را با احتمالی نزدیک به یک انتخاب نماید. به منظور ارزیابی الگوریتم پیشنهادی، تعداد نمونه های گرفته شده توسط الگوریتم پیشنهادی با تعداد نمونه های موردنیاز به روش نمونه گیری استاندارد مقایسه شده است. آزمایشها نشان داده است که تعداد نمونه های گرفته شده توسط الگوریتم پیشنهادی بمراتب از تعداد نمونه های گرفته شده به روش نمونه گیری استاندارد کمتر است.

کلمات کلیدی: گراف تصادفی، درخت پوشای کمینه، ، اتوماتاهای یادگیر، اتوماتای یادگیر توزیع شده

Using Distributed Learning Automata to Find Minimum Spanning Tree in Stochastic Graph

M. R. Meybodi
Computer Engineering Department
Amirkabir University of Technology
Tehran Iran

J. Akbari
Computer Engineering Department
Islamic Azad University
Arak Iran

Abstract: In this paper an algorithm based on distributed learning automata for finding minimum spanning tree in a stochastic graph when the weight density functions for the edges of the graph is unknown is propped. The proposed algorithm with the aid of distributed learning automata tries to find the minimum spanning tree with minimum number of sampling from the edges of the stochastic graph. Using simulation it has been shown that with proper selection of parameters of the distributed learning automata, the propped algorithm is able to find the minimum spanning tree with high propobability. As the algorithm proceeds, the process of sampling from the graph is concentrated on the edges of the minimum spanning tree with minimum expected cost. To evaluate the proposed algorithm, the number of sampling taken by the proposed algorithm is compared with the number of sampling needed by the standard sampling method. The result of comparison shows the efficiency of the proposed algorithm in terms of the number of samplings.

Keywords: Stochastic Graph, Minimal Spanning Tree, Larning Atomata, Dstributed Learning Automata

۱. مقدمه

یک گراف تصادفی[1] را می توان بوسیلهٔ سه تایی $G =< \mathrm{V}, \mathrm{E}, F >$ تعریف کرد بگونه ای که در آن $\mathbf{V} = \{v_1, v_2, ..., v_n\}$ مجموعهٔ رئوس گراف و $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V} = \{e_1, e_2, ..., e_m\}$ مجموعهٔ یالهای گراف در نظر گرفته شود. ماتریس $F_{n\times n}$ ماتریس توزیعهای احتمالی برای مشخصه ای از یالهای گراف (برای مثال طول یال) میباشد. درایهٔ f_{ij} از این ماتریس توزیع احتمالی[2] مقادیر مشخصهٔ یال (v_i, v_j) می باشد. زیرگراف تصادفی $G' =< \mathrm{V}', E', F >$ را *درخت پوشای تصادفی* گراف $G =< \mathrm{V}, E, F >$ میگوییم اگر این زیر گراف یک گراف

[1] Stochastic Graph
[2] Probability Density Function

تصادفی متصل از گراف $G =< \mathrm{V,E},F >$ بطوریکه $\mathrm{V}' = \mathrm{V}$ و $\mathrm{E}' \subseteq \mathrm{E}$ باشد. اگر $T = \{\tau_1, \tau_2, \tau_3, \ldots\}$ مجموعهٔ درخت های پوشای گراف تصادفی $G =< \mathrm{V,E},F >$ و $\overline{L}\,\tau_i$ متوسط وزن درخت پوشای $\tau_i \in T$ باشد در این صورت، درخت پوشای $\tau^* \in T$ درخت پوشای کمینهٔ تصادفی[3] گراف $G =< \mathrm{V,E},F >$ نامیده میشود اگر $\overline{L}\,\tau^* = \min_{\forall \tau_i \in T}\{\overline{L}\,\tau_i\}$.

روشهای مختلفی مانند روش حریصانه و برنامه سازی پویا برای حل مسالهٔ کوتاه ترین در خت پوشا در یک گراف قطعی[4] (گرافی که وزن یالهای آن معین و قطعی می باشد) ارایه گردیده است. الگوریتم Kruskal ,Prim میتوانند مسالهٔ درخت پوشای کمینه در یک گراف قطعی را در زمان چندجمله ای[5] حل نمایند. اما هنگامی که در یک شبکهٔ تصادفی وزن هر یک از یالهای گراف یک متغیر تصادفی گسسته با مؤلفه های آماری نامعین باشد تعیین درخت پوشای کمینه کار چندان آسانی نمی باشد. در این شرایط عامل یادگیر بایستی در یک محیط غیر قطعی و نامعین طی یک فرایند یادگیری پویا و تکرار پذیر نسبت به شناسایی مشخصه های آماری محیط و توزیع حاکم بر وزن یالهای گراف اقدام نماید وا بر اساس توزیع و متوسط وزنی یالها، درخت پوشای کمینه را شناسایی نماید[9,10]. Hutson, Shier روشی را مبتنی بر بکارگیری توزیع های کران دار بمنظور اعمال قیدهایی جهت تعیین وزن یالهای یک درخت پوشای کمینه در گرافهای تصادفی پیشنهاد کردند[17]. Ishii در [12] روشی را ارایه کرد که مبتنی بر شمارش مشاهدات پاسخ های مختلف ناشی از تغییرات یکنواخت پارامتر توزیع در بازهٔ خود، کوتاه ترین درخت پوشا را در یک گراف تصادفی تعیین می نماید. روش های دیگری نیز در [13,14,15,16] برای حل مسالهٔ درخت پوشای کمینه ارایه گردیده است. در [18] روشی بهینه جهت تخمین توزیع حاکم بر وزن یالها بمنظور شناسایی درخت پوشای بهینه در شبکه های تصادفی ارایه گردیده است. در تمامی این روشها فرض بر این است که توزیع احتمالی وزن یالهای گراف تصادفی شناخته شده می باشند.

در این مقاله یک الگوریتم مبتنی بر اتوماتای یادگیر توزیع شده بمنظور تعیین درخت پوشای کمینه در یک گراف تصادفی که در آن تابع توزیع وزنهای یالهای گراف تصادفی از قبل شناخته شده نمی باشند پیشنهاد می گردد. الگوریتم پیشنهادی، بکمک اتوماتای یادگیر توزیع شده سعی میکند یکی از درختهای پوشای گراف تصادفی را که متوسط وزن یالهای آن در بین تمامی درختهای پوشای گراف کمینه باشد با احتمال بالایی پیدا نماید. این الگوریتم، به کمک یک اتوماتای یادگیر توزیع شده سعی می کند با حداقل تعداد نمونه گیری درخت پوشای کمینه را شناسایی نماید. نشان داده میشود که با انتخاب مناسب پارامترهای اتوماتای یادگیر توزیع شده الگوریتم پیشنهادی قادر است درخت پوشای کمینه را با احتمالی نزدیک به یک انتخاب نماید. به منظور ارزیابی الگوریتم پیشنهادی، تعداد نمونه های گرفته شده توسط الگوریتم پیشنهادی با تعداد نمونه های مورد نیاز به روش نمونه گیری استاندارد مقایسه شده است. آزمایشها نشان داده است که تعداد نمونه های گرفته شده توسط الگوریتم پیشنهادی بمراتب از تعداد نمونه های گرفته شده به روش نمونه گیری استاندارد کمتر است. در این مقاله ابتدا بکمک روش نمونه گیری استاندارد، متوسط تعداد نمونه های اخذ شده از هر یال گراف مشروط به آنکه پارامتر خطای نمونه گیری در یک بازهٔ اطمینان از مقدار معینی کمتر باشد، تعیین می گردد. سپس نشان داده می شود که متوسط مجموع دفعات نمونه گیری از گراف در الگوریتم پیشنهادی، همواره به مقدار قابل توجهی از متوسط تعداد نمونه های اخذ شده در روش نمونه گیری استاندارد کمتر است.در ادامهٔ این مقاله و در بخش ۲ به ترتیب اتوماتاهای یادگیر و اتوماتاهای یادگیر توزیع شده به اختصار شرح داده میشود. در بخش ۳ الگوریتم پیشنهادی و در بخش ۴ نیز نتایج آزمایشها آمده است. بخش ۵ نتیجه گیری مقاله میباشد.

۲. اتوماتاهای یادگیر[6]

اتوماتای یادگیر [۸،۴،۵،۷]، ماشینی است که می‌تواند تعدادی متناهی عمل را انجام دهد. هر عمل انتخاب شده توسط یک محیط احتمالی ارزیابی می‌شود و نتیجه ارزیابی در قالب سیگنالی مثبت یا منفی به اتوماتا داده می‌شود و اتوماتا از این پاسخ در انتخاب عمل بعدی تأثیر می‌گیرد. هدف نهایی این است که اتوماتا یاد بگیرد تا از بین اعمال خود، بهترین عمل را انتخاب کند. بهترین عمل، عملی است که احتمال دریافت پاداش از محیط را به حداکثر برساند. کارکرد اتوماتای یادگیر در تعامل با محیط، در شکل۱ مشاهده می‌شود.

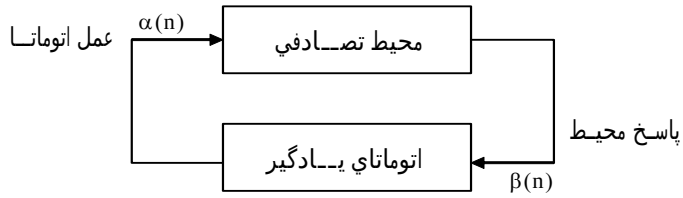
محیط را می‌توان توسط سه تایی $E \equiv \{\alpha, \beta, c\}$ نشان داد که در آن $\alpha \equiv \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ مجموعه ورودی‌ها، $\beta \equiv \{\beta_1, \beta_2, \ldots, \beta_m\}$ مجموعه خروجی‌ها و $c \equiv \{c_1, c_2, \ldots, c_r\}$ مجموعه احتمال‌های جریمه می‌باشد. هرگاه β مجموعه‌ای دوعضوی باشد، محیط از نوع P است. در چنین محیطی $\beta_1 = 1$ به عنوان جریمه و $\beta_2 = 0$ به عنوان پاداش در نظر گرفته می‌شود. در محیط از نوع Q، $\beta(n)$ می‌تواند به طور گسسته یک مقدار از مقادیر محدود در فاصله [۰،۱] را اختیار کند و در

---

[3] Stochastic Minimal Spanning Tree
[4] Deterministic Graph
[5] Polynomial Time
[6] Learning Automata

محیط از نوع S، $\beta(n)$ متغیر تصادفی در فاصله [۰,۱] است. c_i احتمال این‌که عمل α_i نتیجه نامطلوب داشته باشد می‌باشد. در محیط ایستا، مقادیر c_i بدون تغییر می‌مانند، حال آن‌که در محیط غیرایستا این مقادیر در طی زمان تغییر می‌کنند.

اتوماتای یادگیر با ساختار ثابت توسط پنج‌تایی $\{\alpha, \beta, F, G, \phi\}$ نشان داده می‌شود که در آن $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه عمل‌های اتوماتا، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_r\}$ مجموعه ورودی‌های اتوماتا، $\phi(n) \equiv \{\phi_1, \phi_2, ..., \phi_k\}$ مجموعه وضعیت‌های داخلی اتوماتا در لحظه n، $F: \phi \times \beta \to \phi$ تابع تولید وضعیت جدید اتوماتا و $G: \phi \to \alpha$ تابع خروجی می‌باشد که وضعیت کنونی اتوماتا را به خروجی بعدی می‌نگارد.



شکل(۱): ارتباط بین اتوماتای یادگیر و محیط

اتوماتای یادگیر با ساختار متغیر را می‌توان توسط چهارتایی $\{\alpha, \beta, p, T\}$ نشان داد که $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه اعمال اتوماتا، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_r\}$ مجموعه ورودی‌های اتوماتا، $p = \{p_1, .., p_r\}$ بردار احتمال انتخاب هریک از عمل‌ها و $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ الگوریتم یادگیری می‌باشد. الگوریتم زیر یک نمونه از الگوریتم‌های یادگیری خطی است. فرض می‌کنیم عمل α_i در مرحله nام انتخاب شود.

- پاسخ مطلوب از محیط

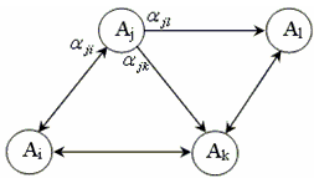
$$\begin{aligned} p_i(n+1) &= p_i(n) + a[1 - p_i(n)] \\ p_j(n+1) &= (1-a)p_j(n) \quad \forall j \quad j \neq i \end{aligned} \tag{۱}$$

- پاسخ نامطلوب از محیط

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

در روابط (۱) و (۲)، a پارامتر پاداش و b پارامتر جریمه می‌باشند. با توجه به مقادیر a و b سه حالت را می‌توان درنظر گرفت: اگر a و b با هم برابر باشند، الگوریتم را L_{RP}، هنگامی‌که b از a خیلی کوچکتر باشد، الگوریتم را $L_{R\varepsilon P}$ و اگر b مساوی صفر باشد آن را L_{RI} می‌نامیم[1,3,4,5,19].

اتوماتای یادگیر توزیع شده[۷]: اتوماتای یادگیر توزیع شده یک شبکه از اتوماتاهای یادگیر است که بمنظور حل یک مسالهٔ خاص به صورت گروهی با یکدیگر همکاری می نمایند[5]. اتوماتای یادگیر توزیع شده را می توان بوسیلهٔ یک گراف جهت دار، همانطور که در شکل ۲ دیده می شود، مدلسازی کرد بگونه ای که در آن مجموعه رئوس گراف متناظر است با مجموعهٔ اتوماتاهای یادگیر در سطح شبکه، ومجموعهٔ یالهای خروجی برای هر راس متناظر است با مجموعهٔ عمل های اتوماتای متناظر با آن راس. اتوماتای یادگیر توزیع شده را می توان توسط چند تایی $< V, E, T, V_0 >$ نمایش داد که در آن $V = \{v_1, v_2, ..., v_n\}$ معرف مجموعهٔ رئوس گراف متناظر با مجموعهٔ اتوماتا های یادگیر $A = \{A_1, A_2, ..., A_n\}$. $E \subset A \times A$ معرف یالهای گراف و $E_i = \{E_{i1}, E_{i2}, ..., E_{ij}, ..., E_{i\,ri}\}$ (r_i تعداد عمل های اتوماتای A_i) متناظر با مجموعهٔ عمل های اتوماتا A_i $\alpha_i = \{\alpha_{i1}, \alpha_{i2}, ..., \alpha_{ij}, ..., \alpha_{i\,ri}\}$ بگونه ای که یال E_{ij} متناظر است با عمل α_{ij} از اتوماتای A_i. و $T = \{T^1, T^2, ..., T^n\}$ معرف مجموعهٔ الگوریتم های یادگیر و V_0 گرهٔ ریشه را در اتوماتای یادگیر توزیع شده نشان می دهد[6].



شکل (۲): اتوماتاهای یادگیر توزیع شده

[7] Distributed Learning Automata

عملکرد اتوماتای توزیع شده را می توان بدین شکل شرح داد. در ابتدا اتوماتای یادگیر ریشه با توجه به بردار احتمالات یکی ازعمل های (یالهای خروجی) خود را انتخاب می نماید(فرضا عمل α_{1j}). انتخاب یال E_{1j} موجب فعال سازی اتوماتای یادگیری که در انتهای دیگر یال قرار دارد می شود. درسایر مراحل بعدی اتوماتای A_j (بازاء تمامی $j=1,2,...,n$) بر اساس بردار احتمال عمل $\underline{p}^j$ عمل α_{ji} را با احتمال p_i^j انتخاب می نماید که موجب فعال سازی اتوماتای A_i می گردد. این سلسله مراتب فعال سازی تا زمانی که یک اتوماتای برگ فعال گردد ادامه پیدا می کند. لازم به ذکر است که در هر تکرار همواره اتوماتای ریشه فعال سازی می شود اما سایر اتوماتاها در یک تکرار به صورت احتمالی فعال می شوند. انتخاب مسیرهای مختلف میان اتوماتای ریشه واتوماتاهای برگ آنقدر تکرار می گردد تا در نهایت یک مسیر با احتمالی نزدیک به ۱ انتخاب گردد. عمل انتخابی توسط اتوماتای برگ به محیط اعمال می گردد وپاسخ محیط به تمامی اتوماتاهای فعال شده در مسیر اتوماتای ریشهٔ تا اتوماتای برگ برگشت داده می شود. سپس هر یک از اتوماتاها بر اساس پاسخ دریافتی ازمحیط، بکمک الگوریتم یادگیری، نسبت به بروز کردن بردار احتمالات عمل خود اقدام می نمایند[4,8].

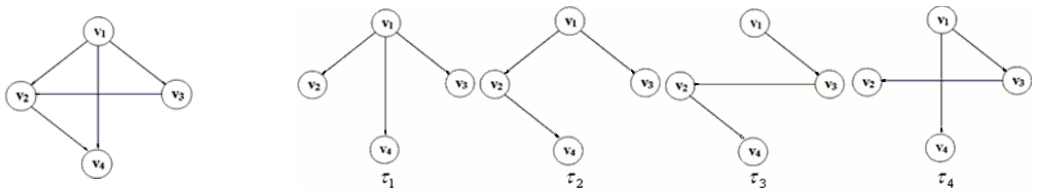
۳. الگوریتم پیشنهادی

مراحل الگوریتم پیشنهادی به شرح زیر است.

گام ۱. در مرحلهٔ نخست از الگوریتم یک شبکه از اتوماتاهای یادگیر را متناظر با گراف تصادفی وزن دار $G=<V,E,W>$ به گونه ای ایجاد مینماییم که در آن $V=\{v_1,v_2,...,v_n\}$ مجموعهٔ رئوس گراف متناظر با شبکهٔ اتوماتاهای یادگیر $A=\{A_1,A_2,...,A_n\}$ و $E\subseteq A\times A=\{E_1,E_2,...,E_n\}$ مجموعهٔ یالهای گراف بگونه ای که بازاء هر $E_i\in E$ مجموعهٔ $E_i=\{E_{(i,1)},E_{(i,2)},...,E_{(i,j)},...,E_{(i,ri)}\}$ (r_i تعداد عمل های اتوماتای A_i) متناظر با مجموعهٔ عمل های $\alpha_i=\{\alpha_{i1},\alpha_{i2},...,\alpha_{ij},...,\alpha_{i\,ri}\}$ اتوماتای A_i باشد. در این صورت یال $E_{(i,j)}$ متناظر است با عمل α_{ij} از اتوماتای A_i و عمل α_{ji} از اتوماتای A_j. وزن هر یک از یالهای گراف یک متغیر تصادفی $W_{i,j}$ با توزیعی گسسته و ناشاخته در نظر گرفته می شود که مقدار آن در هر مرحله میانگین وزن نمونه های تصادفی است که از آن یال گرفته شده است. در شکل ۳ به ترتیب یک گراف ورودی نمونه و تعدادی از درختهای پوشای آن نمایش داده شده است.

گام ۲. در این مرحله اتوماتاهای یادگیر به ترتیب با شروع از اتوماتای آغازین A_1 (متناظر با راس v_1) بطور متوالی فعال شده و یکی از عمل های خود را انتخاب می نمایند.

گام۳. مرحلهٔ دوم از الگوریتم تا زمانی که تعداد یالهای انتخابی توسط گروه اتوماتاهای فعال شده کمتر از $(n-1)$ یال باشد ویا ادامهٔ پیمایش گراف به هر دلیلی امکان پذیر نباشد، ادامه پیدا می کند.



شکل(۳): گراف نمونهٔ G و درخت های پوشای گراف G

گام ۴. در این گام طول درخت پوشای انتخابی در مرحلهٔ t ام یعنی $L\tau_i(t)$، محاسبه و با مقدار T_k(مقدار آستانهٔ[8] تعیین شده برای مرحلهٔ k ام) مقایسه میگردد. مقدار آستانه عبارتست از متوسط طول درخت های انتخابی تا این مرحله و بکمک رابطهٔ $T_k=(k-1)T_{k-1}+L(\tau_i)/k$ تعیین می گردد.

گام ۵. محیط با توجه به نتیجهٔ گام ۴، چنانچه $L\tau_i(k)\le T_k$ باشد (یعنی طول درخت پوشای پیمایش شده کمتر یا مساوی مقدار آستانه باشد)، آنگاه پاسخ محیط مطلوب بوده و تمامی اتوماتا های فعال شده بردار احتمال عمل خود را با دریافت پاداش ازمحیط اصلاح می نمایند. در غیر این صورت تمامی اتوماتاهای فعال شده از طرف محیط جریمه دریافت خواهند کرد.

گام ۶. عمل پیمایش درخت های پوشای گراف تا زمانی که حاصل ضرب احتمالات انتخاب یال های پیمایش شده از یک مقدار معین کمتر بوده و یا تعداد درخت های پیمایش شده از مقدار از پیش تعیین شده ای کمتر باشد، ادامه پیدا می کند. درختی که در آخرین پیمایش و قبل از خاتمهٔ الگوریتم انتخاب گردد به عنوان درخت پوشای بهینه در نظر گرفته می شود. کد الگوریتم پیشنهادی در شکل ۴ آمده است.

---

[8] Threshold

Algorithm : ***Stochastic Minimal Spanning Tree Problem***

Input: Stochastic Graph $G =< V, E, F >$

Output: The Minimal Spanning Tree;

Assumptions:

Let T_k be the dynamic threshold at stage k ;

Let τ_i be ith spanning tree;

Let $L\tau_i$ be the length of τ_i ;

begin Algorithm

Repeat

Let k be the stage number and initially set to 0;

Let v_1 be the first node**;**

while (the number of selected edges are less than $(|V|-1)$ or $v_i \neq v_n$) **do**

Node v_i selects one of its edges, according to its action probability vector, which connects it to node v_j ;

Add edge (v_i, v_j) to the set of selected edges;

Add cost of edge (v_i, v_j) to the cost of minimal spanning tree;

Update action probability vectors such that the formation of a cycle be impossible;

Set v_i to v_{i+1} ;

end while

if ($L\tau_i$ is less than T_k) **then**

Reward the selected action of automata;

else

Penalize the selected action of automata;

end if

Increment stage number k ;

Compute T_k as average cost of traversed spanning trees $T_k = \dfrac{(k-1)T_{k-1} + L(\tau_i)}{k}$

Until (probability of selecting spanning tree τ_i are greater than a pre specified threshold or stage number k are greater than another pre specified threshold);

End Algorithm;

شکل ۴: الگوریتم تعیین درخت پوشای کمینهٔ اتفاقی

در صورتی که از اتوماتاهایی با تعداد اعمال ثابت در طول فرایند یادگیری استفاده شود، پیدایش حلقه در دنبالهٔ عمل های انتخابی توسط اتوماتاهای یادگیر، موجب بروز خطا درعملکرد الگوریتم جهت تشخیص درخت پوشا خواهد شد. بنا براین، اگر عمل α_{ij} توسط اتوماتای A_i انتخاب شود، در این صورت هر اتوماتا متعلق به مجموعهٔ اتوماتاهای $\{A_k \mid A_k \notin \underline{A}^{\bullet}, k \neq i\}$ ($\underline{A}^{\bullet}$ عبارتست از مجموعه اتوماتاهایی که فعال شده اند) عمل های α_{ki} خود را بازاء تمامی $k \neq i$ حذف می نماید. بدین ترتیب که، هر اتوماتا با اصلاح بردار احتمال عمل خود، احتمال انتخاب عمل مورد نظر را تا نزدیک صفر کاهش داده و در مرحلهٔ بعد مجددا آنرا فعال می نماید. [1,2,7]. الگوریتم های مشابهی به ترتیب در [5] برای پیمایش کوتاه ترین مسیر میان دو نقطه از یک گراف تصادفی و در [2] برای حل مسالهٔ کولونی مورچه ها بکمک گروه اتوماتاهای یادگیر ارایه گردیده است، اما با توجه به آن که در تعیین درخت پوشای کمینه، در برخی موارد ممکن است یک راس دلخواه از گراف با بیش از یک یال در درخت پوشا شرکت داشته باشد، بنا براین، فرایند حذف عمل برای الگوریتم پیشنهادی به شکل زیر بیان می شود. با فرض آنکه مسیر $\pi_{i,j} = \{E_{(i,k)}, \pi_{k,l}, E_{(l,j)}\}$ بازاء تمامی $k,l \in V$ به شرط آنکه $i \neq l, j$ و $k \neq j$ دنبالهٔ یالهایی باشد که راس v_i را به v_j متصل نماید و $p_j^i(t)$ و $q_j^i(t)$ بترتیب احتمال انتخاب یال $E_{(i,j)}$ و مسیر $\pi_{i,j}$ در مرحلهٔ t ام باشند، حال اگر یال $E_{(i,j)}$ در t امین مرحله از الگوریتم توسط اتوماتای A_i انتخاب گردد. در این صورت می بایستی احتمال انتخاب یال $E_{(k,i)}$ بازاء هر مسیر $\pi_{j,k}$ به سمت صفر میل نماید.

لازم به ذکر است که درگام دوم از الگوریتم پیشنهادی ممکن است در برخی از حالات، شرایطی بروز نماید که تحت آن وجود داشته باشد اتوماتایی همچون A_i بگونه ای که $\{\forall j = 1,2,...,r_j \; \exists \; p_j^i(t) = 0 \mid p_j^i \in \underline{p}^i = (p_1^i, p_2^i, ..., p_j^i, ..., p_{ri}^i)\}$. در این حالت عملیات پیمایش درخت پوشای گراف را با واسطهٔ اتوماتای A_j (بازاء $j = i+1, i+2, ..., n$) ادامه می دهیم.

نمونه گیری استاندارد: برای بدست آوردن تعداد نمونه برداریهای های مورد نیاز از هر یال، در گراف ورودی G، بگونه ای که بتوان تضمین کرد انحراف متوسط طول نمونه های اخذ شده از متوسط واقعی آن با نرخ اطمینان مطلوبی(مثلا ۹۰ درصد) کمتر از ε باشد(ε

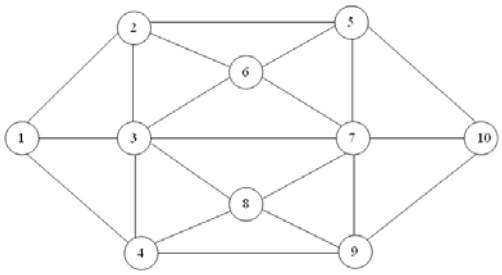
خطای نمونه گیری خوانده می شود و مقداری دلخواه وبه قدر کافی کوچک دارد). این روش را نمونه گیری استاندارد می نامیم. بـرای ایـن کار میتوان از قضیه زیر استفاده کرد. به کمک این قضیهٔ می توان تعداد دفعات نمونه برداری را بگونه ای انتخـاب کـرد کـه خطـای نمونـه گیری ε در یک بازهٔ اطمینان سمت صفر میل نماید[20].

قضیهٔ: فرض کنید $x_1, x_2, ..., x_n$ دنباله ای از متغیر های تصادفی مستقل با توزیع یکسان باشند، بگونه ای که $E(x_i) = \mu$ و $Var(x_i) = \sigma^2$. با فرض آنکه متوسط μ با اطمینان $1-\delta$ در بازهٔ $\bar{x} \pm \frac{\sigma}{\sqrt{n\delta}}$ قرار داشته باشد. آنگاه بازاء هر مقدار دلخواه وبه قدر کافی کوچک برای ε، وجود دارد n_0 ای بگونه ای که بازاء تمامی مقادیر $n \geq n_0$ خواهیم داشت.

$$\forall \varepsilon \, \exists n_0 \; \forall n \geq n_0 \; ; \quad p\{|\bar{x}_n - \mu| \prec \varepsilon\} \succ 1-\delta$$

۴. نتایج آزمایشها

در هر آزمایش ابتدا بکمک روش نمونه گیری استاندارد و با توجه به نتایج تحلیلی بدست آمده در بخش قبل، حداقل تعداد نمونه های مورد نیاز از هریال، بگونه ای که متوسط نمونه های اخذ شده بازاء مقدار معین و به قدرکافی کوچک پارامتر خطای ε با احتمال $1-\delta$ در بازهٔ $(\mu-\varepsilon, \mu+\varepsilon)$ قرار داشته باشد، تعیین می گردد. در تمامی آزمایشات مقدار $\varepsilon = 0.001$ در نظر گرفته شده است. تعداد نمونه گبریهای مورد نیاز در این روش با تعداد نمونه های اخذ شده توسط الگوریتم پیشنهادی مقایسه میگردد و نشان داده میشود که تعداد نمونه گیری های انجام گرفته توسط الگوریتم پیشنهادی به مراتب کمتر از تعداد نمونه گبریهای انجام شده به روش استاندارد میباشد. برای پیدا کردن یکی از درختهای پوشای کمینه میتوان با اجرای الگوریتم Kruskal بر روی گراف، که در ان طول هر یال متوسط طولهای نمونه گیری شده از آن یال میباشد، درخت پوشای کمینه را بدست آورد. به منظور ارزیابی الگوریتم پیشنهادی درخت پوشای کمینه بدست آمده از این طریق میتواند با درخت بدست آمده توسط الگوریتم پیشنهادی مقایسه گردد.



شکل(۵): گراف تصادفی G

الگوریتم پیشنهادی بر روی گراف تصادفی G(شکل ۵) که تابع توزیع احتمالی طول یالهای آن در جدول ۱ نشان داده شده است، آزمایش میگردد. بازهٔ اطمینان بین ۵۰٪ تا ۹۹٪ تغییر داده شده و برای هر مقدار مجموع تعداد نمونه های مورد نیاز از یالها تعیین گردیده است که نتایج آن در جدول ۳ و نمودار ۱ آمده است. همچنین تعداد نمونه های مورد نیاز از هر یال بازاء بازه های اطمینان ۵۰٪، ۷۰٪، ۹۰٪ و۹۹٪ و با در نظر گرفتن $\varepsilon = 0.001$، در جدول ۲ ارایه گردیده است. تعداد نمونه های بدست آمده از این طریق با تعداد نمونه های گرفته شده توسط الگوریتم پیشنهادی مقایسه میشود. الگوریتم پیشنهادی که از الگوریتم یادگیری L_{R-I} استفاده میکند برای مقادیر مختلف پارامتر پاداش برای گراف شکل ۵ آزمایش گردیده است. نتایج ارایه شده در جدول ۴ و نمودارهای ۲ تا ۴ متوسط نتایج بدست آمده برای ۱۰۰ بار تکرار آزمایش میباشند. الگورینم زمانی خاتمه می یابد که احتمال انتخاب درخت پوشای انتخاب شده در یک تکرار به بیش از ۰,۹۵ برسد. احتمال انتخاب درخت پوشا حاصلضرب احتمال انتخاب یالهای موجود در درخت پوشا میباشد.

کاهش مقدار پارامتر a در الگوریتم پیشنهادی موجب افزایش نرخ همگرایی الگوریتم به پاسخ بهینه میگردد ولی درمقابل باعث افزایش تعداد نمونه گیریها از یالهای گراف تصادفی میگردد. این مساله که چه رابطه ای بین تغییرات پارامتر a و تغییرات بازهٔ اطمینان در نمونه گیری استاندارد وجود دارد دست مطالعه میباشد. همانطور که در جداول مشاهده میشود تعداد نمونه های گرفته شده توسط الگوریتم پیشنهادی به مراتب کمتر از تعداد نمونه های مورد نیاز در روش استاندارد میباشد. در جدول ۴ به ترتیب متوسط دفعات نمونه گیری از گراف توسط الگوریتم پیشنهادی، متوسط دفعات پیمایش درخت بهینه و نرخ همگرایی[9] الگوریتم به پاسخ بهینه بازاء مقادیر مختلف پارامتر پاداش برای گراف شکل ۵ نشان داده شده است. مشاهده می شود که با کاهش مقدار پارامتر پاداش سرعت همگرایی الگوریتم به مقدار قابل توجهی کاهش و از سوی دیگر صحت همگرایی افزایش می یابد. همچنین مشاهده می شود که بازاء مقادیر $a \leq 0.008$

[9] Convergence Rate

الگوریتم همواره به یکی از درختهای پوشای کمینه همگرا میشود. آزمایشها نشان داده است که با انتخاب مقدار مناسب برای پارامتر پاداش، الگوریتم میتواند همواره به یکی از درختهای پوشای کمینه همگرا شود.

Edge	Length				Probability			
(1,2)	3.00	5.30	7.40	9.40	0.20	0.20	0.30	0.30
(1,3)	3.50	6.20	7.90	8.50	0.30	0.30	0.20	0.20
(1,4)	4.20	16.10	6.90	8.90	0.20	0.30	0.20	0.30
(2,3)	9.50	2.30	3.60	4.50	0.20	0.20	0.30	0.30
(2,5)	2.60	4.10	5.50	9.00	0.20	0.20	0.40	0.20
(2,6)	5.80	7.00	8.50	9.60	0.30	0.30	0.20	0.20
(3,4)	2.10	3.20	4.50	6.80	0.20	0.20	0.30	0.30
(3,6)	6.80	7.70	8.50	9.60	0.40	0.10	0.10	0.40
(3,7)	6.50	7.20	8.30	9.40	0.50	0.20	0.20	0.10
(3,8)	5.90	7.80	8.60	9.90	0.40	0.30	0.10	0.20
(4,8)	7.00	8.00	8.80	9.40	0.40	0.20	0.20	0.20
(4,9)	1.10	2.20	3.50	4.30	0.20	0.30	0.40	0.10
(5,6)	0.60	1.50	3.90	5.80	0.20	0.20	0.30	0.30
(5,7)	3.20	4.80	6.70	8.20	0.20	0.20	0.30	0.30
(5,10)	6.30	7.80	8.40	9.10	0.20	0.20	0.40	0.20
(6,7)	2.10	4.80	6.60	7.50	0.20	0.40	0.20	0.20
(7,8)	1.60	2.80	5.20	6.00	0.20	0.30	0.30	0.20
(7,9)	3.50	4.00	5.00	7.70	0.20	0.30	0.30	0.20
(7,10)	1.60	3.40	8.20	9.30	0.20	0.20	0.20	0.40
(8,9)	1.70	4.90	6.50	7.80	0.10	0.20	0.40	0.30
(9,10)	4.60	6.40	7.60	8.90	0.40	0.10	0.20	0.30

جدول(۱): توزیع احتمالی طول یالهای گراف تصادفی G

Confidence Interval	Total Number of Samples
50	5105
55	5065
60	5398
65	5304
70	5747
75	5522
80	5595
85	6050
90	6844
92.5	7062
95	7632
97.5	9181
99	12797

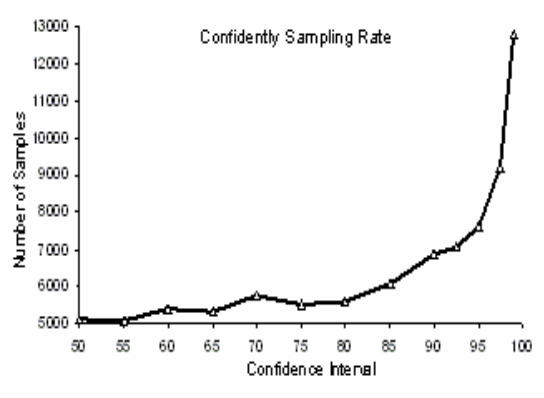
جدول(۳): متوسط تعداد کل نمونه ها ذر نمونه گیری استاندارد

Learning Parameter	Total Number of Samples	Number of traveling Optimal tree	Convergence Rate
0.007	9066	5236	100
0.008	8850	4346	100
0.009	6789	3214	98.89
0.01	5576	2854	97.78
0.03	1122	291	91.76
0.04	881	282	87.78
0.05	608	206	78.89
0.06	375	125	78.33
0.07	258	91	76.67
0.08	181	65	75
0.09	154	59	70
0.1	138	42	68.33

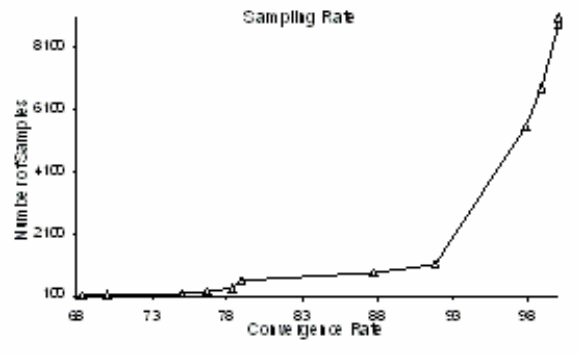
جدول(۴): متوسط تعداد کل نمونه های اخذ شده توسط الگوریتم پیشنهادی

	Confidence Interval			
Edge	50	70	90	99
(1,2)	228	399	342	400
(1,3)	238	205	234	416
(1,4)	423	372	570	427
(2,3)	566	615	621	492
(2,5)	221	303	408	377
(2,6)	231	253	297	480
(3,4)	198	223	299	421
(3,6)	180	223	272	481
(3,7)	166	176	238	1189
(3,8)	159	225	222	460
(4,8)	168	234	350	1303
(4,9)	159	191	262	920
(5,6)	209	232	291	496
(5,7)	227	212	290	369
(5,10)	124	170	349	1795
(6,7)	214	188	367	381
(7,8)	488	579	431	576
(7,9)	186	171	252	619
(7,10)	288	299	257	364
(8,9)	189	225	231	525
(9,10)	235	240	250	298

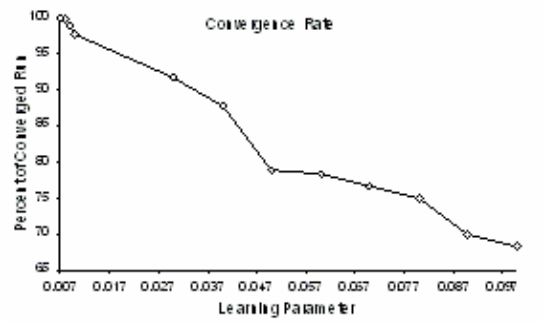
جدول(۲): متوسط تعداد نمونه های اخذ شده از هر یال در روش نمونه گیری استاندارد برای بازه های اطمینان مختلف



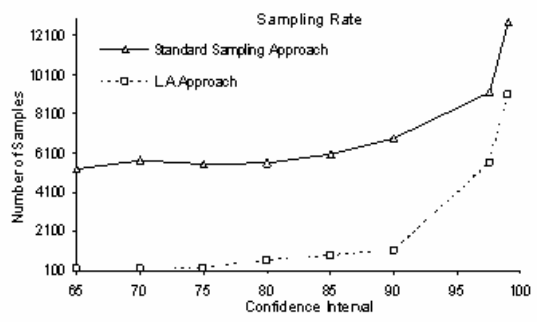
نمودار(۱): متوسط تعداد کل نمونه ها در نمونه گیری استاندارد



نمودار(۲): متوسط تعداد کل نمونه های اخذ شده توسط الگوریتم پیشنهادی



نمودار(۳): درصد اجراهای همگرا شده به پاسخ بهینه برای الگوریتمهای پیشنهادی بازاء پارامترهای پاداش مختلف



نمودار(۴): تعداد نمونه های گرفته شده برای الگوریتم پیشنهادی و نمونه گیری استاندارد

۵. نتیجه گیری

در این مقاله یک الگوریتم مبتنی بر اتوماتای یادگیر توزیع شده بمنظور تعیین درخت پوشای کمینه در یک گراف تصادفی که در آن تابع توزیع وزنهای یالهای گراف تصادفی از قبل شناخته شده نمیباشند پیشنهاد گردید. به منظور ارزیابی الگوریتم پیشنهادی، تعداد نمونه های گرفته شده توسط الگوریتم پیشنهادی با تعداد نمونه های موردنیاز به روش نمونه گیری استاندارد مقایسه گردید و نشان داده شد که تعداد نمونه های گرفته شده توسط الگوریتم پیشنهادی بمراتب از تعداد نمونه های گرفته شده به روش نمونه گیری استاندارد کمتر است.

مراجع

[1] K. Vrbeek, A. Nowe, M. Peeters and K. Tuyls, “Multi-Agent Coordination in Tree Structured Multi-Stage Games", Proc. of Fourth Symposium on Adaptive Agents and Multi-Agent Systems, 2003.

[2] K.Vrbeek, A.Nowe, “Colonies of Learning Automata”, IEEE Transactions on Systems, Man, Cybernetics-PartB: Cybernetics, Vol.32, pp. 772-780, Dec. 2002.

[3] P. S.Sastry, V. V.Phansalkar and M.A.L. Thathachat, “Decentralized Learning of Nash Equilibria in Multi-Person Stochastic Games with Incomplete Information”, IEEE Transactions on Systems, Man and Cybernetics, Vol.24, pp.769-777, May 1994.

[4] M. A. L. Thathachat, P. S. Sastry, “ A Hierarchical System of Learning Automata That Can Learn The Globally Optimal Path”, Information Science, Vol.42, pp.743-766, 1997.

[5] H. Beigy, M. R. Meybodi, “Utilizing Distributed Learning Automata to Solve Stochastic Shortest Path Problems”, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, Vol.14, pp. 591-615, Oct 2006.

[6] H. Beigy, “Intelligent Channel Assignment in Cellular Networks: A Learning Automata Approach”, PhD Thesis, Computer Engineering Department, Amir Kabir University of Technology, Tehran, Iran, 2006.

[7] M. A. L. Thathachar and B. R. Harita, "Learning Automata with Changing Number of Actions", IEEE Transactions on Systems, Man, and Cybernetics, vol. SMG17, pp. 1095-1100, Nov. 1987.

[8] M. A. L. Thathachat, V.V.Phansalkar, "Convergence of Teams and Hierarchies of Learning Automata in Connectionist Systems", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 24, pp.1459-1469, Nov. 995.

[9] P. Spira, A. Pan, "On Finding and Updating Spanning Trees and Shortest Paths", SIAM Journal on Computing, Vol. 4, No. 3, pp. 375–380, 1975.

[10] D. Frigioni, A. Marchetti-Spaccamela, U. Nanni, "Fully Dynamic Algorithms for Maintaining Shortest Paths Trees", Journal of Algorithms, Vol. 34, pp. 251–281, 2000.

[11] P. Narvaez, K. Siu, H. Tzeng, "New Dynamic Algorithms for Shortest Path Tree Computation", IEEE/ACM Transactions on Networking, Vol. 8, No. 6, pp. 734–746, 2000.

[12] H. Ishii, S. Shiode and T. Nishida, "Stochastic Spanning Tree Problem", Discrete Applied Mathematics, Vol. 3, pp. 263-273, 1981.

[13] M. Fredman and D. Willard, "Trans-Dichotomous Algorithms for Minimum Spanning Trees and Shortest Paths", In Proc. 31st Annual IEEE Symp. on Foundations of Computer Science, pp. 719-725, 1990.

[14] P. M. Vaidya, "Minimum Spanning Trees in k-Dimensional Space", SIAM Journal on Computing, Vol. 17, No. 3, pp.572–582, 1988.

[15] D. R. Karger, P.N. Klein and R.E. Tarjan, "A Randomized Linear-Time Algorithm for Finding Minimum Spanning Trees", J. Assoc. Comput. Mach., Vol. 4, No. 2, pp. 321-329, 1995.

[16] R. Ravi and M.X. Goemans " The Constrained Minimum Spanning Tree Problem", Proc. 5th Scandinavian Workshop on Algorithm Theory, pp. 66-75, LNCS 1097, Springer-Verlag, 1996.

[17] K. Hutson and D. Shier, “Bounding Distributions for the Weight of a Minimum Spanning Tree in Stochastic Networks”, Operations Research, Vol. 53, No. 5, pp. 879-886, 2005.

[18] K. Hutson and D. Shier, “Minimum Spanning Trees in Networks with Varying Edge Weights”, Annals of Operations Research, Vol. 146, No. 1, pp. 3-18, 2006.
[19] K. S. Narendra and K. S. Thathachar, "Learning Automata: An Introduction", New York, Printice-Hall, 1989.
[20] A. Papoulis, " Probability, Random Variables, and Stochastic Processes", 3rd Edition, New York, McGraw-Hill, 1991.