*مقاله پژوهشی*

# یک الگوریتم جدید مبتنی بر آتاماتای یادگیر توزیع‌شده برای حل مسئله بهینه‌سازی خطی تصادفی روی گروه جایگشت‌ها

محمدرضا ملاخلیلی میبدی و معصومه زجاجی

**چکیده: در این مقاله ابتدا نوعی از بهینه‌سازی جایگشت معرفی شده است. در این نوع بهینه‌سازی فرض گردیده که تابع هزینه، دارای یک تابع توزیع احتمال ناشناخته است. این فرض باعث می‌شود که پیچیدگی حل مسئله یافتن جایگشت بهینه که به دلیل بزرگی ذاتی فضای جواب‌ها پیچیده است، تشدید شود. یک الگوریتم مبتنی بر آتاماتای یادگیر توزیع‌شده برای حل مسئله از طریق انجام توأمان جستجو در فضای جواب‌های جایگشت و نمونه‌گیری از مقادیر تصادفی ارائه می‌دهیم. ضمن بررسی ریاضی رفتار الگوریتم جدید پیشنهادی، نشان می‌دهیم که با انتخاب مقادیر مناسب پارامترهای الگوریتم یادگیر، این روش جدید می‌تواند جواب بهینه را با احتمالی به اندازه دلخواه نزدیک به ۱۰۰٪ و از طریق هدفمندکردن جستجو به کمک آتاماتای یادگیر توزیع‌شده پیدا کند. نتیجه اتخاذ این سیاست، کاهش تعداد نمونه‌گیری‌ها در روش جدید در مقایسه با روش‌های مبتنی بر نمونه‌گیری استاندارد است. در ادامه، مسئله یافتن درخت پوشای کمینه در گراف تصادفی به عنوان یک مسئله بهینه‌سازی جایگشت تصادفی بررسی گردیده و راه حل ارائه‌شده مبتنی بر آتاماتای یادگیر برای حل آن به کار گرفته شده است.**

***کلیدواژه:*** **آتاماتای یادگیر، آتاماتای یادگیر توزیع‌شده، گراف تصادفی، درخت پوشای کمینه تصادفی.**

## ۱- مقدمه

در مسایل مرتبط با علوم و مهندسی، منظور از بهینه‌سازی، یافتن نقطه کمینه یا بیشینه یک تابع معین است که آن را تابع هدف می‌نامیم. این تابع در مسایل کمینه‌سازی با عنوان تابع هزینه نیز شناخته می‌شود [۱]. در یک طبقه‌بندی کلی، مسایل بهینه‌سازی به دو دسته عمده مسایل بهینه‌سازی ترکیبی و مسایل بهینه‌سازی پیوسته تقسیم می‌شوند. در مسایل بهینه‌سازی ترکیبی، دامنه تعریف مسئله دارای ماهیت گسسته است؛ یعنی جواب‌های ممکن برای مسئله بهینه‌سازی، یک مجموعه شمارش‌پذیر را تشکیل می‌دهند ولی در عین حال تابع هدفی که قصد بهینه‌سازی آن را داریم یک تابع پیوسته است. دسته دوم از مسایل بهینه‌سازی، اصطلاحاً مسایل بهینه‌سازی پیوسته نامیده می‌شوند. در حالت کلی، منظور از یک مسئله بهینه‌سازی پیوسته پیداکردن نقطه کمینه یا بیشینه سراسری تابع $f(x)$ تحت قیودی از نوع برابری یا نابرابری است [۲]. در صورت وجود قیود پیچیده، زیادبودن متغیرها، ناپیوسته‌بودن تابع، تصادفی‌بودن متغیرها و مواردی از این دست، یافتن جواب مسئله بهینه‌سازی از طریق روش‌های تحلیلی و یا حتی عددی ناممکن است.

امروزه یکی از روش‌های مرسوم برای حل این دسته از مسایل پیچیده بهینه‌سازی، استفاده از روش‌های هوشمند است [۳]. تقریباً در اکثر این روش‌ها که از سیستم‌های بیولوژیکی و فیزیکی موجود در طبیعت و یا مدل‌های پیشنهادی از نحوه رفتار یادگیرانه در انسان و یا حیوانات الگو گرفته‌اند، تعداد زیادی ذره در فضای مسئله پخش شده و هم‌زمان به دنبال جواب بهینه سراسری می‌گردند. هر یک از ذرات از هوش فردی بسیار کم و محدودی برخوردار هستند ولی به خاطر وجود یک همکاری دقیق دسته‌جمعی موفق به یافتن جواب بهینه سراسری می‌شوند. تمامی این روش‌ها را می‌توان نوعی جستجوی تصادفی حافظه‌دار به حساب آورد. تصادفی، از این نظر که هر یک از این ذرات تا حدودی به طور تصادفی فضای مسئله را جستجو می‌کنند و حافظه‌دار از این جهت که به مرور، احتمال حضور هر یک از این ذرات در نقطه بهتر (بهینه‌تر) با درس‌گرفتن از حرکت‌های گذشته افزایش می‌یابد.

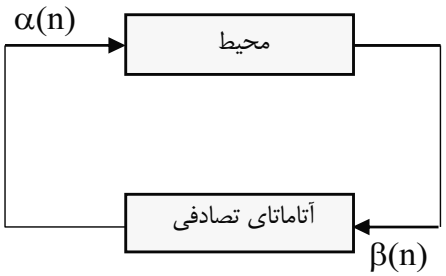
از نظر ریاضی، یک جایگشت $\sigma$ روی مجموعه $[n]=[1,2,\ldots,n]$ از $n$ شیء معین، یک تابع یک‌به‌یک از $[n]$ به $[n]$ است [۴]. روش دیگری که غالباً برای نمایش جایگشت $\sigma$ روی مجموعه $[n]$ استفاده می‌شود، توصیف آن به عنوان یک بردار $n$بعدی از $[n]^n$ و به شکل $\sigma=(\sigma(1),\ldots,\sigma(n))$ است. یافتن یک جایگشت کیفی خوب- یعنی مرتب‌سازی مقادیر یا اشیا با یک نظم معین- در بسیاری از مسایل نظیر رتبه‌بندی نتایج حاصل از جستجو، تخصیص وظیفه به CPU یا سیستم عامل [۵]، مسایل زمان‌بندی [۶] و نظایر آن مورد نیاز است. به عنوان مثال، مسئله فروشنده دوره‌گرد یا TSP را در نظر بگیرید. هدف از حل این مسئله، یافتن توری با کمترین مسافت میان تعدادی شهر است. هر جواب برای این مسئله را می‌توان با یک جایگشت که ترتیب شهرها را نمایش می‌دهد، نشان داد [۷].

به عنوان نمونه‌ای دیگر، مسئله درخت پوشای کمینه یا MST را در نظر بگیرید. یک روش جستجو برای یافتن جواب این مسئله را می‌توان به صورت ترتیبی از انتخاب گره‌ها در نظر گرفت که در هر مرحله، گره انتخاب‌شده، یالی با کمترین وزن ممکن از میان مجموعه یال‌های قابل انتخاب خود را برمی‌گزیند، به گونه‌ای که مجموعه یال‌های انتخاب‌شده کمترین وزن ممکن را داشته باشند [۸]. در این مسئله از آنجا که انتخاب یک گره، انتخاب برخی یال‌ها را برای گره‌های باقیمانده ناممکن می‌کند- به منظور جلوگیری از ایجاد حلقه در درخت- ترتیب انتخاب گره‌ها اهمیت پیدا می‌کند.

این مقاله در تاریخ ۸ آبان ماه ۱۳۹۹ دریافت و در تاریخ ۴ بهمن ماه ۱۴۰۰ بازنگری شد.

محمدرضا ملاخلیلی میبدی، گروه کامپیوتر، واحد میبد، دانشگاه آزاد اسلامی، میبد، ایران، (email: mollakhalili@maybodiau.ac.ir).

معصومه زجاجی، گروه کامپیوتر، واحد میبد، دانشگاه آزاد اسلامی، میبد، ایران، (email: mzojaji@maybodiau.ac.ir).




شکل ۱: آتاماتای یادگیر تصادفی [۱۱].

نکته مشترک در تمام مسایلی از این دست، فضای جستجوی بزرگ- توانی بر حسب تعداد ورودی‌ها مثلاً تعداد شهرها یا تعداد گره‌ها یا تعداد وظایف- مسئله است. فرض بر این است که تابع هدفی وجود دارد که بهینه‌سازی آن مد نظر می‌باشد (مثلاً زمان انجام کل کارها در مسئله تخصیص وظایف در CPU، یا طول تور در TSP و یا وزن درخت پوشا در MST). این تابع هدف می‌تواند میزان کیفی‌بودن یک جواب را مشخص کند. با فرض وجود این تابع، می‌توان مسئله یافتن جایگشت بهینه را به این صورت تصور کرد: یک جایگشت مفروض $\sigma$ و یک تابع هدف $f$ داده شده است. این تابع قادر است برای هر جایگشت یک مقدار عددی تولید کند. تابع $f$ را می‌توان تابع هدف یک مسئله بهینه‌سازی تصور کرد. فرض بر این است که تابع $f$ تابع شناخته‌شده‌ای نیست و شبیه یک جعبه سیاه عمل می‌کند. فرض کنید مقادیر $f$ نرمال‌سازی شده‌اند. در این صورت می‌توان فرض کرد که برای هر جایگشت $\sigma$ به‌طوری که مقدار $f(\sigma)\in[۰,۱]$ باشد آنگاه $f(\sigma)=۰$ به معنای بهترین جایگشت یا جایگشت بهینه و $f(\sigma)=۱$ نشانگر بدترین جایگشت است.

در تمام نمونه‌هایی که تا کنون ذکر شد، فرض بر این می‌باشد که متغیرهای مسئله- هزینه پیمایش یک یال در مسئله TSP، وزن یک یال در MST و غیره- دارای مقادیر قطعی و یا تصادفی با توزیع احتمال شناخته‌شده هستند. مثلاً مدت زمان لازم برای انجام یک وظیفه توسط CPU عددی معین است یا مقدار وزن یال‌ها در مسئله MST و یا مسافت میان شهرها در TSP، اعدادی معین یا احتمالی با یک توزیع شناخته‌شده هستند.

اگر فرض کنیم که این مقادیر، مقادیری احتمالی با توزیع ناشناخته هستند، یافتن جایگشت بهینه در چنین حالتی را بهینه‌سازی خطی تصادفی روی گروه جایگشت می‌نامیم. هدف اصلی این مقاله، ارائه راه حلی مبتنی بر آتاماتای یادگیر برای حل این مسایل مانند درخت پوشای کمینه در گراف تصادفی است [۸]. به‌طوری‌که نمونه‌گیری و جست‌وجو در فضای مسئله همزمان انجام می‌شود. به‌طور مثال برای برقراری ارتباط بین گره‌های حسگر در یک شبکه حسگر بی‌سیم که برخی در محدوده ارتباطی یکدیگر نیستند، اگر محل استقرار گره‌ها ثابت و فاصله بین آن‌ها مشخص باشد با مسئله درخت پوشای کمینه مواجه هستیم. چنانچه گره‌های حسگر پویا بوده و موقعیت آن‌ها براساس زمان تغییر کند [۹] آنگاه با مسئله درخت پوشای کمینه تصادفی مواجه هستیم..

**تعریف ۱:** فرض کنید مجموعه محدود $X$، جایگشت‌های $\{\sigma_۱,\sigma_۲,\ldots,\sigma_m\}$ از $X$ و یک تابع هزینه $f:X^۲\to R$ داده شده است. جایگشت $\sigma^*\in\{\sigma_۱,\sigma_۲,\ldots,\sigma_m\}$ را به گونه‌ای تعیین کنید که $\sum_{i\in X}f(i,\sigma^*(i))$ کمینه باشد. این مسئله به نام بهینه‌سازی خطی روی یک گروه جایگشت شناخته می‌شود [۱۰].

در متون، فرض بر این است که تابع هزینه $f$ تابعی قطعی یا تصادفی با توزیع شناخته‌شده است. اگر فرض کنیم $f$ تابعی تصادفی با توزیع احتمال ناشناخته است، در این صورت این مسئله به عنوان بهینه‌سازی خطی تصادفی روی یک گروه جایگشت شناخته می‌شود. بررسی‌های ما نشان می‌دهد که چنین مدلی از این مسئله تا کنون تعریف نشده است.

ادامه مقاله به این صورت سازماندهی شده است: در بخش ۲ به بررسی مفاهیم مرتبط با آتاماتای یادگیر پرداخته‌ایم. بخش ۳ الگوریتم پیشنهادی را برای حل مسئله بهینه‌سازی جایگشت مبتنی بر آتاماتای یادگیر توزیع‌شده توسعه‌یافته ارائه کرده است. بخش ۴ به بررسی همگرایی الگوریتم پیشنهادی اختصاص یافته و در بخش ۵ ابتدا مسئله یافتن درخت پوشای کمینه در گراف تصادفی به صورت یک مسئله بهینه‌سازی جایگشت تصادفی مشخص و سپس به کمک راه‌حل پیشنهادی حل شده است. در ضمن، کارآیی عملکرد الگوریتم پیشنهادی بررسی گردیده است.

## ۲- آتاماتای یادگیر

آتاماتای یادگیر تصادفی یک واحد تصمیم‌گیرنده تطبیقی است که فرایند یادگیری در آن، از طریق تعاملش با محیط صورت می‌گیرد. آتاماتای یادگیر، مجموعه‌ای از اقدام‌های قابل انجام دارد. این اقدام‌ها به تصادف و بر اساس یک بردار توزیع احتمال، انتخاب شده و به عنوان ورودی به محیط اعمال می‌شوند. محیط، اقدام انجام‌شده را به کمک یک سیگنال تقویتی بازخوردی، مورد ارزیابی قرار می‌دهد. آتاماتای یادگیر بر اساس سیگنال بازخوردی حاصل، بردار توزیع احتمال انتخاب اقدام‌ها را به‌روزرسانی می‌کند. هدف آتاماتا، پیداکردن اقدام بهینه در میان مجموعه اقدام‌های قابل انجام است؛ اقدامی که بیشترین پاداش را از محیط دریافت کند [۱۱] تا [۱۳].

آتاماتای یادگیر تصادفی در حوزه‌های مختلفی مانند پردازش تصویر[۱۴]، شبکه‌های کامپیوتری و شبکه‌های حسگر بی‌سیم [۱۵] تا [۲۰]، محاسبات ابری [۲۱] تا [۲۳]، شبکه‌های اجتماعی [۲۴]، مسائل بهینه‌سازی [۲۵] تا [۳۲] و خوشه‌بندی [۳۳] کاربرد دارد. از مزایای آتاماتای تصادفی می‌توان به موارد زیر اشاره کرد [۳۴]:

- آتاماتاهای یادگیر تصادفی در شرایطی که اطلاعات کافی در دسترس نیست، به خوبی عمل می‌کنند.
- آتاماتاهای یادگیر تصادفی در شرایطی که عدم قطعیت وجود دارد، دارای عملکرد خوبی هستند.
- آتاماتاهای یادگیر تصادفی، عمل جستجو را در فضای احتمال انجام می‌دهند.
- آتاماتاهای یادگیر تصادفی به دلیل ساختار ساده به راحتی قابل پیاده‌سازی هستند.
- آتاماتاهای یادگیر تصادفی به دلیل بار محاسباتی کم در محیط‌های بلادرنگ قابل استفاده هستند.

ارتباط آتاماتای تصادفی با محیط در شکل ۱ نشان داده شده است. از این مجموعه به همراه الگوریتم یادگیری تحت عنوان آتاماتای یادگیر تصادفی نام برده می‌شود. به این ترتیب آتاماتای یادگیر تصادفی را می‌توان با چهارتایی (۱) تعریف کرد

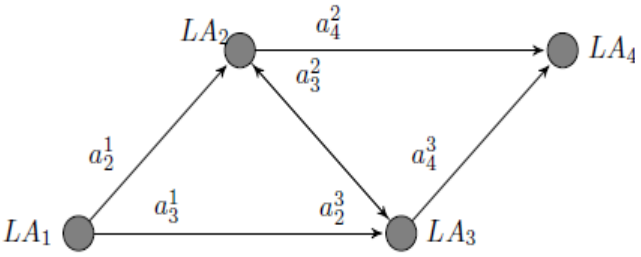
$$SLA\equiv\{\alpha,\beta,p,T,c\} \qquad (۱)$$

به طوری که:

مجموعه اقدام‌های آتاماتا/ مجموعه ورودی‌های محیط

$$\alpha\equiv\{\alpha_۱,\alpha_۲,\ldots,\alpha_r\} \qquad (۲)$$

مجموعه ورودی‌های آتاماتا/ مجموعه خروجی‌های محیط




شکل ۲: آتاماتای یادگیر توزیع‌شده.

$$\beta \equiv \{\beta_1, \beta_2, \ldots, \beta_m\} \quad (۳)$$

بردار احتمال اقدام‌های آتاماتا

$$p \equiv \{p_1, p_2, \ldots, p_r\} \quad (۴)$$

الگوریتم یادگیری

$$T \equiv \varphi \to \alpha \quad (۵)$$

مجموعه احتمالات جریمه که معرف محیط هستند

$$C \equiv \{c_1, c_2, \ldots, c_r\} \quad (۶)$$

الگوریتم یادگیری یک رابطه بازگشتی است که برای انجام تغییرات و به‌روزرسانی در بردار احتمال اقدام‌های آتاماتا در یک آتاماتای یادگیر تصادفی با ساختار متغیر مورد استفاده قرار می‌گیرد. با فرض این‌که آتاماتای یادگیر تصادفی با ساختار متغیر در زمان $k$ از میان مجموعه اقدام‌های $\alpha$ عمل $\alpha_i(k)$ را انتخاب می‌کند و با داشتن بردار احتمال انتخاب اقدام‌های آتاماتا یعنی $p(k)$، اگر $a$ و $b$ میزان افزایش یا کاهش احتمالات اقدام‌ها و $r$ تعداد اقدام‌های قابل انجام توسط آتاماتای یادگیر باشد آنگاه بردار $p(k)$ توسط الگوریتم یادگیری (۷) و (۸) به‌روزرسانی می‌شود. در ضمن مقدار $a$را پارامتر پاداش و $b$ را پارامتر جریمه می‌نامند.

$$p_j(k+1) = \begin{cases} (1-a)p_j(k)+a & , \quad j=i \\ (1-a)p_j(k) & , \quad \forall j \neq i \end{cases} \quad (۷)$$

$$p_j(k+1) = \begin{cases} (1-b)p_j(k) & , \quad j=i \\ (1-b)p_j(k)+\dfrac{b}{r-1} & , \quad \forall j \neq i \end{cases} \quad (۸)$$

رابطه (۷) زمانی مورد استفاده قرار می‌گیرد که عمل $\alpha_i(k)$ منجر به دریافت پاداش از محیط شده باشد و (۸) زمانی مورد استفاده قرار می‌گیرد که این عمل به دریافت جریمه از محیط منجر شده باشد. در (۷) و (۸) اگر $a=b$ باشد، روابط یادگیری خطی را الگوریتم $L_{R-P}$ می‌نامند. اگر $a=\varepsilon b$ باشد، آن را $L_{R-\varepsilon P}$ و اگر $b=0$ باشد، آن را $L_{R-I}$ می‌نامند.

## ۲-۱ آتاماتای یادگیر با مجموعه اقدام‌های متغیر

یک آتاماتای یادگیر با مجموعه اقدام متغیر، آتاماتایی است که در آن تعداد اقدام‌های موجود هر آتاماتا در طول زمان تغییر می‌کند [۳۵]. فرض کنید که مجموعه اقدام‌های آتاماتا را با $\underline{\alpha} \equiv \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ و مجموعه اقدام‌های قابل انجام توسط آتاماتا در زمان $n$ را با $A(k) \subseteq \alpha$ نشان دهیم

$$A(k) \in \{A_1, A_2, \ldots, A_m\} \ , \ m = 2^{n-1} \quad (۹)$$

انتخاب زیرمجموعه $A(k)$ از میان مجموعه اقدام‌های قابل انجام توسط آتاماتا، غالباً توسط شرایط بیرونی تحمیل می‌شود. می‌توان این گونه فرض کرد که این زیرمجموعه از اقدام‌های آتاماتا که از این به بعد مجموعه اقدام‌های فعال آتاماتا نامیده می‌شود، توسط یک عامل بیرونی و با توزیع احتمال

$$\psi_i(k) = \{\psi_1(k), \psi_2(k), \ldots, \psi_m(k)\} \quad (۱۰)$$

انتخاب می‌شود. اگر طبق (۱۱) احتمال انتخاب عمل $\alpha_i$ از میان مجموعه اقدام‌های فعال، به شرط آن‌که مجموعه اقدام‌های فعال آتاماتا برابر با $A(k)$ باشد آنگاه (۱۲) را خواهیم داشت

$$\hat{p}_i(k) = \text{Prob}[\alpha(k) = \alpha_i \,|\, A(k)] \ , \ \alpha_i \in A(k) \quad (۱۱)$$

$$\hat{p}_i(k) = \frac{p_i(k)}{K(k)} \quad (۱۲)$$

که در این رابطه، $K(k)$ مجموع احتمالات تمام اقدام‌های فعال (اقدام‌های عضو $A(k)$ ) است و داریم $K(k) = \sum_{\alpha_i \in A(k)} p_i(k)$ که در آن $p_i(k) = \text{Prob}[\alpha(k) = \alpha_i]$ احتمال انتخاب اقدام $\alpha_i$ در میان مجموعه تمام اقدام‌های آتاماتا (اعم از این که فعال باشد یا نباشد) است. بدین ترتیب نحوه انتخاب اقدام توسط آتاماتا یا مجموعه اقدام متغیر بدین صورت است که: فرض کنید آتاماتا مجموعه اقدام‌های فعال $A(k)$ را داشته باشد. ضریب نرمال‌کننده $K(k) = \sum_{\alpha_i \in A(k)} p_i(k)$ برای این مجموعه از اقدام‌های فعال محاسبه شده و بردار احتمال انتخاب اقدام‌های آتاماتا مطابق با رابطه بالا نرمال‌سازی می‌شود؛ به طوری که مجموع احتمال انتخاب اقدام‌های فعال همچنان ۱ باشد. پس از این، آتاماتا بر اساس بردار جدید $\hat{p}_i(k)$ یکی از اقدام‌های (فعال) خود را انتخاب کرده و به محیط اعمال می‌کند. در مرحله به‌روزرسانی نیز بردار $\hat{p}_i(k)$ مطابق با الگوریتم یادگیری مورد استفاده توسط آتاماتا به‌روزرسانی می‌شود و سپس با استفاده مجدد از رابطه بالا، بردار $p_i(k) = \hat{p}_i(k).K(k)$ به دست می‌آید. در صورتی که الگوریتم یادگیری مورد استفاده $L_{R-I}$ باشد، این روش ویژگی‌های $\varepsilon-$optimality و absolute expediency را دارد [۳۵].

## ۲-۲ آتاماتای یادگیر توزیع‌شده

آتاماتای یادگیر توزیع‌شده شبکه‌ای است از آتاماتاهای یادگیر که برای حل یک مسئله خاص با یکدیگر همکاری دارند. یک DLA در قالب یک گراف ارائه می‌شود و با چندتایی $(V, E, T, v_.)$ تعریف می‌گردد که در آن $V$ مجموعه رئوس گراف، $E$ مجموعه یال‌ها، $T$ مجموعه‌ای از الگوریتم‌های یادگیر مورد استفاده توسط آتاماتاها در DLA و $v_.$ گره ریشه DLA است [۳۶].

در این شبکه از آتاماتاهای همکار در هر زمان تنها یک آتاماتا فعال است. تعداد اعمال قابل انجام توسط یک آتاماتا در DLA برابر است با تعداد آتاماتاهایی که به این آتاماتا متصل شده‌اند. انتخاب یک عمل توسط آتاماتا در این شبکه، باعث فعال‌شدن آتاماتای متصل‌شده به این آتاماتا و متناظر با این عمل می‌گردد. به عبارت معادل، انتخاب یک عمل توسط یک آتاماتا در این شبکه متناظر با فعال‌شدن یک آتاماتای دیگر در این شبکه است.

وجود یال $(LA_i, LA_j)$ در این گراف به آن معنا می‌باشد که انتخاب عمل $\alpha_j^i$ توسط $LA_i$ باعث فعال‌شدن $LA_j$ می‌گردد (شکل ۲).

**Algorithm 1:** The DLA-based algorithm for solving Permutation Optimization

**Input:** $f$: objective function, $n$: the number of elemnts, Thresholds $P_{TH}, k$

**Output:** Permutation $X$

Construct a Complete Graph $G=(V,E)$ with $n+1$ node $v=\{0,1,\ldots,n\}$ and DLA from graph $G$
Let $k=0$ be the stage number
Let $X$ denotes the constructed permutation
Let $TH_k$ be the cost of the best solution up to stage $k$ and initially set to $\infty$
$X \leftarrow \phi$
**begin algorithm**
**repeat**
  Let $P, A$ denote Passive and Active sets of DLA
  $P \leftarrow V, A \leftarrow \phi$
  $u \leftarrow 0$
  $A \leftarrow A+u$
  $P \leftarrow P-u$
  **while** $|X|<n$ **do**
    $e=(u,v)=$ select_action $(F)$
    each automaton in $P$ prunes its action-set for cycle avoidance
    $X \leftarrow X+v$
    $A \leftarrow A+v, P \leftarrow P-v$
    $u \leftarrow v$
  **end while**
  **if** $|X|=n$ **then**
    **if** $f(X)<TH_k$ **then**
      reward DLA
    **else**
      penalize DLA
    **end if**
    $TH_k=$ update $(TH_k, f(X))$
  **else**
    Do nothing
  **end if**
  $k \leftarrow k+1$
  $P(X)=\prod_{i=1,u_i \in X}^{n-1} p_{u_{i+1}}^{u_i}$
  Enable all disabled actions
**until** $(P(X)>P_{POM}$ or $k>K)$
**end Algorithm 1**

شکل ۳: الگوریتم پیشنهادی برای حل مسئله بهینه‌سازی جایگشت.

تعداد اعمال قابل انتخاب توسط $LA_k$ برابر است با درجه خروجی این رأس. بردار احتمالات مربوط به عمل‌های قابل انجام توسط آتاماتای $LA_k$ به صورت $p^k=\{p_1^k, p_2^k, \ldots, p_{r_k}^k\}$ نمایش داده می‌شود. در این مجموعه عدد $p_m^k$ نشان‌دهنده احتمال مربوط به عمل $\alpha_m^k$ است. انتخاب عمل $\alpha_m^k$ توسط $LA_k$ باعث فعال‌شدن $LA_m$ می‌شود و $r_k$ تعداد اعمال قابل انجام توسط آتاماتای $LA_k$ را نشان می‌دهد.

## ۳- الگوریتم پیشنهادی

هر جایگشت از یک مجموعه $n$عضوی را می‌توان به یک مسیر هامیلتونی در یک گراف کامل نگاشت کرد. به این ترتیب برای حل مسئله بهینه‌سازی جایگشت روی یک مجموعه $n$عضوی به کمک DLA به این صورت عمل می‌کنیم:

الف) ابتدا گراف DLA ساخته می‌شود. این گراف شامل $n+1$ رأس است و یک رأس (با شماره ۰) به عنوان رأس ریشه در نظر گرفته می‌شود. برای هر یک از عناصر جایگشت، یک گره در شبکه DLA منظور می‌شود. هر جایگشت به یک مسیر هامیلتونی با شروع از گره ریشه نگاشت می‌شود و ترتیب گره‌ها در یک مسیر، ترتیب آنها در جایگشت را مشخص می‌کند. هر گره در DLA به عنوان یک آتاماتا به تعداد ورودی‌های مسئله مورد نظر اقدام قابل انجام دارد. از آنجا که در اغلب مسایل بهینه‌سازی جایگشت، تعداد ورودی‌ها با تعداد مکان‌ها یکسان است (در حقیقت یک جایگشت روی تمام ورودی تعریف می‌شود)، به این ترتیب یک گراف کامل با $n+1$ رأس به دست می‌آید که هر $LA$ نشان‌دهنده یکی از عناصر جایگشت است و به تعداد $n$ اقدام قابل انجام برای هر $LA$ وجود دارد.

ب) پس از ساختن گراف DLA، آتاماتاها در DLA در وضعیت Passive قرار می‌گیرند.

ج) آتاماتای شماره صفر- ریشه- فعال شده و بر اساس بردار احتمال انتخاب اقدام‌ها، اقدامی را به تصادف از میان مجموعه اقدام‌های خود انتخاب می‌کند. انتخاب این اقدام باعث فعال‌شدن آتاماتای متناظر با سمت دیگر یال می‌شود.

د) به منظور جلوگیری از بروز حلقه در مسیر از آتاماتاهای با تعداد اقدام‌های متغیر استفاده شده است. با انتخاب یک یال- انجام اقدام متناظر با یال توسط یک آتاماتا- این یال از مجموعه یال‌های قابل انتخاب سایر گره‌های Passive حذف می‌شود.

ه) تا زمان Activeشدن کلیه آتاماتاها و تشکیل یک مسیر هامیلتونی- به دست آمدن یک جایگشت- فرایند انجام اقدام توسط یک آتاماتا و فعال‌کردن یک آتاماتای دیگر ادامه پیدا می‌کند.

و) فرایند یافتن جایگشت تا زمان تحقق شرایط پایانی ادامه پیدا خواهد کرد.

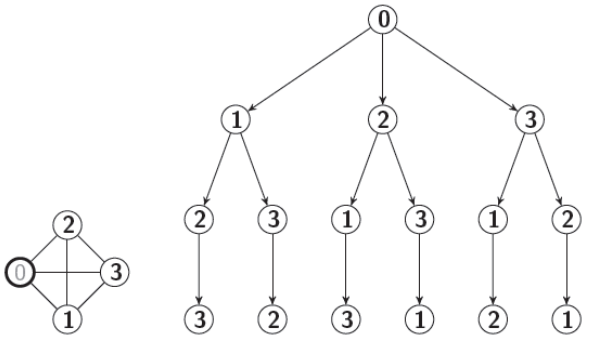
شبه‌کد الگوریتم پیشنهادی برای حل مسئله بهینه‌سازی جایگشت به کمک eDLA در شکل ۳ آورده شده است. الگوریتم مورد استفاده برای به‌روزرسانی بردار احتمال انتخاب اقدام‌های آتاماتا، الگوریتم $L_{R-I}$ است. به منظور ساخت یک جایگشت قابل قبول، از آتاماتاهای یادگیر با تعداد اقدام متغیر استفاده شده است.

برای هر جایگشت می‌توان یک احتمال انتخاب آن جایگشت توسط DLA تعریف کرد. فرض کنید جایگشت $\sigma=(\sigma(1),\ldots,\sigma(n))$ توسط DLA تعریف شده و نیز فرض کنید بردار $P^k=(p_1^k, p_2^k, \ldots, p_n^k)$، بردار احتمال انتخاب اقدام‌های آتاماتای $k$ام (متناظر با عنصر $k$ام) باشد. در این صورت احتمال انتخاب جایگشت $\sigma$ توسط (۱۳) تعریف می‌شود

$$q(\sigma)=\prod_{i=1}^{n} p_{\sigma(i)}^{i-1} \qquad (۱۳)$$

## ٤- بررسی همگرایی الگوریتم پیشنهادی

برای مباحث این بخش نمادگذاری‌های زیر را در نظر می‌گیریم. فرض کنید که تعداد ورودی‌های مسئله $n$ و تعداد تمام جایگشت‌های ممکن مسئله $r \le n!$ باشد. نیز فرض کنید در $k$امین تکرار الگوریتم، جایگشت $\sigma_i$ که یکی از جواب‌ها است، به دست آمده است. احتمال جایگشت $\sigma_i$ را مطابق با (۱۳) تعریف می‌کنیم. بردار احتمال انتخاب جایگشت‌ها را در $k$امین تکرار الگوریتم با $q(k)$ نمایش می‌دهیم. علاوه بر این برای سهولت در نوشتار از نمایش جایگشت به شکل یک مسیر متشکل از تعدادی یال در گراف DLA استفاده می‌کنیم.



شکل ۴: درخت جایگشت‌های روی سه عنصر با شماره‌های ۱، ۲ و ۳.

جدول ۱: نحوه تعریف احتمال انتخاب هر جایگشت برای جایگشت روی سه عنصر.

| جایگشت | احتمال |
|---|---|
| $\sigma_1=(1,2,3)$ | $q_1(k)=p_1(k)p_1(k)$ |
| $\sigma_2=(1,3,2)$ | $q_2(k)=p_1(k)p_2(k)$ |
| $\sigma_3=(2,1,3)$ | $q_3(k)=p_2(k)p_1(k)$ |
| $\sigma_4=(2,3,1)$ | $q_4(k)=p_2(k)p_3(k)$ |
| $\sigma_5=(3,1,2)$ | $q_5(k)=p_3(k)p_1(k)$ |
| $\sigma_6=(3,2,1)$ | $q_6(k)=p_3(k)p_2(k)$ |

**لم ۱:** اگر $d_i(k)$ و $c_i(k)$ احتمالات مربوط به جریمه و تنبیه‌شدن جایگشت $\sigma_i$ حاصل‌شده در $k$امین تکرار را نشان دهند و $q(k)$ بر اساس الگوریتم ارائه‌شده در شکل ۳ تغییر کند، در این صورت امید ریاضی شرطی $E[q_i(k+1)-q_i(k)|q(k)]$ طبق (۱۴) تعریف می‌شود

$$E[q_i(k+1)-q_i(k)|q(k)] \geq aq_i(k)\sum_{\substack{j=1\\ j\neq i}}^{r} q_j(k)[d_i(k)-d_j(k)] \qquad (14)$$

**اثبات:** برای اثبات یک جایگشت روی سه عنصر را در نظر گرفته و محاسبات را انجام می‌دهیم (شکل ۴). این اثبات قابل تعمیم است.

احتمال منتسب به هر یک از ۶ جایگشت در جدول ۱ آورده شده است. ضمناً فرض می‌کنیم مطابق آنچه که در [۳۷] پیشنهاد گردیده است، نرخ یادگیری در هر یک از سطوح مطابق با (۱۵) تغییر می‌کند

$$a_{\pi_{i,j}}(k)=\frac{a_{\pi_{i,j-1}}(k)}{p^{\pi_{i,j}}_{\pi_{i,j-1}}(k+1)} \qquad (15)$$

که $a_{\pi_{i,j}}(k)$ مقدار نرخ یادگیری در زمان $k$ را برای $j$امین آتاماتای واقع در مسیر $\pi_i$ نشان می‌دهد [۳۸]. ما در ادامه، محاسبات را برای جایگشت $\sigma_4$ انجام خواهیم داد. منظور از $\downarrow$ و $\uparrow$ در (۱۶) به ترتیب کاهش و افزایش در مقدار احتمال یک اقدام بر اساس روش $L_{R-I}$ است اما

$$p_2 \downarrow p_3 = p_2(1-a)p_3 = q_4(1-a) \qquad (17)$$

و

$$p_2 \uparrow p_3 \downarrow = \{p_2(1-a)+a\}\{p_3(1-\frac{a}{p_2(1-a)+a})\} = \{p_2(1-a)+a\}p_3 - ap_3 = p_2p_3(1-a) = q_4(1-a) \qquad (18)$$

و در ادامه (۱۹) را خواهیم داشت. با جایگذاری مقادیر در (۱۶)، (۲۰) به دست می‌آید و با ساده‌سازی جبری (۲۰)، (۲۱) حاصل می‌شود. در نتیجه

$$E[q_4(k+1)|q(k)]-q_4(k) = aq_4(k)\sum_{\substack{j=1\\ j\neq 4}}^{6} q_j(k)\{d_4(k)-d_j(k)\} \qquad (22)$$

و حکم ثابت است. ■

**لم ۲:** با فرض این که جایگشت $\sigma_i$ جواب مسئله یافتن جایگشت بهینه باشد، در این صورت $\Delta q_i(k+1)=E[q_i(k+1)-q_i(k)|q(k)]$ همواره مثبت است.

**اثبات:** الگوریتم ۱ در **Error! Reference source not found.** همواره $n-1$ یال را از میان یال‌های گراف DLA انتخاب می‌کند تا یکی از $r$ جواب ممکن را بسازد. متوسط مقدار $q_i(k)$ برای جایگشت $\sigma_i$ طبق (۱) تعریف می‌شود به‌طوری‌که برابراست با ضرب مقادیر احتمال اقدام‌های متناظر با یال‌های تشکیل‌دهنده جایگشت $\sigma_i$.

$$\Delta q_i(k+1)=E[q_i(k+1)-q_i(k)|q(k)] \geq aq_i(k)\sum_{j=1,j\neq i}^{6} q_j(k)\{d_i(k)-d_j(k)\} \qquad (23)$$

$$E[q_4(k+1)|q(k)]=\sum_{\beta=0}^{1}\sum_{j=1}^{6}E[q_4(k+1)|q(k),\alpha_j,\beta]\times P[\beta|\alpha_j]\times P[\alpha_j|q(k)] = q_1(k)\{c_1(k)q_4(k)+d_1(k)p_2 \downarrow p_3\}$$
$$+q_2(k)\{c_2(k)q_4(k)+d_2(k)p_2 \downarrow p_3\}+q_3(k)\{c_3(k)q_4(k)+d_3(k)p_2 \uparrow p_3 \downarrow\}+q_4(k)\{c_4(k)q_4(k)+d_4(k)p_2 \uparrow p_3 \uparrow\} \qquad (16)$$
$$+q_5(k)\{c_5(k)q_4(k)+d_5(k)p_2 \downarrow p_3\}+q_6(k)\{c_6(k)q_4(k)+d_6(k)p_2 \downarrow p_3\}$$

$$p_2 \uparrow p_3 \uparrow = \{p_2(1-a)+a\}\{p_3(1-\frac{a}{p_2(1-a)+a})+\frac{a}{p_2(1-a)+a}\} = \{p_2(1-a)+a\}p_3 - ap_3 + a = p_2p_3(1-a)+a = q_4(1-a)+a \qquad (19)$$

$$E[q_4(k+1)|q(k)] = q_1(k)\{c_1(k)q_4(k)+d_1(k)q_4(k)(1-a)\}+q_2(k)\{c_2(k)q_4(k)+d_2(k)q_4(k)(1-a)\}$$
$$+q_3(k)\{c_3(k)q_4(k)+d_3(k)q_4(k)(1-a)\}+q_4(k)\{c_4(k)q_4(k)+d_4(k)\{q_4(k)(1-a)+a\}\} \qquad (20)$$
$$+q_5(k)\{c_5(k)q_4(k)+d_5(k)q_4(k)(1-a)\}+q_6(k)\{c_6(k)q_4(k)+d_6(k)q_4(k)(1-a)\}$$

$$E[q_4(k+1)|q(k)] = q_1(k)\{q_4(k)-ad_1(k)q_4(k)\}+q_2(k)\{q_4(k)-ad_2(k)q_4(k)\}+q_3(k)\{q_4(k)-ad_3(k)q_4(k)\}$$
$$+q_4(k)\{q_4(k)-ad_4(k)q_4(k)+ad_4(k)\}+q_5(k)\{q_4(k)-ad_5(k)q_4(k)\}+q_6(k)\{q_4(k)-ad_6(k)q_4(k)\} \qquad (21)$$
$$= q_4(k)-\{aq_4(k)\sum_{\substack{j=1\\ j\neq 4}}^{6}q_j(k)d_j(k)\}+ad_4(k)q_4(k)\{1-q_4(k)\} = q_4(k)+aq_4(k)\sum_{\substack{j=1\\ j\neq 4}}^{6}q_j(k)\{d_4(k)-d_j(k)\}$$

چون جایگشت $\sigma_i$ بهینه است، با توجه به قانون قوی اعداد بزرگ برای مقادیر بزرگ $k$، مقدار $\{d_i(k)-d_j(k)\}$ برای هر $j\neq i$ مثبت است و در نتیجه سمت راست (۲۳) مثبت و بنابراین برای مقادیر بزرگ $k$، طرف راست (۲۳) مقداری مثبت می‌باشد و در نتیجه $\Delta q_i(k+1)\geq 0$ ■

**گزاره ۱:** $\lim_{k\to\infty} q_i(k)=q_i^*\in\{0,1\}$ با احتمال ۱ وجود دارد.

لم ۲ بیان‌کننده آن است که $\{q_i(k)\}$ یک زیرمارتینگل است. با استفاده از قضایای حدی مارتینگل نتیجه می‌شود که $\lim_{k\to\infty} q_i(k)=q_i^*$ با احتمال ۱ وجود دارد. علاوه بر این اگر فرض کنیم $q_i(k)\neq 0$ و $q_i(k)\neq 1$، در این صورت با یک احتمال ناصفر به ازای تمام مقادیر $k$ داریم $q_i(k)=q_i(k+1)$ و بنابراین الزاماً $q_i^*\in\{0,1\}$ ■

گزاره ۱ نشان می‌دهد که احتمال انتخاب جایگشت بهینه یکی از دو مقدار ۰ یا ۱ است. به عبارت دیگر، الگوریتم پیشنهادی همواره همگرا می‌شود اما احتمال این که به جوابی غیر از جواب بهینه همگرا شود وجود دارد. در این بخش نشان خواهیم داد که تحت شرایطی و با انتخاب پارامترهای مناسب برای الگوریتم یادگیری، می‌توان احتمال همگرایی الگوریتم پیشنهادی به جواب بهینه را به اندازه کافی به ۱ نزدیک کرد. برای این کار از روش پیشنهادی [۳۹] که در موارد مشابه نیز مورد استفاده قرار گرفته است [۳۸] و [۴۰]، استفاده می‌کنیم.

فرض کنید $V_r=\{e_1,e_2,\ldots,e_r\}$ مجموعه حالت‌های جاذب فرایند $\{q(k)\}$ باشد و $q^*\in V_r$ حالتی را نشان دهد که $q(k)$ به آن همگرا می‌شود. تعریف می‌کنیم

$$\Gamma_i[q]=\Pr[q^*=e_i\,|\,q(0)=q] \qquad (۲۴)$$

که $\Gamma_i[q]$ احتمال همگرایی الگوریتم به جایگشت $\sigma_i$ با شروع از بردار احتمال $q$ را نشان می‌دهد. فرض کنید

$$S_r=\{q\,|\,q_j\geq 0,\ \textstyle\sum_{j=1}^{r} q_j=1, j=1,2,\ldots,r\} \qquad (۲۵)$$

فرضاً $C(S_r): S_r\to R$ فضای حالت تمام توابع مشتق‌پذیر پیوسته حقیقی مقدار با مشتق محدود تعریف‌شده روی $S_r$ باشد که $R$ یک خط حقیقی است. اگر $g[q]\in C(S_r)$ باشد، الگوریتم پیشنهادی در شکل ۳ نظیر یک عملگر $U$ با میانگین $Ug[q]=E[g[q(k+1)]\,|\,q(k)=q]$ عمل می‌کند که در این رابطه $E[.]$ نشان‌دهنده امید ریاضی است. نشان داده شده [۳۹] و [۴۰] که $U$ عملگری خطی است و توابع نامنفی را نامنفی نگه می‌دارد. یعنی

$$\text{if } g[q]\geq 0\Rightarrow Ug[q]\geq 0\ ,\ \forall q\in S_r \qquad (۲۶)$$

نسبت به عملگر $U$ می‌توان دو گروه از توابع موسوم به زیرمنظم و ابرمنظم را تعریف کرد. اگر $Ug[q]\geq g[q]$ باشد، تابع $g[q]$ را زیرمنظم و اگر $Ug[q]\leq g[q]$ باشد، تابع $g[q]$ را ابرمنظم گویند.

نشان داده شده [۳۹] که $\Gamma_i[q]$ تنها جواب پیوسته $U\Gamma_i[q]=\Gamma_i[q]$ با شرایط مرزی زیر است

$$\Gamma_i[e_j]=\begin{cases}1 & , \ i=j\\ 0 & , \ i\neq j\end{cases} \qquad (۲۷)$$

اما حل معادله تابعی $U\Gamma_i[q]=\Gamma_i[q]$ برای تعیین احتمال انتخاب جایگشت بهینه بر حسب احتمال اولیه توسط الگوریتم پیشنهادی، کار سختی است. برای این کار از یک کران پایین برای این تابع استفاده می‌کنیم. از گزاره قبلی می‌دانیم که احتمال انتخاب جایگشت بهینه $q_i^*\in\{0,1\}$ است. تعریف می‌کنیم

$$\Phi_i[x,q]=\frac{\exp\frac{-xq_i}{a}-1}{\exp\frac{-x}{a}-1} \qquad (۲۸)$$

که در آن $x>0$ بایستی انتخاب شود. واضح است که تابع تعریف‌شده در (۲۸) شرایط مرزی تعریف‌شده در (۲۷) را ارضا می‌کند. نشان خواهیم داد که $\Phi_i[x,q]$ یک تابع زیرمنظم بوده و لذا قادر است که یک کران پایین برای $\Gamma_i[q]$ تعریف کند. می‌توان نشان داد که $\Phi_i[x,q]$ وقتی زیرمنظم است که $\theta_i[x,q]=\exp(-xq_i/a)$ ابرمنظم باشد ($\exp(-x/a)-1$ همواره منفی است).

**لم ۳:** اگر $\sigma_i$ جایگشت بهینه $\theta_i[x,q]=\exp(-xq_i/a)$ باشد که در آن $q_i$ احتمال انتخاب جایگشت $\sigma_i$ و $a$ نرخ یادگیری مورد استفاده در الگوریتم است، در این صورت $x$ وجود دارد که $\theta_i[x,q]$ ابرمنظم خواهد بود.

**اثبات:** برای تعیین شرایطی که تحت آن شرایط، تابع $\theta_i[x,q]$ یک تابع ابرمنظم است، از تعریف ابرمنظم و محاسبه مقدار امید ریاضی استفاده می‌کنیم. فرض کنید جایگشت $\sigma_i$ جواب بهینه است، در این صورت

$$U\theta_i[x,q]=E[\exp\frac{-xq_i(k+1)}{a}\,|\,q(k)=q] \qquad (۲۹)$$

برای محاسبه مقدار امید ریاضی در (۲۹) به این ترتیب عمل می‌کنیم

$$\begin{aligned}&E[\exp\frac{-xq_i(k+1)}{a}\,|\,q(k)=q]\\ &=\sum_{j=1}^{r} E[\exp\frac{-xq_i(k+1)}{a}\,|\,q(k)=q,\sigma_j]q_j\\ &=\sum_{j=1}^{r}\exp\frac{-xq_i(k+1)}{a}\times q_j\times d_j\\ &+\sum_{j=1}^{r}\exp\frac{-xq_i(k)}{a}\times q_j\times(1-d_j)\end{aligned} \qquad (۳۰)$$

با ساده‌سازی (۳۰) به (۳۱) و (۳۲) می‌رسیم

$$\begin{aligned}&\sum_{j=1}^{r}\exp(\frac{-xq_i(k)}{a})\times q_j\times(1-d_j)\\ &=\theta_i[x,q]\sum_{j=1}^{r} q_j\times(1-d_j)\\ &=\theta_i[x,q]-\theta_i[x,q]\sum_{j=1}^{r} q_j\times d_j\end{aligned} \qquad (۳۲)$$

در (۳۱) داریم

$$\begin{aligned}\sum_{j=1}^{r}\exp(\frac{-xq_i(k+1)}{a})q_jd_j &= q_id_i\exp\frac{-x\prod_{e(m,n)\in\sigma_i}\{p_n^m+a(1-p_n^m)\}}{a}\\ &+\sum_{\substack{j=1\\ j\neq i}}^{r} q_jd_j\exp\frac{-x\prod_{e(m,n)\in\sigma_i\cap\sigma_j}\{p_n^m+a(1-p_n^m)\}\prod_{e(m,n)\in\sigma_i-\sigma_j}\{p_n^m(1-a)\}}{a}\end{aligned} \qquad (۳۱)$$

$$\prod_{e(m,n)\in\sigma_i-\sigma_j}\{p_n^m(1-a)\}$$
$$=\frac{\prod_{e(m,n)\in\sigma_i-\sigma_j}\{p_n^m(1-a)\}\prod_{e(m,n)\in\sigma_i\cap\sigma_j}\{p_n^m(1-a)\}}{\prod_{e(m,n)\in\sigma_i\cap\sigma_j}\{p_n^m(1-a)\}} \quad (۳۳)$$
$$=\frac{\prod_{e(m,n)\in\sigma_i}\{p_n^m(1-a)\}}{\prod_{e(m,n)\in\sigma_i\cap\sigma_j}\{p_n^m(1-a)\}}$$

با توجه به این که $q_i=\prod_{e(m,n)\in\sigma_i}p_n^m$ و با توجه به (۲۹) تا (۳۳) خواهیم داشت

$$U\theta_i[x,q]=q_id_i\times\exp\{-\frac{x}{a}(q_i+a(1-q_i))\}$$
$$+\sum_{\substack{j=1\\j\neq i}}^{r}q_jd_j\exp\{-\frac{xq_i(1-a)}{a}D(p_n^m)\} \quad (۳۴)$$
$$+\theta_i[x,q]-\theta_i[x,q]\sum_{j=1}^{r}q_j\times d_j$$

اما روشن است که

$$D(p_n^m)=\frac{\prod_{e(m,n)\in\sigma_i\cap\sigma_j}\{p_n^m+a(1-p_n^m)\}}{\prod_{e(m,n)\in\sigma_i\cap\sigma_j}\{p_n^m(1-a)\}}\geq 1 \quad (۳۵)$$

با توجه به نزولی‌بودن تابع $e^{-x}$ داریم

$$\exp\{-\frac{x}{a}q_i(1-a)\frac{\prod_{e(m,n)\in\sigma_i\cap\sigma_j}\{p_n^m+a(1-p_n^m)\}}{\prod_{e(m,n)\in\sigma_i\cap\sigma_j}\{p_n^m(1-a)\}}\} \quad (۳۶)$$
$$\leq\exp\{-\frac{x}{a}q_i(1-a)\}$$

از (۳۵) و (۳۶) به (۳۷) می‌رسیم

$$U\theta_i[x,q]-\theta_i[x,q]\leq q_id_i\times\exp\{-\frac{x}{a}(q_i+a(1-q_i))\}$$
$$+\sum_{\substack{j=1\\j\neq i}}^{r}q_jd_j\times\exp\{-\frac{x}{a}q_i(1-a)\}-\theta_i[x,q]\sum_{j=1}^{r}q_jd_j \quad (۳۷)$$

اما

$$\theta_i[x,q]\sum_{j=1}^{r}q_jd_j=q_id_i\theta_i[x,q]+\sum_{\substack{j=1\\j\neq i}}^{r}q_jd_j\theta_i[x,q] \quad (۳۸)$$

بنابراین (۳۹) حاصل می‌شود. تعریف می‌کنیم

$$U\theta_i[x,q]-\theta_i[x,q]\leq q_id_i(\exp\{-\frac{x}{a}(q_i+a(1-q_i))\}-\theta_i[x,q])+\sum_{\substack{j=1\\j\neq i}}^{r}q_jd_j(\exp\{-\frac{x}{a}q_i(1-a)\}-\theta_i[x,q])$$
$$=\theta_i[x,q](q_id_i\exp\{-x(1-q_i)\}-1+\sum_{\substack{j=1\\j\neq i}}^{r}q_jd_j(\exp\{xq_i\}-1)) \quad (۳۹)$$
$$=-xq_i\theta_i[x,q]((1-q_i)d_i\frac{\exp\{-x(1-q_i)\}-1}{-x(1-q_i)}-\sum_{\substack{j=1\\j\neq i}}^{r}q_j\times d_j\frac{\exp\{xq_i\}-1}{xq_i})$$

$$V[x]=\begin{cases}\frac{\exp(x)-1}{x} & , \quad x\neq 0\\ 1 & , \quad x=0\end{cases} \quad (۴۰)$$

لذا

$$U\theta_i[x,q]-\theta_i[x,q]\leq$$
$$-xq_i\theta_i[x,q]((1-q_i)d_iV[-x(1-q_i)]-\sum_{\substack{j=1\\j\neq i}}^{r}q_jd_jV[xq_i]) \quad (۴۱)$$

$(1-q_i)d_iV[-x(1-q_i)]-\sum_{\substack{j=1\\j\neq i}}^{r}q_jd_jV[xq_i]\geq 0$ شرط لازم برای ابرمنظم‌بودن $\theta_i[x,q]$ است. لذا

$$(1-q_i)d_iV[-x(1-q_i)]\geq\sum_{\substack{j=1\\j\neq i}}^{r}q_jd_jV[xq_i] \quad (۴۲)$$

یا

$$\frac{V[-x(1-q_i)]}{V[xq_i]}\geq\frac{\sum_{\substack{j=1\\j\neq i}}^{r}q_jd_j}{(1-q_i)d_i} \quad (۴۳)$$

در (۴۳) داریم

$$\frac{\sum_{\substack{j=1\\j\neq i}}^{r}q_j\times d_j}{(1-q_i)d_i}=\frac{\sum_{\substack{j=1\\j\neq i}}^{r}q_j\times d_j}{\sum_{\substack{j=1\\j\neq i}}^{r}q_j\times d_i}=\frac{\sum_{\substack{j=1\\j\neq i}}^{r}q_j\times\frac{d_j}{d_i}}{\sum_{\substack{j=1\\j\neq i}}^{r}q_j} \quad (۴۴)$$

لذا

$$\min_{j\neq i}\frac{d_j}{d_i}\leq\frac{\sum_{\substack{j=1\\j\neq i}}^{r}q_j\times\frac{d_j}{d_i}}{\sum_{\substack{j=1\\j\neq i}}^{r}q_j}\leq\max_{j\neq i}\frac{d_j}{d_i} \quad (۴۵)$$

با جایگذاری (۴۳) در (۴۵)، شرط ابرمنظم‌بودن به (۴۶) تقلیل می‌یابد

$$\frac{V[-x(1-q_i)]}{V[xq_i]}\geq\max_{j\neq i}\frac{d_j}{d_i} \quad (۴۶)$$

تعریف می‌کنیم $H[x]=\ln V[x]$. با توجه به محدب‌بودن $H[x]$ می‌توان نشان داد [۳۹]

$$\frac{1}{V[x]}\leq\frac{V[-x(1-q_i)]}{V[xq_i]}\leq V[-x] \quad (۴۷)$$

از آنجا که $1/V[x]$ تابعی پیوسته و یکنوا کاهشی با $V[0]=1$ است، مقدار $x=x^*$ وجود دارد که برای تمام مقادیر در بازه $x\in(0,x^*]$ داریم

$\max_{j\neq i}(d_j/d_i)=1/V[x^*]\le 1/V[x]$. با انتخاب $x=x^*$ نابرابری (۴۲) ارضا می‌شود که در این صورت $\theta_i[x,q]$ ابرمنظم خواهد بود. ■

**قضیه ۱:** فرض کنید $q_i(k)$ احتمال متناظر با جایگشت بهینه $\sigma_i$ را در مرحله $k$ام نشان دهد. اگر بردار $q(k)=[q_i(k)]_{i=1,\dots,r}$ مطابق با الگوریتم شکل ۳ تغییر کند، در این صورت $\lim_{n\to\infty} q_i(n)\overset{a.s.}{=}1$.

**اثبات:** با توجه به لم ۳ و تعریف تابع $\Phi_i[x,q]$، مشاهده می‌شود که به ازای هر $\varepsilon>0$ مفروض، ثابت مثبت $a^*<1$ به گونه‌ای موجود است که نابرابری $\Gamma_i[q]\le\Phi_i[x,q]\le 1-\varepsilon$ برقرار می‌باشد. این به آن معنا است که $\lim_{n\to\infty} q_i(n)\overset{a.s.}{=}1$ و به این ترتیب اثبات قضیه ۱ کامل می‌شود. ■

## ۵- شبیه‌سازی و نتایج عددی

همچنان که در مقدمه گفتیم، مسایل زیادی وجود دارند که از طریق بهینه‌سازی جایگشت (قطعی یا تصادفی) می‌توان آنها را حل کرد. در این قسمت به بررسی حل مسئله درخت پوشای کمینه تصادفی از طریق بهینه‌سازی جایگشت تصادفی پرداخته‌ایم.

### ۵-۱ حل مسئله درخت پوشای کمینه تصادفی و حل آن از طریق بهینه‌سازی خطی تصادفی روی گروه جایگشت به کمک DLA

فرض کنید $G$ یک گراف وزن‌دار فاقد جهت است. در این گراف، زیرگراف $T=(V,E')$ درخت پوشای $G$ نامیده می‌شود، اگر:

الف) $T$ یک زیرگراف متصل از $G$ و شامل تمام رئوس $G$ است.

ب) مجموعه یال‌های زیرگراف $T$ زیرمجموعه‌ای از یال‌های گراف $G$ بوده و تعداد آنها $|E'|=|V|-1=n-1$ است.

درخت پوشای کمینه، درختی پوشا با کمترین وزن ممکن در میان مجموعه تمام درخت‌های پوشای گراف $G$ است.

یافتن درخت پوشای کمینه، یکی از مسایل بهینه‌سازی شبکه است که نخستین بار توسط بروفکا مطرح و حل شد [۴۱]. الگوریتم‌های متعددی برای حل این مسئله روی گراف‌های قطعی ارائه گردیده [۴۲] تا [۴۴] و علاوه بر آن، تا کنون روش‌های متفاوتی شامل راه حل‌های تقریبی، موازی و توزیع‌شده نیز برای حل آن ارائه گردیده است [۴۵] تا [۵۰]. این مسئله به غیر از کاربرد در بهینه‌سازی انواع مختلف شبکه‌ها نظیر شبکه‌های آب، برق، تلفن و نظایر آن، دارای کاربردهای متنوعی در مسایل گوناگون نظیر طراحی مدارهای الکترونیکی [۵۱]، خوشه‌بندی [۵۲] تا [۵۴] و بسیاری از حوزه‌های دیگر است [۵۵] تا [۵۸].

نسخه قطعی این مسئله (یعنی حالتی که یال‌های گراف دارای وزن قطعی هستند)، راه حل‌های متفاوتی با زمان چندجمله‌ای دارد. یکی از این راه حل‌ها، به راه حل کروسکال موسوم است [۴۳]. در این راه حل، به شکل حریصانه، یالی با کمترین وزن در هر مرحله به جواب اضافه می‌شود، به نحوی که موجب بروز پیدایش حلقه در زیرگراف نشود. این کار تا رسیدن تعداد یال‌ها به اندازه یک واحد کمتر از تعداد گره‌های گراف ادامه پیدا می‌کند.

حل مسئله یافتن MST در گرافی که وزن یال‌ها به شکل تصادفی- متغیر با زمان- تغییر می‌کند، به طور فزاینده‌ای سخت می‌شود. این سختی در یافتن راه حل، زمانی که تابع توزیع احتمال وزن یال‌ها از قبل شناخته‌شده نباشد، سخت‌تر نیز خواهد شد.

برای حل مسئله SMST تا کنون راه حل‌های مختلفی ارائه گردیده است. در [۸] با فرض شناخته‌شده بودن تابع توزیع احتمال وزن یال‌ها، حل مسئله به حل یک مسئله هم‌ارز در حالت قطعی تقلیل داده شده و از طریق حل این مسئله هم‌ارز به کمک الگوریتمی با زمان چندجمله‌ای، مسئله، حل گردیده است. نویسندگان در [۵۹] نسخه تصادفی مسئله یافتن درخت پوشای گلوگاهی را ارائه کرده‌اند که در آن وزن یال‌ها، متغیرهایی تصادفی هستند و نشان داده‌اند که تحت محدودیت‌هایی منطقی می‌توان مسئله را به معادل قطعی آن کاهش داد. مرجع [۶۰] روشی را برای حل مسئله درخت پوشای کمینه تصادفی ارائه داده است. این روش که به روش حذف بازه موسوم است، بهبودهای متعددی را روی الگوریتم [۸] ارائه و نشان داده که بهبودهای ارائه‌شده، جواب‌های بهتری را در زمان کمتر نسبت به روش [۸] به دست می‌آورند.

مشکل اکثر راه حل‌های ارائه‌شده تا اینجا، آن است که با فرض مشخص‌بودن توزیع احتمالی وزن یال‌ها، مبادرت به حل مسئله می‌کنند.

در [۶۱] نویسنده روشی برای حل مسئله یافتن درخت پوشای کمینه در گراف‌های تصادفی و با فرض نامشخص‌بودن توزیع احتمالی وزن یال‌ها، ارائه داده است. این راه حل مبتنی بر استفاده از آتاماتای یادگیر است. در روش پیشنهادی، در هر تکرار، آتاماتای یادگیر، یالی از گراف را برای نمونه‌گیری نامزد کرده و بر اساس پاره‌ای تجزیه و تحلیل‌های آماری مشخص می‌شود که آیا نمونه‌گیری از آن یال ضرورتی دارد یا خیر.

در [۶۲] نویسنده، الگوریتمی مبتنی بر شبکه‌ای از آتاماتاهای یادگیر برای یافتن درخت پوشای کمینه تصادفی ارائه داده است. در این شبکه در هر مرحله، تصادفاً آتاماتایی انتخاب شده و یالی را به مجموعه یال‌های درخت پوشای کمینه اضافه می‌کند. در بخش بررسی تجربی الگوریتم‌های پیشنهادی روش مورد استفاده را بیشتر تشریح خواهیم کرد.

در [۶۳] نویسنده، دو مسئله بهینه‌سازی روی شبکه های تصادفی را بررسی کرده‌است شامل یافتن کوتاهترین مسیر و یافتن درخت پوشای کمینه. در این مسئله، فرضی برای شبکه مطرح شده که کاربرد آن را در شبکه‌های کامپیوتری مناسب می‌کند. این فرض عبارت است از: مشاهده ناپذیر بودن مستقیم لینک‌ها. با برقراری این فرض در مورد هریک از این دو مسئله، تنها مقدار تجمیعی وزن یال‌ها برای تصمیم‌گیری موجود است. این مقاله هر یک از این مسائل را به عنوان یک مسئله MAB کلاسیک بیان کرده و راه‌حلی برای آن‌ها ارائه داده است.

در [۶۴] نویسنده ضمن معرفی eDLA برای حل مسئله یافتن زیرگراف بهینه در گراف‌های تصادفی از طریق نمونه‌گیری، eDLA را برای حل مسئله درخت پوشای کمینه تصادفی به کار گرفته است. راه حل ارائه‌شده در این مقاله نیز با این فرض آمده که وزن یال‌های نمونه‌گیری شده به صورت مستقیم قابل مشاهده نیستند و تنها مقدار وزن درخت پوشا، پس از انتخاب قابل مشاهده است.

ما در این قسمت، مسئله یافتن درخت پوشای کمینه را به عنوان یک مسئله بهینه‌سازی خطی تصادفی روی گروه جایگشت به کمک آتاماتای یادگیر حل می‌کنیم.

### ۵-۲ بیان فرمال مسئله درخت پوشای کمینه تصادفی به صورت یک مسئله بهینه‌سازی جایگشت تصادفی

گراف تصادفی فاقد جهت $G=(V,E,Q)$ داده شده که در آن $V$ مجموعه رئوس گراف، $E$ مجموعه یال‌های گراف و $Q$ تابع توزیع احتمال وزن یال‌ها است؛ به گونه‌ای که $Q=[q_{i,j}]_{n\times n}$ و $q_{i,j}$ تابع توزیع احتمال وزن یال بین رئوس $i$ و $j$ است. فرض بر این است که تابع توزیع احتمال از قبل، شناخته‌شده نیست. هدف، تعیین جایگشت

$\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ است به طوری که وزن درخت پوشای حاصل از به کارگیری این جایگشت کمترین مقدار ممکن باشد. به عبارت دیگر هدف، کمینه‌سازی تابع $\sum_{i=1}^{n-1} \arg\min_j W(\pi_i, j)$ است که در آن $W(s,t)$ نشان‌دهنده وزن یال $(s,t)$ از مجموعه $E$ است.

یک راه حل مبتنی بر آتاماتاهای یادگیر و از طریق حل مسئله جایگشت به کمک DLA برای این مسئله شامل مراحل زیر است:

**الف) ساخت یک شبکه از آتاماتاها متناظر با گراف مسئله:** به هر یک از گره‌های گراف یک آتاماتای یادگیر تخصیص داده می‌شود. هر آتاماتا در یکی از دو وضعیت فعال و غیر فعال قرار دارد. در ابتدای کار تمامی آتاماتاها در وضعیت غیر فعال هستند. هر آتاماتای یادگیر به تعداد یال‌های گره متناظر در گراف، اقدام قابل انجام دارد و تعداد اقدام‌های قابل انجام آتاماتا متغیر است. زمانی که اقدام متناظر با یال $(s,t)$ توسط آتاماتای متناظر با $s$ انجام شود، در مراحل بعدی آتاماتای متناظر با $t$ اقدام $(s,t)$ را در مجموعه اقدام‌های خود نخواهد داشت. این کار برای جلوگیری از تشکیل حلقه در جواب نهایی اعمال می‌شود.

**ب) ساخت شبکه DLA برای یافتن یک جایگشت:** مطابق با الگوریتم ۱ شکل ۳ یک DLA با تعداد $n = |V| + 1$ رأس برای پیداکردن جایگشت بهینه ساخته می‌شود.

**ج) نمونه‌گیری از درخت‌های پوشا:** تا زمان تحقق شرط پایانی (رسیدن به یک جایگشت با احتمالی بیش از یک مقدار آستانه مثل ۹۰٪ یا رسیدن تعداد تکرارها به یک مقدار بیشینه از قبل تعیین شده)، مراحل زیر تکرار می‌شود:

۱) DLA یک جایگشت انتخاب می‌کند. نحوه کار در الگوریتم ۱ شکل ۳ نشان داده شده و در بخش قبلی توضیح داده شد.

۲) بر اساس جایگشت معین‌شده در مرحله قبل، آتاماتاها در گراف به ترتیب فعال شده و تصادفاً یالی را از میان مجموعه یال‌های خود انتخاب می‌کنند. این کار تا ساخت درخت پوشا ادامه می‌یابد. با انتخاب یک یال، این یال از مجموعه یال‌های قابل انتخاب سایر آتاماتاها حذف می‌شود. این کار با غیر فعال کردن اقدام متناظر با آن یال در سایر آتاماتاهای فعال‌نشده، صورت می‌گیرد.

۳) بر اساس وزن درخت پوشای به دست آمده و وزن درخت‌های ساخته‌شده در مراحل قبل، درخت جدید ارزیابی می‌شود. چنانچه وزن آن از میانگین وزن قبلی‌ها کمتر باشد، پاسخ محیط پاداش و در غیر این صورت جریمه خواهد بود.

۴) DLA و کلیه آتاماتاهای متناظر با گره‌ها در گراف، از پاسخ محیط برای به‌روزرسانی بردار احتمال انتخاب اقدام‌های خود استفاده می‌کنند.

**د)** آخرین درخت ساخته‌شده، درخت پوشای کمینه تصادفی خواهد بود.

این الگوریتم را در ادامه الگوریتم Alg۱ می‌نامیم.

## ۳-۵ پیشنهادهایی برای بهبود عملکرد الگوریتم

برای بهبود عملکرد الگوریتم پیشنهادی می‌توان تغییراتی در الگوریتم داد که این تغییرات را در ادامه بررسی خواهیم کرد.

### الف) تغییر در نحوه پاداش‌دهی به اقدام‌های انجام‌شده توسط آتاماتاها

بررسی‌ها نشان می‌دهند که شیوه تقسیم پاداش در یک سیستم چندعامله، می‌تواند تأثیر بسزایی در عملکرد آن داشته باشد.

**بهبود ۱)** یک راه حل برای بهبود شیوه پاداش‌دهی، تقسیم پاداش بین اقدام‌ها بر اساس احتمال انتخاب اقدام‌ها است. به عبارت دیگر در این شیوه، یالی که بیشترین احتمال انتخاب را دارد، پاداش بیشتری دریافت می‌کند. فرض کنید آتاماتای $LA_i$ آتاماتای متناظر با گره $i$ است. تعداد یال‌های متناظر با گره $i$ یا تعداد اقدام‌های $LA_i$ را با $r_i$ نمایش می‌دهیم. فرض کنید بردار احتمال انتخاب اقدام‌های این آتاماتا در زمان $t$ برابر با $P^i(t) = \{p_1^i(t), p_2^i(t), \ldots, p_{r_i}^i(t)\}$ است. نرخ یادگیری برای هر یک از اقدام‌های آتاماتا بر اساس (۴۷) تعیین می‌شود که در آن $a$ نرخ یادگیری آتاماتا را نشان می‌دهد

$$a_i^j(t) = a p_i^j(t) \qquad (۴۸)$$

الگوریتم Alg۱ با این تغییرات، الگوریتم Alg۲ نامیده می‌شود.

### ب) تغییر در نحوه انتخاب اقدام آتاماتا

**بهبود ۲:** از آنجا که وزن یال انتخاب‌شده در هر مرحله توسط آتاماتای متناظر با هر رأس گراف باید کمینه باشد، آتاماتاهای متناظر با رئوس گراف بایستی از یک توزیع احتمالی مبتنی بر وزن برای انتخاب یال‌های خود استفاده کنند. برای انجام این کار به این صورت عمل کرده‌ایم.

فرض کنید آتاماتای $LA_i$ آتاماتای متناظر با گره $i$ است. تعداد یال‌های متناظر با گره $i$ یا تعداد اقدام‌های $LA_i$ را با $r_i$ نمایش می‌دهیم. فرض کنید بردار احتمال انتخاب اقدام‌های این آتاماتا با $P^i(t) = \{p_1^i(t), p_2^i(t), \ldots, p_{r_i}^i(t)\}$ و متوسط وزن نمونه‌گیری شده از هر یک از یال‌های این رأس با $W^i = \{w(i,1), w(i,2), \ldots, w(i,r_i)\}$ نشان داده شده است. برای انتخاب اقدام‌های آتاماتا از (۴۹) استفاده می‌کنیم

$$P^{ti} = \{p_j^{ti} = \frac{(p_j^i \times w^{-1}(i,j))^\beta}{\sum_{h=1}^{r_i} (p_h^i \times w^{-1}(i,h))^\beta} \Big| j = 1,2,\ldots,r_i\} \qquad (۴۹)$$

در (۴۹)، $\beta$ یک ضریب عددی است. در آزمایش‌هایی که در ادامه انجام شده است، مقدار این ضریب به شکل تجربی ۱۰ در نظر گرفته شده است. الگوریتم Alg۱ با این تغییرات را الگوریتم Alg۳ نامیده‌ایم.

### ج) تغییر در مقدار آستانه پویا و نحوه ارزیابی عملکرد آتاماتاها

**بهبود ۳:** در مسایل بهینه‌سازی و جستجوی تصادفی، برای افزایش سرعت همگرایی و کاهش حساسیت الگوریتم به خطاهای مدل، یک نویز به صورت تعمدی در فرایند جستجو تزریق می‌شود. علاوه بر این، بررسی‌ها نشان می‌دهند که به دلیل نزدیک‌شدن مقدار آستانه پویا در الگوریتم پیشنهادی به وزن درخت پوشای کمینه بهینه، از یک مرحله به بعد، قدرت یادگیری DLA کاهش می‌یابد. برای حل این مشکل، به جای مقدار آستانه پویا که یک میانگین‌گیری روی وزن درخت‌های قبلی است، از معیار دیگری استفاده می‌کنیم که علاوه بر وزن، واریانس وزن درخت‌های به دست آمده را نیز در محاسبات لحاظ کند. اما برای این که پیچیدگی محاسباتی این روش کاهش یابد، ناچاریم از تخمین‌های ساده نظیر تخمین‌های stochastic gradient استفاده کنیم. برای به دست آوردن معیار جدید به این صورت عمل کرده‌ایم:

۱) تفاوت وزن درخت به دست آمده $(W_{k+1})$ در هر مرحله با میانگین وزن درخت‌های به دست آمده تا قبل از آن مرحله $(WTH_k)$ محاسبه می‌گردد که این مقدار $Err$ نامیده می‌شود $(Err_{k+1} = W_{k+1} - WTH_k)$.

۲) از مقدار $Err$ برای به‌روزرسانی مقدار آستانه پویا استفاده می‌شود $(WTH_{k+1} = WTH_k + aErr_{k+1})$.

۳) از یک متغیر نظیر $V$ به عنوان تخمینی از واریانس استفاده می‌کنیم. مقدار این متغیر در هر مرحله از طریق یک تخمین‌گر خطی به صورت $V_{k+1} = V_k + \beta(abs(Err_{k+1}) - V_k)$ محاسبه می‌شود و کاملاً واضح است که این تخمین‌گر، انحراف از میانگین $(mdev)$ (نه انحراف معیار $(sdev)$) را محاسبه می‌کند. اما می‌دانیم که

$$mdev^2 = (\sum |W - W_{TH}|)^2 \geq \sum |W - W_{TH}|^2 = \delta^2 = sdev^2 \quad (۵۰)$$

رابطه (۵۰) نشان می‌دهد که انحراف از میانگین بزرگ‌تر از انحراف معیار و لذا محافظه‌کارانه‌تر است، ضمن این که محاسبه آن فوق‌العاده ساده‌تر از واریانس می‌باشد.

۴) برای مقایسه و تعیین پاداش یا جریمه آتاماتاها از کران بالای $WTH_k/2 + 2 \times V_k$ استفاده می‌کنیم. این ترکیب به صورت تجربی معین شده است.

الگوریتم Alg۱ با این تغییرات را الگوریتم Alg۴ نامیده‌ایم.

**بهبود ٤:** الگوریتم Alg۱ با تغییرات حاصل از ترکیب بهبودهای ۱، ۲ و ۳ را الگوریتم Alg۵ نامیده‌ایم.

## ٥-٤ آزمایش ۱

هدف از این سری آزمایش‌ها، بررسی نحوه عملکرد الگوریتم پیشنهادی و بهبودهای ارائه‌شده در حل مسئله یافتن درخت پوشای کمینه تصادفی است. برای این منظور از ۳ گراف alex۱−b، alex۲−b و alex۳−b استفاده کرده‌ایم [۶۵]. در تمامی این گراف‌ها، یال‌های دارای یک توزیع احتمالی گسسته با مقادیر مثبت هستند و توزیع احتمالی وزن یال‌ها به گونه‌ای است که یال‌های با وزن کمتر، دارای احتمال بیشتری هستند. این ویژگی، گراف‌ها را با دنیای واقعی منطبق‌تر می‌کند. برای بررسی عملکرد الگوریتم‌های پیشنهادی، مسئله یافتن درخت پوشای کمینه را به عنوان یک مسئله بهینه‌سازی خطی تصادفی روی گروه جایگشت به کمک آتاماتای یادگیر توزیع‌شده حل کرده‌ایم. برای مقایسه عملکرد الگوریتم از پنج شاخص متوسط تعداد نمونه‌گیری از یال‌ها $(AVS)$، متوسط تعداد تکرارهای لازم برای همگرایی $(AVI)$، متوسط زمان لازم برای یافتن درخت پوشای کمینه $(AVT)$، درصد همگرایی $(PC)$ و تعداد کل درخت‌های متمایز جستجوشده $(DST)$ استفاده گردیده است. شاخص $DST$ در کل زمان اجرای الگوریتم محاسبه شده و یک شاخص متوسط نیست. نحوه انجام آزمایش به این صورت بوده که از الگوریتم خواسته شده تا کار یافتن درخت پوشای کمینه تصادفی را از طریق نمونه‌گیری از یال‌ها انجام دهد. در الگوریتم پیشنهادی، نحوه فعال‌شدن گره‌ها از طریق حل مسئله بهینه‌سازی جایگشت به کمک الگوریتم ۱ ارائه‌شده در شکل ۳ صورت می‌گیرد. فرایند یافتن درخت پوشا تا رسیدن به درختی با احتمال بیش از ۹۰٪ یا نمونه‌گیری از ۵۰۰۰۰ درخت پوشا ادامه می‌یابد. درخت به دست آمده با درخت پوشای کمینه تصادفی مقایسه گردیده و چنانچه این دو درخت یکسان باشند، الگوریتم همگرا و در غیر این صورت واگرا است. مقادیر شاخص‌ها روی ۵۰ بار تکرار هر دو الگوریتم روی گراف‌ها محاسبه شده‌اند.

بر اساس نتایج مندرج در جداول می‌توان مشاهده کرد که:

۱) در تمام الگوریتم‌های پیشنهادی، تعداد نمونه‌ها متناسب با مقدار نرخ یادگیری است؛ به گونه‌ای که با افزایش نرخ یادگیری، تعداد نمونه‌ها کاهش یافته و بالعکس با کاهش نرخ یادگیری، تعداد نمونه‌ها افزایش می‌یابد (بررسی جدول‌ها از جدول ۲ تا ۶). علاوه بر این، بررسی‌ها نشان می‌دهند که با کاهش نرخ یادگیری، دقت $(PC)$ و متوسط مدت زمان لازم برای یافتن درخت پوشای کمینه $(AVT)$ افزایش می‌یابند (بررسی جدول‌ها از جدول ۷ تا ۹). در مورد دقت الگوریتم، میزان وابستگی دقت الگوریتم‌های ۱ تا ۳ به نرخ یادگیری بسیار بیشتر از الگوریتم‌های ۴ و ۵ است، به گونه‌ای که به عنوان مثال الگوریتم ۵ روی گراف alex۱−b تقریباً مستقل از نرخ یادگیری همواره همگرا است.

۲) در یک نرخ یادگیری ثابت، مستقل از گراف مسئله و ویژگی‌های آن، متوسط تعداد نمونه‌ها در الگوریتم‌های ۱ تا ۳ از الگوریتم‌های ۴ و ۵ کمتر است. با توجه به این که فصل مشترک الگوریتم‌های ۴ و ۵ استفاده از معیار مبتنی بر واریانس معرفی‌شده در بهبود ۳ است، بنابراین به نظر می‌رسد که استفاده از این معیار مبتنی بر واریانس، منجر به افزایش تعداد متوسط نمونه‌ها می‌شود. این ویژگی در مورد متوسط تعداد تکرارهای لازم برای همگرایی $(AVI)$ نیز صادق است. یعنی با فرض ثابت‌بودن نرخ یادگیری، مقدار این شاخص در الگوریتم‌های ۴ و ۵ از مقادیر متناظرشان در الگوریتم‌های ۱ تا ۳ بیشتر است.

۳) شاخص $DST$ نشان می‌دهد که الگوریتم‌های ۳ و ۵ به شدت جستجوی محدودشده‌تری دارند، به گونه‌ای که تعداد درخت‌ها تا حدود ۹۰٪ کاهش یافته است. این نتایج به همراه نتایج مربوط به دقت $(PC)$ این دو الگوریتم بیانگر آن است که دخالت‌دادن وزن متوسط یال‌های نمونه‌گیری شده به همراه احتمال انتخاب هر یک از اقدام‌ها می‌تواند باعث دقت الگوریتم و سرعت همگرایی بالای آن شود.

۴) بررسی نتایج در جداول ۷ تا ۹ نشان می‌دهند که الگوریتم ۳ به لحاظ متوسط زمان لازم برای پیداکردن جواب بهینه در یک نرخ یادگیری ثابت از هر یک از الگوریتم‌های ۱و ۲ عملکرد بهتری دارد. این ویژگی مستقل از نوع گراف برقرار است. علاوه بر این مشاهده می‌شود که الگوریتم‌های ۴ و ۵ به دلیل استفاده از معیار مقایسه‌ای بهبودیافته مبتنی بر واریانس، نسبت به سایر الگوریتم‌ها تقریباً در نرخ‌های مختلف یادگیری، از نظر دقت $(PC)$ عملکرد بهتری دارند. به عنوان مثال، در حالی که الگوریتم‌های ۱ تا ۳ روی گراف alex۲−b حداکثر دقت ۵۰٪ را دارند، الگوریتم‌های ۴ و ۵ به ترتیب دقت ۹۰٪ و ۹۵٪ را دارا هستند. بررسی جداول نشان می‌دهند که به ازای مقادیر نرخ یادگیری کوچک‌تر از ۰/۰۹، الگوریتم‌های ۴ و ۵ با دقت بالای ۹۰٪ همگرا هستند. به نظر می‌رسد که معیار جدید مقایسه‌ای مبتنی بر واریانس می‌تواند چالش تعیین مقدار نرخ یادگیری در آتاماتای یادگیر را که یکی از چالش‌های سنتی استفاده از این روش محسوب می‌شود تا حدود زیادی برطرف کند.

۵) علی‌رغم دقت مشابه الگوریتم‌های ۴ و ۵، بررسی جداول حاکی از برتری الگوریتم ۵ نسبت به الگوریتم ۴ از نظر متوسط زمان لازم برای یافتن درخت پوشای بهینه است. میزان بهبود در متوسط زمان لازم برای یافتن درخت پوشای کمینه در الگوریتم ۵ نسبت به الگوریتم ۴ بین ۳۰٪ (در شبکه‌های alex۱−b و alex۲−b) تا ۵۰٪ در شبکه alex۳−b متغیر است.

جدول ۲: مقادیر شاخص‌های مقایسه‌ای روی سه گراف نمونه به ازای مقادیر مختلف نرخ یادگیری.

| Learning rate | Alg۱ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Alex۱ – b | | | Alex۲ – b | | | Alex۳ – b | | |
| | AVS | AVI | DST | AVS | AVI | DST | AVS | AVI | DST |
| ۰/۳ | ۱۱۹۰ | ۴۱۷ | ۲۳۹ | ۱۲۶۸ | ۳۸۴ | ۲۴۵ | ۱۷۰۶ | ۲۴۷۷ | ۵۷۸ |
| ۰/۲ | ۱۵۸۸ | ۳۸۱ | ۲۶۴ | ۱۷۰۰ | ۵۵۶ | ۲۷۸ | ۲۲۲۶ | ۱۳۴۲ | ۷۵۵ |
| ۰/۱ | ۳۰۹۵ | ۶۱۹ | ۳۲۸ | ۳۹۶۴ | ۱۲۷۸ | ۳۱۸ | ۴۷۵۱ | ۲۴۲۵ | ۹۵۱ |
| ۰/۰۹ | ۳۷۹۱ | ۸۴۳ | ۳۲۵ | ۴۳۰۷ | ۱۰۶۶ | ۳۲۴ | ۵۷۲۰ | ۳۵۴۲ | ۹۷۶ |
| ۰/۰۸ | ۴۶۴۵ | ۹۳۷ | ۳۳۲ | ۵۰۳۳ | ۱۵۲۱ | ۳۴۱ | ۶۸۱۳ | ۳۸۵۷ | ۱۰۴۰ |
| ۰/۰۷ | ۵۳۹۲ | ۱۲۰۴ | ۳۳۴ | ۶۲۴۷ | ۱۷۶۱ | ۳۴۵ | ۷۹۰۵ | ۲۸۴۱ | ۹۷۹ |
| ۰/۰۶ | ۶۵۲۲ | ۹۹۰ | ۳۲۱ | ۸۲۶۲ | ۱۹۲۱ | ۳۴۷ | ۹۷۴۲ | ۴۲۸۶ | ۱۰۷۱ |
| ۰/۰۵ | ۹۲۶۴ | ۱۴۶۸ | ۳۵۱ | ۱۰۶۷۵ | ۲۱۸۵ | ۳۴۶ | ۱۳۵۵۰ | ۳۷۹۵ | ۱۰۰۱ |
| ۰/۰۴ | ۱۴۷۵۱ | ۲۰۲۸ | ۳۴۸ | ۱۶۰۶۲ | ۳۰۶۱ | ۳۶۶ | ۱۸۱۷۷ | ۳۹۱۸ | ۱۰۹۰ |
| ۰/۰۳ | ۲۳۷۳۷ | ۳۳۳۸ | ۳۵۲ | ۲۷۴۰۱ | ۵۹۲۲ | ۳۸۵ | ۳۱۴۳۶ | ۶۷۰۸ | ۱۱۷۵ |
| ۰/۰۲ | ۵۴۷۳۹ | ۷۶۴۸ | ۳۸۲ | ۶۲۸۱۳ | ۹۴۴۶ | ۴۱۶ | ۶۳۸۰۸ | ۱۰۸۶۱ | ۱۱۴۰ |
| ۰/۰۱ | ۱۷۶۶۷۷ | ۲۴۳۱۹ | ۴۰۶ | ۲۲۵۰۱۷ | ۳۲۵۴۴ | ۴۲۱ | ۲۳۸۳۲۹ | ۲۹۷۹۵ | ۱۳۲۶ |

جدول ۳: مقادیر شاخص‌های مقایسه‌ای روی سه گراف نمونه به ازای مقادیر مختلف نرخ یادگیری.

| Learning rate | Alg۲ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Alex۱ – b | | | Alex۲ – b | | | Alex۳ – b | | |
| | AVS | AVI | DST | AVS | AVI | DST | AVS | AVI | DST |
| ۰/۳ | ۱۱۹۱ | ۵۰۱ | ۲۵۰ | ۱۲۶۵ | ۳۸۱ | ۲۵۸ | ۱۷۵۵ | ۲۷۵۳ | ۵۷۷ |
| ۰/۲ | ۱۶۰۱ | ۶۰۴ | ۲۷۸ | ۱۷۷۱ | ۶۰۵ | ۲۷۵ | ۲۲۵۰ | ۵۵۰۰ | ۷۴۰ |
| ۰/۱ | ۳۳۸۳ | ۷۹۰ | ۳۲۴ | ۳۷۵۲ | ۱۳۷۳ | ۳۲۶ | ۴۷۰۹ | ۲۱۳۰ | ۹۴۹ |
| ۰/۰۹ | ۳۷۷۲ | ۷۷۶ | ۳۲۶ | ۴۳۲۲ | ۱۲۶۰ | ۳۲۹ | ۵۳۸۳ | ۲۲۹۲ | ۹۹۰ |
| ۰/۰۸ | ۴۴۳۴ | ۷۷۸ | ۳۲۹ | ۵۱۹۶ | ۱۳۴۴ | ۳۲۶ | ۶۲۲۳ | ۴۵۹۸ | ۱۰۰۲ |
| ۰/۰۷ | ۵۴۲۲ | ۹۶۳ | ۳۴۲ | ۵۹۲۸ | ۱۳۶۸ | ۳۵۲ | ۷۸۲۱ | ۳۳۱۵ | ۱۰۰۴ |
| ۰/۰۶ | ۷۳۱۰ | ۱۴۰۱ | ۳۲۸ | ۸۰۳۵ | ۲۹۷۷ | ۳۰۷ | ۱۰۱۴۳ | ۳۷۹۸ | ۱۰۴۷ |
| ۰/۰۵ | ۹۴۴۴ | ۱۴۶۳ | ۳۶۲ | ۱۲۶۵۲ | ۲۶۲۶ | ۳۴۲ | ۱۳۳۰۳ | ۴۱۸۳ | ۱۰۴۰ |
| ۰/۰۴ | ۱۴۱۲۲ | ۲۰۱۹ | ۳۶۱ | ۱۷۶۶۹ | ۳۴۹۷ | ۳۴۴ | ۱۸۸۱۶ | ۳۳۶۴ | ۱۱۲۷ |
| ۰/۰۳ | ۲۲۷۷۶ | ۳۳۳۴ | ۳۵۳ | ۲۶۳۰۲ | ۳۹۷۴ | ۳۷۷ | ۳۲۰۶۸ | ۵۷۰۷ | ۱۱۳۸ |
| ۰/۰۲ | ۵۰۸۱۲ | ۷۱۳۹ | ۴۰۱ | ۶۴۹۷۷ | ۱۰۰۲۲ | ۴۱۶ | ۷۰۷۴۸ | ۱۰۱۶۱ | ۱۱۶۰ |
| ۰/۰۱ | ۲۰۱۰۲۳ | ۲۶۸۸۴ | ۴۰۱ | ۱۹۷۷۰۰ | ۲۸۶۸۷ | ۴۱۲ | ۲۷۲۶۹۳ | ۳۲۱۰۲ | ۱۳۰۵ |

جدول ۴: مقادیر شاخص‌های مقایسه‌ای روی سه گراف نمونه به ازای مقادیر مختلف نرخ یادگیری.

| Learning rate | Alg۳ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Alex۱ – b | | | Alex۲ – b | | | Alex۳ – b | | |
| | AVS | AVI | DST | AVS | AVI | DST | AVS | AVI | DST |
| ۰/۳ | ۱۱۸۴ | ۷۲ | ۳۰ | ۱۳۵۶ | ۲۰۵ | ۲۹ | ۱۸۱۹ | ۲۳۹ | ۹۵ |
| ۰/۲ | ۱۵۲۴ | ۱۱۱ | ۳۰ | ۱۸۳۴ | ۴۱۸ | ۳۴ | ۲۴۱۱ | ۸۰۰ | ۱۰۴ |
| ۰/۱ | ۳۱۷۹ | ۳۷۱ | ۳۴ | ۴۱۹۷ | ۹۸۴ | ۳۴ | ۴۷۴۱ | ۸۸۰ | ۱۰۹ |
| ۰/۰۹ | ۳۸۷۶ | ۴۵۲ | ۳۱ | ۵۲۶۷ | ۱۰۹۵ | ۳۳ | ۵۳۴۲ | ۱۰۸۲ | ۱۲۵ |
| ۰/۰۸ | ۴۴۴۵ | ۵۱۵ | ۳۰ | ۵۲۷۸ | ۱۱۴۴ | ۳۶ | ۷۰۵۴ | ۱۳۴۳ | ۱۰۵ |
| ۰/۰۷ | ۶۱۶۵ | ۸۲۷ | ۲۹ | ۶۵۶۷ | ۱۳۱۶ | ۳۴ | ۸۰۹۴ | ۱۴۰۸ | ۱۲۶ |
| ۰/۰۶ | ۷۰۸۱ | ۹۱۰ | ۳۱ | ۹۱۶۰ | ۱۶۷۱ | ۳۷ | ۹۵۵۵ | ۱۳۹۸ | ۱۲۲ |
| ۰/۰۵ | ۹۲۳۹ | ۱۲۷۶ | ۳۲ | ۱۲۰۹۹ | ۲۸۰۸ | ۳۲ | ۱۳۲۵۷ | ۲۳۰۱ | ۱۲۷ |
| ۰/۰۴ | ۱۴۱۴۰ | ۱۹۰۰ | ۳۳ | ۱۷۸۶۴ | ۳۹۳۷ | ۳۸ | ۱۹۰۰۷ | ۲۹۹۲ | ۱۲۰ |
| ۰/۰۳ | ۲۱۴۸۲ | ۲۹۴۹ | ۳۴ | ۲۹۰۲۷ | ۵۵۰۷ | ۴۲ | ۳۰۸۹۸ | ۴۹۹۵ | ۱۲۱ |
| ۰/۰۲ | ۴۷۰۰۳ | ۶۷۲۹ | ۳۵ | ۶۸۳۶۶ | ۱۱۷۳۲ | ۴۱ | ۵۰۸۱۸ | ۷۰۶۸ | ۱۲۴ |
| ۰/۰۱ | ۱۸۴۴۵۶ | ۲۵۳۷۶ | ۳۶ | ۲۳۸۷۴۲ | ۴۰۲۹۰ | ۳۶ | ۱۷۱۵۷۴ | ۲۰۶۸۲ | ۱۲۸ |

جدول ۵: مقادیر شاخص‌های مقایسه‌ای روی سه گراف نمونه به ازای مقادیر مختلف نرخ یادگیری.

| Learning rate | Alg۴ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Alex۱ – b | | | Alex۲ – b | | | Alex۳ – b | | |
| | AVS | AVI | DST | AVS | AVI | DST | AVS | AVI | DST |
| ۰/۳ | ۲۴۰۷ | ۷۰۰ | ۳۱۵ | ۳۳۴۵ | ۷۲۸ | ۲۸۸ | ۵۱۷۷ | ۲۷۲۱ | ۹۲۸ |
| ۰/۲ | ۵۱۵۳ | ۷۱۶ | ۳۴۰ | ۸۸۳۵ | ۱۶۰۱ | ۲۸۱ | ۷۹۷۴ | ۱۷۷۷ | ۸۸۸ |
| ۰/۱ | ۳۲۱۴۸ | ۴۵۶۴ | ۳۱۴ | ۲۳۷۳۶ | ۲۹۱۵ | ۳۱۱ | ۲۲۸۱۰ | ۲۸۵۱ | ۸۱۳ |
| ۰/۰۹ | ۴۴۲۸۸ | ۶۲۰۷ | ۳۲۸ | ۳۳۲۳۰ | ۴۴۹۳ | ۳۳۷ | ۳۳۲۳۳ | ۴۲۳۰ | ۷۹۶ |
| ۰/۰۸ | ۵۰۴۲۲ | ۷۲۲۸ | ۳۲۱ | ۴۴۸۳۸ | ۵۹۲۶ | ۳۵۴ | ۴۰۱۶۵ | ۴۹۱۳ | ۸۵۸ |
| ۰/۰۷ | ۷۱۶۶۸ | ۱۰۱۱۸ | ۳۴۱ | ۴۳۳۵۲ | ۵۳۱۰ | ۳۴۰ | ۵۰۹۶۴ | ۶۱۳۷ | ۸۸۸ |
| ۰/۰۶ | ۱۰۴۹۳۱ | ۱۴۷۶۲ | ۳۴۷ | ۵۶۶۶۳ | ۷۱۱۸ | ۳۴۱ | ۷۱۴۴۴ | ۷۹۵۸ | ۸۷۷ |
| ۰/۰۵ | ۱۴۸۰۹۸ | ۲۱۰۳۷ | ۳۳۶ | ۸۲۶۳۷ | ۱۰۲۲۲ | ۳۵۶ | ۱۲۱۷۴۵ | ۱۳۶۶۱ | ۸۵۲ |
| ۰/۰۴ | ۲۱۰۱۴۸ | ۲۶۹۱۷ | ۳۵۷ | ۱۳۰۱۸۰ | ۱۶۱۶۸ | ۳۶۸ | ۱۷۲۳۶۹ | ۱۹۰۱۳ | ۸۷۵ |
| ۰/۰۳ | ۳۰۵۵۸۵ | ۳۸۵۲۲ | ۳۶۵ | ۲۲۷۸۹۲ | ۲۷۵۷۶ | ۳۶۶ | ۲۷۳۱۰۷ | ۲۸۴۶۱ | ۹۹۱ |
| ۰/۰۲ | ۳۴۶۳۹۲ | ۴۰۴۳۰ | ۳۷۲ | ۳۵۹۴۸۳ | ۳۹۸۰۸ | ۳۹۸ | ۴۲۸۴۹۷ | ۴۰۴۲۹ | ۱۰۱۲ |
| ۰/۰۱ | ۳۵۰۸۴۰ | ۴۳۹۲۰ | ۴۰۸ | ۴۰۰۶۲۰ | ۴۴۹۴۷ | ۴۱۹ | ۴۵۱۲۰۵ | ۴۴۷۹۸ | ۱۱۶۱ |

جدول ۶: مقادیر شاخص‌های مقایسه‌ای روی سه گراف نمونه به ازای مقادیر مختلف نرخ یادگیری.

| Learning rate | Alg۵ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Alex۱ – b | | | Alex۲ – b | | | Alex۳ – b | | |
| | AVS | AVI | DST | AVS | AVI | DST | AVS | AVI | DST |
| ۰/۳ | ۳۷۸۷ | ۴۷۸ | ۳۲ | ۵۸۱۹ | ۱۱۸۶ | ۳۰ | ۴۳۴۹ | ۱۴۳۲ | ۱۱۱ |
| ۰/۲ | ۷۴۱۳ | ۹۳۹ | ۳۲ | ۱۱۶۲۳ | ۱۸۶۶ | ۳۰ | ۷۲۸۰ | ۹۸۵ | ۱۱۰ |
| ۰/۱ | ۳۶۲۱۱ | ۵۰۵۳ | ۳۴ | ۲۱۹۴۲ | ۲۶۸۸ | ۳۳ | ۲۳۱۶۲ | ۳۱۲۳ | ۱۱۴ |
| ۰/۰۹ | ۴۴۷۳۹ | ۶۲۷۱ | ۳۴ | ۳۰۵۵۵ | ۳۷۸۹ | ۳۵ | ۲۹۷۳۱ | ۴۰۵۹ | ۱۱۷ |
| ۰/۰۸ | ۵۴۹۰۹ | ۷۷۲۴ | ۳۲ | ۴۳۸۴۰ | ۵۶۰۱ | ۳۶ | ۳۸۸۹۴ | ۴۷۵۵ | ۱۰۶ |
| ۰/۰۷ | ۷۱۴۵۶ | ۱۰۰۸۸ | ۳۲ | ۵۳۵۸۰ | ۶۷۹۷ | ۳۳ | ۴۲۴۱۴ | ۵۰۸۵ | ۱۱۷ |
| ۰/۰۶ | ۹۹۹۴۳ | ۱۴۱۵۸ | ۳۲ | ۶۰۰۹۱ | ۷۵۶۴ | ۳۳ | ۵۱۹۴۹ | ۶۱۲۷ | ۹۹ |
| ۰/۰۵ | ۱۴۹۱۱۴ | ۲۰۹۵۸ | ۲۹ | ۸۱۷۵۷ | ۱۰۱۲۲ | ۳۴ | ۷۳۵۶۵ | ۹۳۵۰ | ۱۲۴ |
| ۰/۰۴ | ۲۲۱۰۰۹ | ۲۹۷۹۶ | ۳۱ | ۱۲۴۴۳۰ | ۱۵۴۶۳ | ۳۹ | ۱۳۵۰۷۴ | ۱۷۳۰۰ | ۱۰۹ |
| ۰/۰۳ | ۳۱۳۹۸۷ | ۴۰۰۶۹ | ۲۸ | ۲۱۴۹۳۱ | ۲۶۱۲۴ | ۳۸ | ۲۰۹۶۶۲ | ۲۵۸۰۷ | ۱۲۵ |
| ۰/۰۲ | ۳۳۹۸۰۴ | ۴۱۲۹۶ | ۳۶ | ۳۶۵۰۸۴ | ۳۹۲۰۱ | ۳۲ | ۳۵۱۶۳۳ | ۳۵۷۵۳ | ۱۲۴ |
| ۰/۰۱ | ۳۵۰۸۴۰ | ۴۲۶۹۳ | ۳۵ | ۳۹۷۲۹۳ | ۴۵۱۹۷ | ۳۴ | ۴۴۴۲۲۰ | ۴۴۱۳۴ | ۱۱۸ |

جدول ۷: مقادیر شاخص‌های AVT و PC روی گراف Alex۱ – b به ازای مقادیر مختلف نرخ یادگیری در الگوریتم‌های پیشنهادی.

| Learning rate | Alex۱ – b | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Alg۱ | | Alg۲ | | Alg۳ | | Alg۴ | | Alg۵ | |
| | AVT | PC | AVT | PC | AVT | PC | AVT | PC | AVT | PC |
| ۰/۳ | ۲/۴۳ | ۱۲٪ | ۲/۹۶ | ۱۰٪ | ۰/۳۵ | ۶۸٪ | ۴/۳۵ | ۳۲٪ | ۲/۰۱ | ۸۸٪ |
| ۰/۲ | ۲/۳۳ | ۲۸٪ | ۳/۶۶ | ۱۸٪ | ۰/۵۰ | ۸۸٪ | ۴/۱۸ | ۸۶٪ | ۳/۹۷ | ۱۰۰٪ |
| ۰/۱ | ۳/۹۹ | ۵۲٪ | ۵/۲۱ | ۴۶٪ | ۱/۷۲ | ۹۰٪ | ۲۶/۷۶ | ۹۸٪ | ۲۱/۵۷ | ۱۰۰٪ |
| ۰/۰۹ | ۵/۴۶ | ۵۰٪ | ۴/۵۵ | ۵۴٪ | ۱/۸۵ | ۹۶٪ | ۳۴/۵۸ | ۱۰۰٪ | ۲۴/۷۱ | ۱۰۰٪ |
| ۰/۰۸ | ۶/۲۶ | ۵۸٪ | ۵/۲۵ | ۶۶٪ | ۲/۳۵ | ۱۰۰٪ | ۴۰/۲۴ | ۹۸٪ | ۳۲/۷۴ | ۱۰۰٪ |
| ۰/۰۷ | ۷/۴۸ | ۵۴٪ | ۵/۸۸ | ۶۸٪ | ۳/۶۱ | ۹۲٪ | ۵۶/۲۹ | ۱۰۰٪ | ۳۹/۳۲ | ۱۰۰٪ |
| ۰/۰۶ | ۵/۹۰ | ۸۲٪ | ۸/۲۷ | ۶۶٪ | ۳/۷۲ | ۹۸٪ | ۸۴/۱۳ | ۱۰۰٪ | ۵۴/۶۷ | ۱۰۰٪ |
| ۰/۰۵ | ۹/۵۸ | ۸۲٪ | ۹/۴۴ | ۸۴٪ | ۵/۶۶ | ۹۴٪ | ۱۱۹/۱۲ | ۱۰۰٪ | ۷۹/۱۳ | ۱۰۰٪ |
| ۰/۰۴ | ۱۱/۷۱ | ۹۸٪ | ۱۲/۳۷ | ۹۴٪ | ۷/۹۴ | ۱۰۰٪ | ۱۵۶/۳۷ | ۱۰۰٪ | ۱۱۳/۸۴ | ۱۰۰٪ |
| ۰/۰۳ | ۲۴/۴۷ | ۹۸٪ | ۲۲/۹۲ | ۹۴٪ | ۱۴/۰۴ | ۱۰۰٪ | ۲۷۱/۵۱ | ۱۰۰٪ | ۱۴۲/۲۶ | ۱۰۰٪ |
| ۰/۰۲ | ۵۱/۱۱ | ۱۰۰٪ | ۴۴/۳۲ | ۱۰۰٪ | ۲۹/۰۱ | ۹۸٪ | ۳۲۶/۶۹ | ۱۰۰٪ | ۱۹۸/۵۹ | ۱۰۰٪ |
| ۰/۰۱ | ۱۷۵/۳۶ | ۱۰۰٪ | ۱۸۵/۳۵ | ۱۰۰٪ | ۱۰۸/۶۵ | ۱۰۰٪ | ۳۱۹/۴۰ | ۹۸٪ | ۲۲۴/۱۸ | ۱۰۰٪ |

جدول ۸: مقادیر شاخص‌های AVT و PC روی گراف Alex۲ – b به ازای مقادیر مختلف نرخ یادگیری در الگوریتم‌های پیشنهادی.

| Learning rate | Alex۲ – b | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Alg۱ | | Alg۲ | | Alg۳ | | Alg۴ | | Alg۵ | |
| | AVT | PC | AVT | PC | AVT | PC | AVT | PC | AVT | PC |
| ۰/۳ | ۲/۷۱ | ۱۲٪ | ۲/۵۸ | ۱۲٪ | ۱/۱۵ | ۲۸٪ | ۴/۸۹ | ۴۲٪ | ۶/۰۵ | ۵۲٪ |
| ۰/۲ | ۳/۸۶ | ۱۸٪ | ۴/۳۲ | ۱۸٪ | ۲/۲۵ | ۲۸٪ | ۱۰/۶۷ | ۶۲٪ | ۹/۴۵ | ۷۲٪ |
| ۰/۱ | ۹/۹۹ | ۳۰٪ | ۱۰/۶۵ | ۲۶٪ | ۵/۵۳ | ۴۲٪ | ۱۹/۹۴ | ۹۸٪ | ۱۳/۶۹ | ۹۸٪ |
| ۰/۰۹ | ۸/۴۰ | ۴۰٪ | ۹/۸۴ | ۳۴٪ | ۶/۴۶ | ۵۰٪ | ۳۱/۴۳ | ۹۰٪ | ۱۸/۸۶ | ۹۸٪ |
| ۰/۰۸ | ۱۱/۰۹ | ۳۴٪ | ۱۰/۶۰ | ۴۰٪ | ۶/۲۰ | ۴۸٪ | ۳۵/۷۷ | ۹۲٪ | ۲۵/۷۳ | ۹۶٪ |
| ۰/۰۷ | ۱۲/۷۶ | ۳۸٪ | ۱۰/۰۳ | ۴۶٪ | ۶/۹۰ | ۵۴٪ | ۳۴/۲۴ | ۱۰۰٪ | ۳۴/۸۱ | ۹۶٪ |
| ۰/۰۶ | ۱۴/۴۸ | ۴۸٪ | ۲۱/۴۰ | ۳۰٪ | ۹/۰۶ | ۶۲٪ | ۴۷/۱۳ | ۹۸٪ | ۳۴/۵۰ | ۹۸٪ |
| ۰/۰۵ | ۱۵/۸۳ | ۵۶٪ | ۱۸/۹۵ | ۵۶٪ | ۱۴/۶۲ | ۵۰٪ | ۶۶/۸۹ | ۱۰۰٪ | ۴۳/۴۲ | ۱۰۰٪ |
| ۰/۰۴ | ۲۱/۵۴ | ۶۲٪ | ۲۴/۸۵ | ۶۰٪ | ۱۹/۲۲ | ۵۴٪ | ۹۶/۵۹ | ۱۰۰٪ | ۷۲/۸۷ | ۱۰۰٪ |
| ۰/۰۳ | ۳۸/۹۱ | ۵۶٪ | ۲۹/۰۶ | ۸۰٪ | ۳۴/۸۳ | ۶۴٪ | ۱۹۳/۵۲ | ۱۰۰٪ | ۱۳۶/۳۹ | ۱۰۰٪ |
| ۰/۰۲ | ۸۳/۰۲ | ۸۲٪ | ۷۳/۹۴ | ۸۰٪ | ۵۷/۶۳ | ۷۲٪ | ۲۸۱/۱۵ | ۱۰۰٪ | ۱۸۲/۵۰ | ۱۰۰٪ |
| ۰/۰۱ | ۳۰۷/۹۳ | ۸۲٪ | ۲۵۳/۲۱ | ۸۲٪ | ۲۲۹/۴۵ | ۷۲٪ | ۳۶۶/۱۷ | ۱۰۰٪ | ۲۳۸/۵۵ | ۱۰۰٪ |

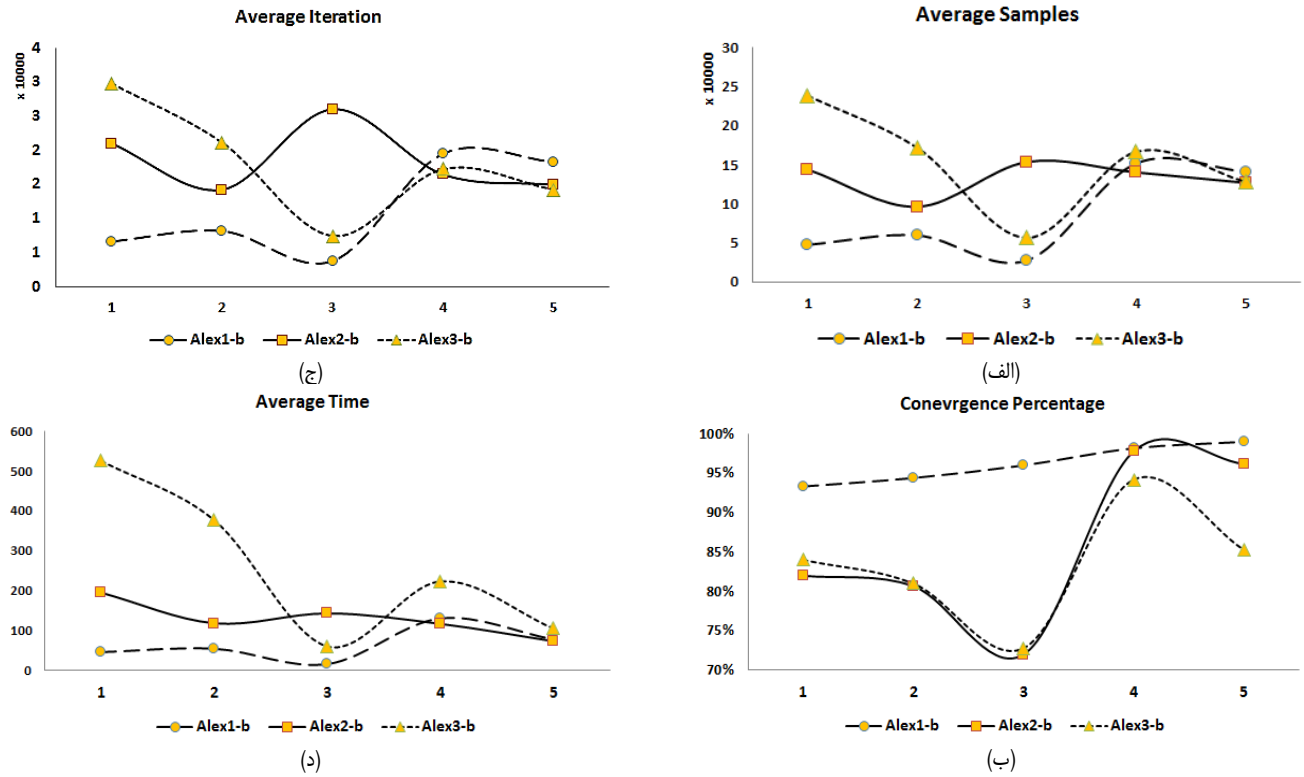
جدول ۹: مقادیر شاخص‌های AVT و PC روی گراف Alex۳ – b به ازای مقادیر مختلف نرخ یادگیری در الگوریتم‌های پیشنهادی.

| Learning rate | Alex۳ – b | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Alg۱ | | Alg۲ | | Alg۳ | | Alg۴ | | Alg۵ | |
| | AVT | PC | AVT | PC | AVT | PC | AVT | PC | AVT | PC |
| ۰/۳ | ۲۴/۰۰ | ۲٪ | ۲۸/۱۰ | ۲٪ | ۱/۷۴ | ۲۶٪ | ۳۰/۸۸ | ۱۶٪ | ۹/۶۶ | ۴٪ |
| ۰/۲ | ۱۴/۶۳ | ۸٪ | ۶۲/۱۱ | ۲٪ | ۵/۹۲ | ۱۶٪ | ۱۹/۶۵ | ۴۲٪ | ۶/۵۱ | ۶۸٪ |
| ۰/۱ | ۳۱/۳۵ | ۱۶٪ | ۲۷/۰۴ | ۱۸٪ | ۶/۶۷ | ۴۴٪ | ۳۰/۴۱ | ۸۴٪ | ۲۰/۹۴ | ۷۸٪ |
| ۰/۰۹ | ۴۳/۴۰ | ۱۴٪ | ۲۵/۵۰ | ۲۰٪ | ۷/۶۵ | ۴۲٪ | ۳۷/۶۹ | ۸۴٪ | ۲۶/۹۵ | ۷۸٪ |
| ۰/۰۸ | ۴۵/۲۶ | ۱۶٪ | ۵۴/۴۲ | ۱۲٪ | ۹/۱۸ | ۴۸٪ | ۴۹/۵۸ | ۸۸٪ | ۲۹/۱۱ | ۸۸٪ |
| ۰/۰۷ | ۳۴/۰۹ | ۲۶٪ | ۴۰/۴۸ | ۲۲٪ | ۱۰/۰۸ | ۵۴٪ | ۶۳/۶۷ | ۹۰٪ | ۳۰/۵۱ | ۹۰٪ |
| ۰/۰۶ | ۵۰/۸۹ | ۲۲٪ | ۴۶/۲۴ | ۲۶٪ | ۹/۱۷ | ۶۶٪ | ۸۴/۰۷ | ۹۸٪ | ۳۹/۰۳ | ۹۲٪ |
| ۰/۰۵ | ۴۴/۶۷ | ۳۶٪ | ۴۷/۷۱ | ۳۲٪ | ۱۵/۶۴ | ۵۸٪ | ۱۳۲/۸۱ | ۹۸٪ | ۵۲/۶۲ | ۸۶٪ |
| ۰/۰۴ | ۵۲/۹۸ | ۴۸٪ | ۴۰/۷۲ | ۵۸٪ | ۱۹/۶۸ | ۶۶٪ | ۱۹۰/۸۳ | ۱۰۰٪ | ۱۱۱/۰۸ | ۸۶٪ |
| ۰/۰۳ | ۸۴/۶۶ | ۵۰٪ | ۶۷/۴۲ | ۶۰٪ | ۳۳/۷۰ | ۶۶٪ | ۳۵۲/۳۲ | ۱۰۰٪ | ۱۷۶/۸۱ | ۸۴٪ |
| ۰/۰۲ | ۱۳۸/۵۵ | ۶۴٪ | ۱۳۴/۹۳ | ۷۶٪ | ۵۲/۷۲ | ۷۸٪ | ۵۴۵/۲۳ | ۱۰۰٪ | ۲۹۶/۰۸ | ۹۰٪ |
| ۰/۰۱ | ۵۲۵/۶۹ | ۸۴٪ | ۶۱۹/۲۵ | ۸۶٪ | ۱۸۷/۴۸ | ۸۸٪ | ۷۴۹/۹۷ | ۱۰۰٪ | ۳۶۹/۵۷ | ۹۸٪ |

برای این که قضاوت در مورد نحوه عملکرد الگوریتم‌های ارائه‌شده روی گراف‌های مختلف ساده‌تر باشد، از روی مقادیر جداول مربوط به شاخص‌ها، خلاصه‌سازی صورت گرفته است. بدین صورت که در مجموع آزمایش‌های صورت گرفته برای گراف‌های Alex1-b و Alex2-b و Alex3-b به ترتیب اجراهای با بیش از ۹۰٪، ۸۰ ٪و ۷۰٪ همگرایی درنظر گرفته شده و سپس متوسط مقادیر شاخص روی این گروه از اجراها، محاسبه شده‌است. شکل ۵ مقادیر متوسط شاخص‌های AVS[1]، AVI[2]، AVT[3] و PC[4] را در الگوریتم‌های مختلف نشان می‌دهد. نتایج این شکل تأیید می‌کنند که الگوریتم Alg۴ بهترین الگوریتم از نظر دقت همگرایی (احتمال همگرایی به جواب بهینه) است. به نظر می‌رسد که الگوریتم Alg۵ در مجموع از نظر میزان دقت نسبت به زمان لازم برای رسیدن به جواب بهینه و تعداد متوسط نمونه‌گیری در هر اجرا و متوسط تکرار لازم برای همگراشدن، در مقایسه با سایر الگوریتم‌های پیشنهادی در این مقاله، الگوریتم بهتری است. هیچ حکم کلی دیگری نمی‌توان در خصوص مقایسه الگوریتم‌ها ارائه داد و عملکرد الگوریتم‌ها روی گراف‌های مختلف، متفاوت است.

یک روش دیگر برای مقایسه الگوریتم‌های مختلف ارائه‌شده در این مقاله، استفاده از شاخصی است که آن را کمترین تلاش برای حصول بهترین نتیجه نامیده‌ایم. منظور از این شاخص، برآورد متوسط حداقل تعداد تکرارها، تعداد نمونه‌گیری‌ها و متوسط زمان لازم برای رسیدن به محتمل‌ترین نتیجه است. در جدول ۱۰ نتایج به ازای الگوریتم‌های مختلف روی گراف‌های متفاوت این مقاله نشان داده شده‌است. این جدول نیز نشان می‌دهد که در مجموع الگوریتم Alg۴، بهترین الگوریتم از جمیع جهات می‌باشد و با وجود این، این حکم کلی نیست و کاملاً به گراف وابسته است.

1. Average Samples
2. Average Iteration
3. Average Time
4. Convergence Percentage




شکل ۵: نتایج مقایسه‌ای متوسط شاخص‌ها روی گراف‌های نمونه در الگوریتم‌های مختلف. مقادیر محور افقی به ترتیب الگوریتم‌های Alg۱ تا Alg۵ را نشان می‌دهند.

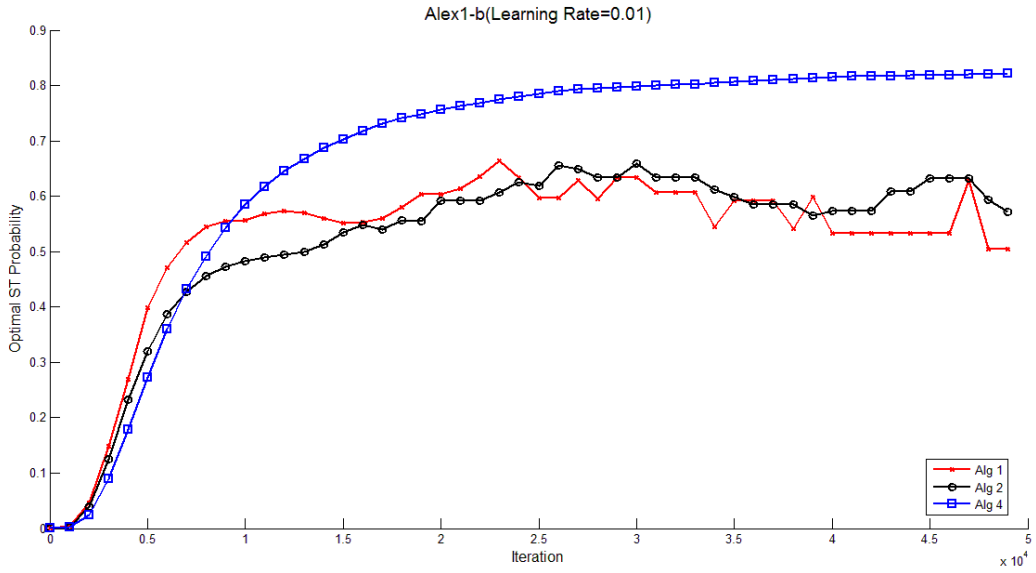
جدول ۱۰: مقادیر شاخص‌ها در مورد کمترین تلاش برای بهترین نتیجه.

| rate | PC | DST | AVI | AVS | AVT | Alg# | Graph |
|---|---|---|---|---|---|---|---|
| ۰٫۰۲ | ۱۰۰٪ | ۳۸۲ | ۷۶۴۸ | ۵۴۳۷۹ | ۵۱٫۱۱ | ۱ | |
| ۰٫۰۲ | ۱۰۰٪ | ۴۰۱ | ۷۱۳۹ | ۵۰۸۱۲ | ۴۴٫۳۲ | ۲ | |
| ۰٫۰۴ | ۱۰۰٪ | ۳۳ | ۱۹۰۰ | ۱۴۱۴۰ | ۷٫۹۴ | ۳ | Alex۱ – b |
| ۰٫۰۹ | ۱۰۰٪ | ۳۲۸ | ۶۲۰۷ | ۴۴۲۸۸ | ۳۴٫۵۸ | ۴ | |
| ۰٫۲ | ۱۰۰٪ | ۳۲ | ۹۳۹ | ۷۴۱۳ | ۳٫۹۷ | ۵ | |
| ۰٫۰۲ | ۸۲٪ | ۴۱۶ | ۹۴۴۶ | ۶۲۸۱۳ | ۸۳٫۰۲ | ۱ | |
| ۰٫۰۳ | ۸۰٪ | ۳۷۷ | ۳۹۷۴ | ۲۶۳۰۲ | ۷۳٫۹۴ | ۲ | |
| ۰٫۰۲ | ۷۲٪ | ۴۱ | ۱۱۷۳۲ | ۶۸۳۶۶ | ۵۷٫۶۳ | ۳ | Alex۲ – b |
| ۰٫۰۷ | ۱۰۰٪ | ۳۴۰ | ۵۳۱۰ | ۴۳۳۵۲ | ۳۴٫۲۴ | ۴ | |
| ۰٫۰۵ | ۱۰۰٪ | ۳۴ | ۱۰۱۲۲ | ۸۱۷۵۷ | ۴۳٫۴۲ | ۵ | |
| ۰٫۰۱ | ۸۴٪ | ۱۳۲۶ | ۲۹۷۹۵ | ۲۳۸۳۲۹ | ۵۲۵٫۶۹ | ۱ | |
| ۰٫۰۱ | ۸۶٪ | ۱۳۰۵ | ۳۲۱۰۲ | ۲۷۲۶۹۳ | ۶۱۹٫۲۵ | ۲ | |
| ۰٫۰۱ | ۸۸٪ | ۱۲۸ | ۲۰۶۸۲ | ۱۷۱۵۷۴ | ۱۸۷٫۴۸ | ۳ | Alex۳ – b |
| ۰٫۰۴ | ۱۰۰٪ | ۸۷۵ | ۱۹۰۱۳ | ۱۷۲۳۶۹ | ۱۹۰٫۸۳ | ۴ | |
| ۰٫۰۵ | ۹۸٪ | ۱۱۸ | ۴۴۱۳۴ | ۴۴۴۲۲۰ | ۳۶۹٫۵۷ | ۵ | |

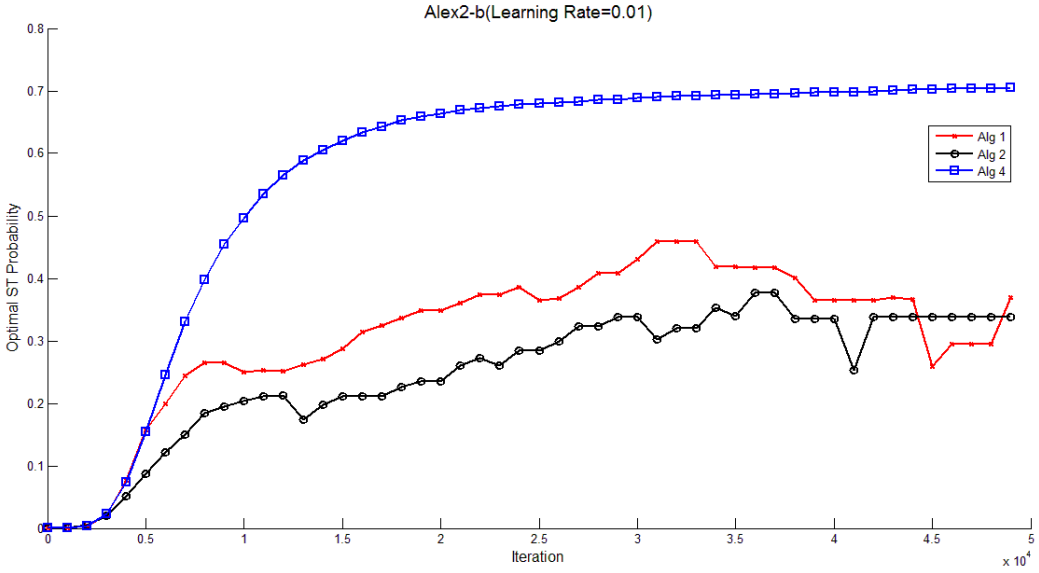
## ۵-۵ آزمایش ۲

برای مقایسه الگوریتم‌ها از یک منظر دیگر ابتدا نمودارهای "رفتار هر زمان" را برای یک الگوریتم معرفی می‌کنیم. این نمودار به این صورت ساخته می‌شود که در هر گام از تکرار الگوریتم، احتمال انتخاب درخت پوشای بهینه، محاسبه شده است. احتمال انتخاب درخت پوشای بهینه برابر با حاصل‌ضرب احتمال انتخاب اقدام‌های متناظر با یال‌های تشکیل‌دهنده درخت پوشا در گراف است. به این ترتیب مجموعه‌ای از نقاط $(x,y)$ به دست می‌آید که $x$ نشان‌دهنده گام اجرای الگوریتم و $y$ نشان‌دهنده احتمال انتخاب درخت پوشای بهینه است. مجموعه تمام این نقاط $(x,y)$ یک منحنی به دست می‌دهد که آن را منحنی "رفتار هر زمان" الگوریتم می‌نامیم. چنین نموداری علاوه بر توصیف رفتار الگوریتم در زمان، نشان‌دهنده میزان دقت جواب زیربهینه‌ای است که در هر مرحله به دست می‌آید.

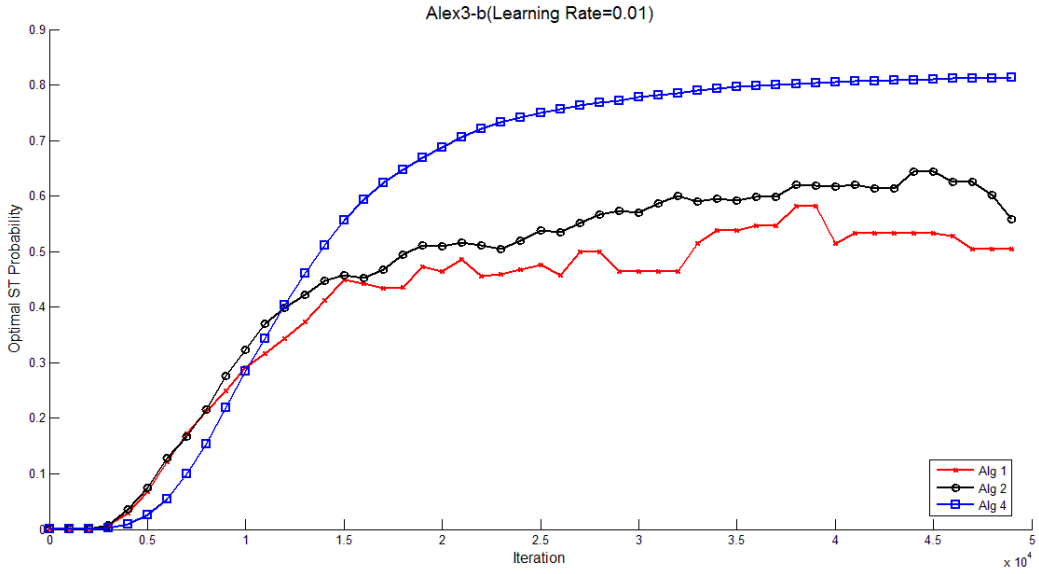
در شکل‌های ۶ تا ۸ این نمودارها برای هر ۳ الگوریتم Alg۱، Alg۲ و Alg۴ روی گراف‌های تصادفی alex۱ – b، alex۲ – b و alex۳ – b رسم شده‌اند. ساخت چنین گرافی که صرفاً بر اساس احتمال انتخاب اقدام‌های آتاماتاها صورت می‌گیرد، در ۲ الگوریتم Alg۳ و Alg۵ به دلیل دخالت‌دادن وزن متوسط یال‌ها در نحوه انتخاب اقدام‌ها، ممکن نیست. بر اساس این نمودارها می‌توان مشاهده کرد که:



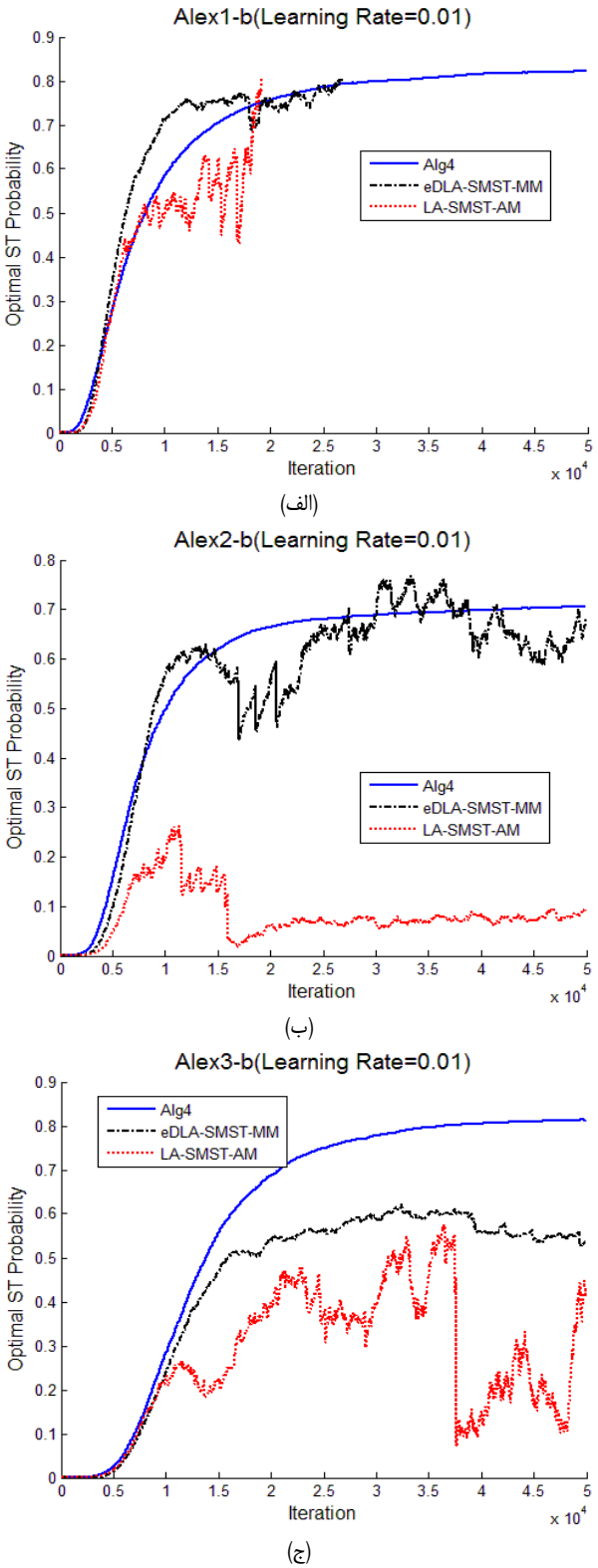

شکل ۶: نحوه تغییر احتمال انتخاب درخت پوشای بهینه در گراف Alex۱−b در الگوریتم‌های مختلف (منحنی‌های رفتار در هر زمان).




شکل ۷: نحوه تغییر احتمال انتخاب درخت پوشای بهینه در گراف Alex۲−b در الگوریتم‌های مختلف (منحنی‌های رفتار در هر زمان).




شکل ۸: نحوه تغییر احتمال انتخاب درخت پوشای بهینه در گراف Alex۳−b در الگوریتم‌های مختلف (منحنی‌های رفتار در هر زمان).




شکل ۹: عملکرد مقایسه‌ای الگوریتم‌های مختلف روی گراف‌های نمونه.

۱) هر سه شکل نشان‌دهنده رفتار مستحکم الگوریتم Alg۴ در مقایسه با دو تای دیگر است. این به دلیل استفاده از واریانس در محاسبه شاخصه مقایسه عملکرد انتخاب اقدام‌های آتاماتاها است. بنابراین همان گونه که نتایج جداول شاخص‌ها نیز تأیید می‌کنند، معیار مقایسه‌ای جدید پیشنهادی، تأثیر بسزایی در نحوه همگرایی الگوریتم و سرعت آن دارد.

۲) هر سه نمودار را می‌توان به سه قسمت تقریباً مساوی تقسیم کرد. در یک‌سوم اولیه (حدود ۱۵۰۰۰ تا ۲۰۰۰۰ دور تکرار اولیه هر یک از سه الگوریتم) احتمال انتخاب درخت پوشای بهینه در هر یک از سه الگوریتم تا عدد مشخصی تغییر می‌کند. در دوسوم بعدی علی‌رغم تکرارهای بیشتر، تغییر چندانی در نتایج مشاهده نمی‌شود و یا میزان این تغییرات در برابر تلاشی که می‌شود، بسیار کم است. علاوه بر این می‌توان مشاهده کرد که احتمال انتخاب درخت پوشای بهینه در الگوریتم Alg۴ در بخش دوم در مقایسه با الگوریتم‌های Alg۲ و Alg۱ فاصله چشم‌گیری دارد.

۳) بررسی رفتارهای این سه الگوریتم در مقایسه با رفتار دو الگوریتم دیگر، نشان‌دهنده کاربردپذیری این سه الگوریتم در مواردی است که اصولاً گراف شبکه به عنوان یک جعبه سیاه فرض می‌شود. بر اساس این فرض، الگوریتم نمی‌تواند از مکاشفه‌هایی[1] که بر مبنای وضعیت داخلی گره‌ها در گراف شبکه است برای بهبود عملکرد الگوریتم استفاده کند. به بیان دیگر، در این حالت فرض می‌شود که الگوریتم پیشنهادی برای حل مسئله به هیچ اطلاعی در خصوص مشخصه‌های وزنی مربوط به یال‌های تشکیل‌دهنده درخت پوشا دسترسی ندارد و تنها مجموع وزن یال‌های تشکیل‌دهنده درخت به عنوان وزن درخت پوشا در دسترس هستند [۶۳]. از این منظر الگوریتم Alg۴ برتری نسبی بر دو الگوریتم دیگر Alg۲ و Alg۱ دارد.

## ۵-۶ آزمایش ۳

در آزمایشی دیگر، الگوریتم Alg۴ را با الگوریتم پیشنهادشده در [۶۲] که آن را LA-SMST-AM نامیده‌ایم و نیز الگوریتم پیشنهادی در [۶۴] که آن را eDLA-SMST-MM نامیده‌ایم، مقایسه کرده‌ایم. تفاوت این سه الگوریتم در آن است که الگوریتم LA-SMST-AM به صورت کاملاً تصادفی یک آتاماتا را در میان مجموعه آتاماتاهای متناظر با گره‌های گراف تصادفی برای انتخاب یالی که بایستی نمونه‌گیری شود، انتخاب می‌کند. اما الگوریتم eDLA-SMST-MM بر اساس یک مجموعه از قواعد همسایگی این کار را انجام می‌دهد که در نتیجه آن، تعداد انتخاب‌ها محدودتر می‌شوند. هرچه گراف تصادفی کامل‌تر باشد، این دو روش به یکدیگر نزدیک‌تر می‌شوند، به گونه‌ای که در گراف کامل، این دو روش با یکدیگر تفاوتی ندارند. الگوریتم Alg۴ که در این مقاله پیشنهاد شده است، نحوه فعال‌کردن آتاماتاها را بر اساس جایگشتی که توسط DLA پیشنهاد داده است انجام می‌دهد. علاوه بر این، معیار مقایسه در الگوریتم Alg۴ و eDLA-SMST-MM مبتنی بر واریانس است اما در LA-SMST-AM این معیار مقدار میانگین وزنی وزن درخت‌های قبلی به دست آمده بدون لحاظ‌کردن واریانس است. نتایج این سه مقایسه را در شکل ۹ مشاهده می‌کنید.

همان گونه که شکل ۹ نشان می‌دهد، الگوریتم پیشنهادی در این مقاله در مقایسه با الگوریتم‌های پیشین مبتنی بر آتاماتای یادگیر، پایدارتر است. ضمناً با توجه به این که تفاوت Alg۴ و eDLA-SMST-MM در نحوه انتخاب آتاماتای بعدی است، می‌توان نتیجه گرفت که روش پیشنهادی در این مقاله تأثیر زیادی در پایدارشدن الگوریتم و رفتار یکنوای آن در میل به جواب بهینه دارد. در خصوص الگوریتم LA-SMST-AM با توجه به این که هم نحوه انتخاب آتاماتاها متفاوت است و هم نحوه پاداش‌دهی، نتیجه‌گیری خاصی نمی‌توان کرد. تنها، عملکرد ضعیف نسبی آن در مقایسه با سایر الگوریتم‌ها قابل مشاهده است.

1. Heuristics

## ٦- نتیجه‌گیری

در این مقاله، ابتدا بهینه‌سازی جایگشت تصادفی به عنوان موردی خاص از بهینه‌سازی که در آن تابع هزینه تابعی تصادفی با توزیع نامعلوم است، معرفی گردید. در ادامه یک الگوریتم مبتنی بر آتاماتای یادگیر توزیع‌شده برای حل این مسئله ارائه شد. همگرایی آن مورد بررسی قرار گرفت و نشان داده شد که با انتخاب مقادیر مناسب برای نرخ یادگیری، می‌توان احتمال انتخاب جواب بهینه را تا حد لازم به ۱۰۰٪ نزدیک کرد. در ادامه، مسئله یافتن درخت پوشا با کمترین وزن به یک مسئله بهینه‌سازی جایگشت تصادفی نگاشت شد. الگوریتم‌های متعدد بهبودیافته برای آن ارائه گردید و برتری Alg۳ از نظر نحوه همگرایی و زیربهینه بودن جواب‌ها در هر لحظه بر سایر الگوریتم‌های ارائه‌شده در این مقاله و دیگر الگوریتم‌هایی که تا کنون برای حل این مسئله ارائه گردیده است، نشان داده شد.

## مراجع


[1] D. P. Williamson and D. B. Shmoys, *The Design of Approximation Algorithms*, Cambridge University Press, 2011.

[2] V. V Vazirani, *Approximation Algorithms*, Berlin: springer, 2001.

[3] E. G. Talbi, *Metaheuristics: from Design to Implementation*, vol. 74, John Wiley & Sons, 2009.

[4] C. C. Sims, "Graphs and finite permutation groups," *Math. Zeitschrift*, vol. 95, no. 1, pp. 76-86, Feb. 1967.

[5] G. Cooperman and E. Robinson, "Memory-based and disk-based algorithms for very high degree permutation groups," in *Proc. of the Int. Symp. on Symbolic and Algebraic Computation*, pp. 66-73, Philadelphia, PA, USA, 3-6 Aug. 2003.

[6] J. Grabowski and J. Pempera, "The permutation flow shop problem with blocking. A tabu search approach," *Omega*, vol. 35, no. 3, pp. 302-311, Jun. 2007.

[7] J. Kaabi and Y. Harrath, "Permutation rules and genetic algorithm to solve the traveling salesman problem," *Arab J. Basic Appl. Sci.*, vol. 26, no. 1, pp. 283-291, Jan. 2019.

[8] H. Ishii, S. Shiode, T. Nishida, and Y. Namasuya, "Stochastic spanning tree problem," *Discret. Appl. Math.*, vol. 3, no. 1, pp. 263-273, Nov. 1981.

[9] S. A. Dehkordi, *et al*., "A survey on data aggregation techniques in IoT sensor networks," *Wirel. Networks*, vol. 26, no. 2, pp. 1243-1263, Feb. 2020.

[10] C. Buchheim and M. Jünger, "Linear optimization over permutation groups," *Discret. Optim.*, vol. 2, no. 4, pp. 308-319, Dec. 2005.

[11] M. A. L. Thathachar and P. S. Sastry, *Networks of Learning Automata: Techniques for Online Stochastic Optimization*, Springer Science & Business Media, 2003.

[12] M. A. L. Thathachar and P. S. Sastry, "Varieties of learning automata: an overview," *IEEE Trans. Syst. Man, Cybern. Part B*, vol. 32, no. 6, pp. 711-722, Dec. 2002.

[13] M. L. Tsetlin, *Automaton Ttheory and Modeling of Biological Systems*, vol. 102, pp.160-196, Academic Press New York, 1973.

[14] Q. Sang, Z. Lin, and S. T. Acton, "Learning automata for image segmentation," *Pattern Recognit. Lett.*, vol. 74, pp. 46-52, Apr. 2016.

[15] J. A. Torkestani and M. R. Meybodi, "Mobility-based multicast routing algorithm for wireless mobile ad-hoc networks: a learning automata approach," *Comput. Commun.*, vol. 33, no. 6, pp. 721-735, Apr. 2010.

[16] M. Shojafar, S. Abolfazli, H. Mostafaei, and M. Singhal, "Improving channel assignment in multi-radio wireless mesh networks with learning automata," *Wirel. Pers. Commun.*, vol. 82, no. 1, pp. 61-80, May 2015.

[17] M. Fahimi and A. Ghasemi, "A distributed learning automata scheme for spectrum management in self-organized cognitive radio network," *IEEE Trans. Mob. Comput.*, vol. 16, no. 6, pp. 1490-1501, Aug. 2016.

[18] H. Mostafaei, "Stochastic barrier coverage in wireless sensor networks based on distributed learning automata," *Comput. Commun.*, vol. 55, pp. 51-61, Jan. 2015.

[19] H. Mostafaei and M. R. Meybodi, "Maximizing lifetime of target coverage in wireless sensor networks using learning automata," *Wirel. Pers. Commun.*, vol. 71, no. 2, pp. 1461-1477, Jul. 2013.

[20] H. Mostafaei, A. Montieri, V. Persico, and A. Pescapé, "A sleep scheduling approach based on learning automata for WSN partialcoverage," *J. Netw. Comput. Appl.*, vol. 80, pp. 67-78, Feb. 2017.

[21] S. Misra, P. V. Krishna, K. Kalaiselvan, V. Saritha, and M. S. Obaidat, "Learning automata-based QoS framework for cloud IaaS," *IEEE Trans. Netw. Serv. Manag.*, vol. 11, no. 1, pp. 15-24, Feb. 2014.

[22] P. V. Krishna, S. Misra, D. Nagaraju, V. Saritha, and M. S. Obaidat, "Learning automata based decision making algorithm for task offloading in mobile cloud," in *Proc. IEEE Int. Conf. Comput. Inf. Telecommun. Syst.*, 6 pp., Kunming, China, 6-8 Jul. 2016.

[23] A. A. Rahmanian, M. Ghobaei-Arani, and S. Tofighy, "A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment," *Futur. Gener. Comput. Syst.*, vol. 79, pp. 54-71, Feb. 2018.

[24] A. Rezvanian and M. R. Meybodi, "A new learning automata-based sampling algorithm for social networks," *Int. J. Commun. Syst.*, vol. 30, no. 5, pp. 1-21, Mar. 2017.

[25] J. A. Torkestani and M. R. Meybodi, "Solving the minimum spanning tree problem in stochastic graphs using learning automata," in *Proc. Int. Conf. on Information Management and Engineering*, pp. 643-647, Kuala Lumpur, Malaysia, 3-5 Apr. 2009.

[26] M. R. Meybodi and H. Beigy, "Solving stochastic shortest path problem using Monte Carlo sampling method: a distributed learning automata approach," *Neural Networks and Soft Computing, Springer*, pp. 626-631, 2003.

[27] A. Rezvanian and M. R. Meybodi, "Finding minimum vertex covering in stochastic graphs: a learning automata approach," *Cybern. Syst.*, vol. 46, no. 8, pp. 698-727, Nov. 2015.

[28] M. R. Mollakhalili Meybodi and M. R. Meybodi, "Extended distributed learning automata," *Appl. Intell.*, vol. 41, no. 3, pp. 923-940, Oct. 2014.

[29] J. A. Torkestani and M. R. Meybodi, "A new vertex coloring algorithm based on variable action-set learning automata," *Comput. Informatics*, vol. 29, no. 3, pp. 447-466, 2010.

[30] M. R. Mirsaleh and M. R. Meybodi, "A new memetic algorithm based on cellular learning automata for solving the vertex coloring problem," *Memetic Comput.*, vol. 8, no. 3, pp. 211-222, Sept. 2016.

[31] J. A. Torkestani and M. R. Meybodi, "A learning automata-based heuristic algorithm for solving the minimum spanning tree problem in stochastic graphs," *J. Supercomput.*, vol. 59, no. 2, pp. 1035-1054, Feb. 2012.

[32] M. Alipour, "A learning automata based algorithm for solving traveling salesman problem improved by frequency-based pruning," *Environment*, vol. 46, no. 17, pp. 7-13, Jan. 2012.

[33] M. Hasanzadeh-Mofrad and A. Rezvanian, "Learning automata clustering," *J. Comput. Sci.*, vol. 24, pp. 379-388, Jan. 2018.

[34] A. Rezvanian, A. M. Saghiri, S. M. Vahidipour, M. Esnaashari, and M. R. Meybodi, *Recent Advances in Learning Automata*, vol. 754, Springer Nature, 2018.

[35] M. A. L. Harita and B. Thathachar, "Learning automata with changing number of actions," *IEEE Trans. Syst. Man, Cybern.-Part A Syst. Humans*, vol. 17, no. 6, pp. 1095-1100, Nov./Dec. 1987.

[36] H. Beigy and M. Meybodi, Intelligent Channel Assignment in Cellular Networks: A Learning Automata Approach, Amirkabir University of Technology, 2004.

[37] M. Thathachar and K. Ramakrishnan, "A hierarchical system of learning automata," *IEEE Trans. Syst. Man. Cybern.*, vol. 11, no. 3, pp. 236-241, Mar. 1981.

[38] H. Beigy and M. R. Meybodi, "Utilizing distributed learning automata to solve stochastic shortest path problems," *Int. J. Uncertainty, Fuzziness Knowledge-Based Syst.*, vol. 14, no. 5, pp. 591-615, Oct. 2006.

[39] F. Norman, "On the linear model with two absorbing," *J. Math. Psychol.*, vol. 5, no. 2, pp. 225-241, Jun. 1968.

[40] S. Lakshmivarahan and M. Thathachar, "Bounds on the convergence probabilities of learning automata," *IEEE Trans. Syst. Man, Cybern.-Part A Syst. Humans*, vol. 6, no. 11, pp. 756-763, Jun. 1976.

[41] J. Nešetřil, E. Milková, and H. Nešetřilová, "Otakar Borůvka on minimum spanning tree problem translation of both the 1926 papers, comments, history," *Discrete Math.*, vol. 233, no. 1-3, pp. 3-36, Apr. 2001.

[42] R. C. Prim, "Shortest connection networks and some generalizations," *Bell Syst. Tech. J.*, vol. 36, no. 6, pp. 1389-1401, Nov. 1957.

[43] J. B. Kruskal, "On the shortest spanning sub tree of a graph and the traveling salesman problem," *American Mathematical Society*, vol. 17, no. 1, pp. 48-50, Feb. 1956.

[44] R. L. Graham and P. Hell, "On the history of the minimum spanning tree problem," *IEEE Ann. Hist. Comput.*, vol. 7, no. 1, pp. 43-57, Jan. 1985.
[45] M. Mares, "Two linear time algorithms for MST on minor closed graph classes," *Arch. Math.*, vol. 40, pp. 315-320, 2004.
[46] M. Khan, V. S. A. Kumar, G. Pandurangan, and G. Pei, "A fast distributed approximation algorithm for minimum spanning trees in the SINR model," in *Proc. of the 26th Int. Conf. on Distributed Computing*, *DISC'12,* pp. 409-410, Salvador, Brazil, 16-18 Oct. 2012.
[47] D. R. Karger, P. N. Klein, and R. E. Tarjan, "A randomized linear-time algorithm to find minimum spanning trees," *J. ACM*, vol. 42, no. 2, pp. 321-328, Mar. 1995.
[48] K. W. Chong, Y. Han, and T. W. Lam, "Concurrent threads and optimal parallel minimum spanning trees algorithm," *J. ACM*, vol. 48, no. 2, pp. 297-323, Mar. 2001.
[49] J. Garay, S. Kutten, and D. Peleg, "A sublinear time distributed algorithm for minimum-weight-spanning tree," *SIAM J. Comput.*, vol. 27, no. 1, pp. 302-326, Feb. 1998.
[50] M. Khan and G. Pandurangan, "A fast distributed approximation algorithm for minimum spanning trees," *Distributed Computing,* vol. 20, no.6, pp. 391-402, Apr. 2008.
[51] H. Ohlsson, O. Gustafsson, and L. Wanhammar, "Implementation of low complexity FIR filters using a minimum spanning tree," in *Proc. of the 12th IEEE Mediterranean Electrotechnical Conf.,* pp. 261-264, Dubrovnik, Croatia, 12- 15 May 2004.
[52] Y. Xu, V. Olman, and D. Xu, "Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees," *Bioinformatics*, vol. 18, no. 4, pp. 536-545, Apr. 2002.
[53] N. Päivinen, "Clustering with a minimum spanning tree of scale-free-like structure," *Pattern Recognit. Lett.*, vol. 26, no. 7, pp. 921-930, May 2005.
[54] T. Asano, B. Bhattacharya, M. Keil, and F. Yao, "Clustering algorithms based on minimum and maximum spanning trees," in *Proc. of the 4th Annual Symp. on Computational Geometry, SCG'88*, pp. 252-257, Urbana-Champaign, IL, USA, 6-8 Jun. 1988.
[55] B. Ma, A. Hero, J. Gorman, and O. Michel, "Image registration with minimum spanning tree algorithm," in *Proc. Int. Conf. on Image*, vol. 1, pp. 481-484, Vancouver, Canada, 10-13 Sept. 2000.
[56] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," *Operations Research Forum*, vol. 3, no. 1, 4 pp., Mar. 2022.
[57] P. H. A. Sneath, "The application of computers to taxonomy," *J. Gen. Microbiol.*, vol. 17, no. 1, pp. 201-226, Aug. 1957.
[58] K. J. Supowit, D. A. Plaisted, and E. M. Reingold, "Heuristics for weighted perfect matching," in *Proc. of the 12th Annual ACM Symp. on Theory of Computing, STOC'80*, pp. 398-419, Los Angeles, CA, USA, 28- 30 Apr. 1980.
[59] H. Lshii and T. Nishida, "Stochastic bottleneck spanning tree problem," *Networks*, vol. 13, no. 3, pp. 443-449, Sept. 1983.
[60] I. B. Mohd, "Interval elimination method for stochastic spanning tree problem," *Appl. Math. Comput.*, vol. 66, no. 2-3, pp. 325-341, Dec. 1994.
[۶۱] م. قربعلی‌پور درو و م. ر. میبدی، "یافتن درخت پوشای مینیمم در گرافهای تصادفی با استفاده از اتوماتاهای یادگیر،" *مجموعه مقالات چهاردهمین کنفرانس سالانه انجمن کامپیوتر ایران*، ۷ صص.، تهران، ۲۱-۲۰ اسفند ۱۳۸۷.
[62] J. Akbari Torkestani and M. R. Meybodi, "Learning automata-based algorithms for solving stochastic minimum spanning tree problem," *Appl. Soft Comput. J.*, vol. 11, no. 6, pp. 4064-4077, Sept. 2011.
[63] K. Liu and Q. Qing Zhao, Stochastic Online Learning for Network Optimization under Random Unknown Weights, pp. 1-18, 2012.
[64] M. R. M. Meybodi and M. R. Meybodi, Extended Distributed Learning Automata: A New Method for Solving Stochastic Graph Optimization Problems, arXiv:1308.2772, p. 37, Aug. 2013.
[65] K. R. Hutson and D. R. Shier, *Online Supplement to 'Minimum Spanning Trees in Networks with Varying Edge Weights'*, pp. 1-8, retived on 29 Oct. 2020 from http://www.math.clemson.edu/~shierd/Shier/appendix2.pdf.

**محمدرضا ملاخلیلی میبدی** تحصیلات خود را در مقاطع کارشناسی و کارشناسی ارشد مهندسی کامپیوتر به ترتیب در سال‌های ۱۳۸۰ و ۱۳۸۲ از دانشگاه‌های شهید بهشتی و صنعتی امیرکبیر تهران و دکترای مهندسی کامپیوتر را در سال ۱۳۹۳ در واحد علوم و تحقیقات دانشگاه آزاد به پایان رسانده است. وی هم‌اکنون عضو هیأت‌علمی دانشگاه آزاد اسلامی واحد میبد است. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: شبکه‌های کامپیوتری و وب، شبکه‌های مبتنی بر نرم‌افزار، محاسبات نرم و کاربردهای آن، یادگیری والگوریتم‌ها، گراف‌های تصادفی و شبکه‌های پیچیده.

**معصومه زجاجی** در سال ۱۳۸۴ مدرک کارشناسی مهندسی کامپیوتر خود را از دانشگاه آزاد اسلامی واحد همدان و در سال ۱۳۹۰ مدرک کارشناسی ارشد مهندسی کامپیوتر خود را از دانشگاه آزاد اسلامی واحد قزوین دریافت نمود. در سال ۱۳۹۳ به دوره دکترای مهندسی کامپیوتر در دانشگاه آزاد اسلامی واحد میبد وارد گردید و در سال ۱۳۹۹ موفق به اخذ درجه دکترا در مهندسی کامپیوتر از دانشگاه مذکور گردید. زمینه‌های علمی مورد علاقه نام‌برده شامل موضوعاتی مانند شبکه‌های حسگر بی‌سیم، علم داده، یادگیری ماشین می‌باشد.