دهمین
کنفرانس سالانه انجمن کامپیوتر ایران
۲۷ تا ۲۹ بهمن ۱۳۸۳

تطبیق پارامترهای الگوریتم کلونی مورچه‌ها با استفاده از اتوماتاهای یادگیر[1]

فردین ابدالی محمدی محمد رضا میبدی
آزمایشگاه محاسبات نرم
دانشکده مهندسی کامپی‌وتر و فناوری اطلاعات
دانشگاه صنعتی امیرکبیر
تهران ایران

چکیده: الگوریتم‌های کلونی مورچه‌ها[2] یک گروه از الگوریتم‌های بهینه‌سازی می‌باشند که از کاوش مورچه‌ها برای غذا در طبیعت الهام گرفته‌اند. در این الگوریتم‌ها تعدادی عامل که همان مورچه‌ها می‌باشند به منظور یافتن راه حل مناسب فضای راه حل‌ها را جستجو می‌کنند. الگوریتم‌های کلونی مورچه‌ها دارای پارامترهای متعددی از جمله اهمیت نسبی فرمون روی مسیرها و ضریب تبخیر فرمون در هر مرحله می‌باشند که همگرایی و کارایی الگوریتم‌ها تا حد زیادی به آنها وابسته است. تا به حال مقدار مطلوب پارامترها با توجه به نوع مساله از طریق آزمایش و خطا تعیین می‌گردیده است. در این مقاله روشی مبتنی بر اتوماتاهای یادگیر برای تطبیق پارامترهای الگوریتم "سیستم کلونی مورچه‌ها"[3] ارایه می‌گردد و از طریق آزمایشهای مختلف کارایی روش پیشنهادی مورد بررسی قرار می‌گیرد و با تنها روش گزارش شده برای این منظور که مبتنی بر الگوریتم‌های ژنتیکی می‌باشد مقایسه می‌گردد.

کلمات کلیدی: کلونی مورچه‌ها، تطبیق پارامتر، اتوماتاهای یادگیر

۱- مقدمه

مورچه‌های طبیعی در هنگام یافتن غذا و برگشت به طرف لانه مقداری فرمون[4] روی مسیری که پیمایش می‌کنند قرار می‌دهند. این عمل به سایر مورچه‌ها کمک می‌کند تا مکان منابع غذایی را شناسایی کنند. فرمون به جای گذاشته شده بر روی زمین به آرامی تبخیر می‌شود. مسایل کلونی مورچه‌ها معمولا توسط یک گراف مدل می‌شوند. مسیرها توسط یالهای گراف و

---

[1] قسمتی از این کار تحقیقاتی با حمایت مالی مرکز تحقیقات مخابرات ایران انجام گرفته است.

[2] Ant Colony
[3] Ant Colony System
[4] Pheromone

مقدار فرمون هر یال وزن آن یال می‌باشد. هر مورچه سعی می‌کند با شروع از یک گره آغازین از طریق پیمایش گراف کوتاهترین دور را در گراف پیدا کند. مورچه‌ها در هر گره بر اساس یک احتمال (احتمال انتخاب مسیر) " گرهٔ بعدی" را انتخاب کرده و بر اساس فرمول "بهنگام‌سازی فرمون" مقدار فرمون یال‌ها را تغییر می‌دهند. با این مکانیزم هر مورچه سعی می‌کند کوتاهترین دور را در گراف پیدا کند.

در الگوریتم‌های کلونی مورچه‌ها هر مورچه در ابتدا در یک گره بصورت تصادفی قرار داده می‌شود و دارای یک حافظه است که راه حل‌های جزیی که تا به حال بدست آمده‌اند را در خود حفظ می‌کند. با شروع از گره آغازین، هر مورچه از یک گره به گرهٔ دیگر حرکت می‌کند. در گرهٔ i، مورچهٔ شمارهٔ k، گره بعدی j را که توسط آن مورچه پیموده نشده است با احتمالی که با استفاده از تابع احتمال انتخاب مسیر محاسبه می‌شود انتخاب می‌کند. بعد از اینکه هر مورچه یک دور کامل تولید نمود مقادیر فرمون یال‌های دور تولید شده بهنگام می‌شود. در کلونی مورچه‌ها عمل بهنگام‌سازی مقادیر فرمون ابتدا با کاهش مقدار فرمون با یک فاکتور ثابت (ضریب تبخیر) و سپس قرار دادن مقدار مشخصی از فرمون توسط هر مورچه روی یال‌های دور پیموده شده انجام می‌شود. ضریب تبخیر که مقداری بین صفر و یک دارد برای اجتناب از انباشتن بی‌حد دنبالهٔ فرمون است و الگوریتم را قادر می‌سازد تا تصمیم‌های بدی که قبلاً گرفته شده را فراموش کند. الگوریتم به مرور مورچه‌های بیشتری را در طول مسیرهای کوتاهتر هدایت می‌کند و در نتیجه مقدار فرمون قرار داده شده روی این مسیرها بالا خواهد رفت که این باعث افزایش احتمال انتخاب مسیرهای کوتاهتر در آینده خواهد شد.

الگوریتم‌های کلونی مورچه‌ها دارای پارامترهای متعددی از جمله اهمیت نسبی فرمون روی مسیرها و ضریب تبخیر فرمون می‌باشند که همگرایی و کارایی الگوریتم‌ها به مقدار زیادی به آنها وابسته است. معمولاً مقدار مطلوب این پارامترها با توجه به نوع مساله از طریق آزمایش و خطا تعیین می‌گردد. در [3] مطالعاتی درباره اثرات پارامترهای الگوریتم کلونی مورچه‌ها انجام گرفته است. در این مقاله با آزمایش‌های مختلف به روش آزمایش و خطا نشان داده شده است که پارامترهای الگوریتم‌های کلونی مورچه‌ها در بازه‌های خاصی نتایج بهتری تولید می‌کنند. تنها تلاشی که برای تطبیق پارامترهای الگوریتم سیستم کلونی مورچه‌ها که یکی از بهترین الگوریتم‌های کلونی مورچه می‌باشد انجام گرفته است توسط مارسین[5] و تونی[6] در سال ۲۰۰۲ بوده است [4]. مارسین و تونی پارامترهای الگوریتم ACS را با استفاده از الگوریتم‌های ژنتیکی تنظیم کرده‌اند و نشان داده‌اند که این روش می‌تواند کارایی الگوریتم‌های کلونی مورچه‌ها را به مقدار زیادی بهبود بخشد. الگوریتم ارایه شده توسط آنها Meta ACS نامیده می‌شوند. در این مقاله با استفاده از اتوماتاهای یادگیر پارامترهای الگوریتم "سیستم کلونی مورچه‌ها" (ACS) بطور تطبیقی تعیین می‌گردد. الگوریتم ACS یکی از الگوریتم‌های کلونی مورچه‌ها است که نسبت به سایر الگوریتم‌های کلونی مورچه‌ها نتایج بهتری بدست آورده است. روش پیشنهادی با الگوریتم Meta ACS مقایسه شده است. نتایج شبیه‌سازی‌ها برتری الگوریتم پیشنهادی را نشان می‌دهد. قبلا اتوماتاهای یادگیر برای تطبیق پارامترهای شبکه‌های عصبی مورد استفاده قرار گرفته است[9][10][11][12][13].

ادامه مقاله بصورت زیر سازماندهی شده است. در بخش ۲ الگوریتم ACS و در بخش ۳ اتوماتای یادگیر و انواع آن معرفی می‌گردد. در بخش ٤ الگوریتم پیشنهادی برای تطبیق پارامترهای الگوریتم ACS و نتایج شبیه‌سازی‌ها ارایه می‌شود. بخش پایانی نتیجه گیری می‌باشد.

۲- الگوریتم ACS

[5] Marcin
[6] Tony

الگوریتم ACS یکی از بهترین الگوریتم‌های کلونی مورچه‌ها است که نسبت به سایر الگوریتم‌های کلونی مورچه‌ها نتایج بهتری تولید می‌کند[2]. تفاوت الگوریتم‌های مختلف کلونی مورچه‌ها در نحوهٔ انتخاب گرهٔ بعدی و روش بهنگام سازی فرمون توسط مورچه‌ها است. در الگوریتم ACS نه تنها مورچه‌ها هنگام عبور از هر یال فرمون آنرا بهنگام می‌کنند بلکه در پایان هر تکرار مورچه‌ای که بهترین دور را تولید کرده نیز یک بار دیگر فرمون مسیری که پیموده را بهنگام می‌نماید. تفاوت دیگری که الگوریتم ACS با دیگر الگوریتم‌های کلونی مورچه‌ها دارد اینست که مورچه‌ها برای انتخاب گرهٔ بعدی از ۲ فرمول استفاده می‌کنند که انتخاب یکی از این فرمول‌ها توسط یک متغیر تصادفی مشخص می‌گردد. یعنی مورچه‌ای که در گرهٔ r قرار دارد، گرهٔ s را مطابق فرمول زیر انتخاب می‌کند :

$$s = \begin{cases} \max_{j \in N_i^k} \{\tau_{ij}(t).\eta_{ij}^{\beta}\} & \text{if } q \prec Q_0 \\ \textit{choose according to AS equation} & \text{otherwise} \end{cases} \qquad (1)$$

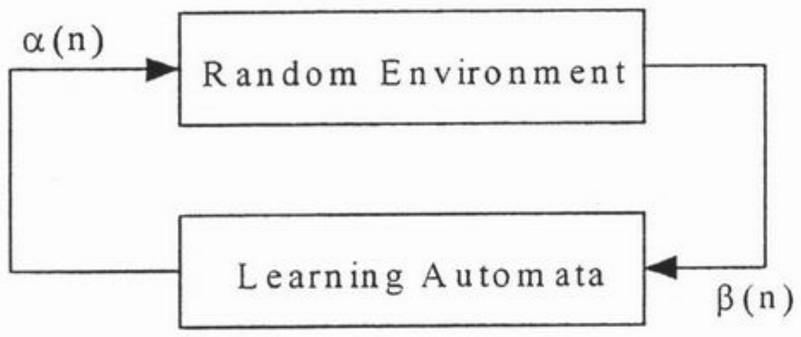
که N_i^k همسایه‌های ملاقات نشده مورچهٔ kام است، $\tau_{ij}(t)$ مقدار فرمون روی یال (i,j) در تکرار t و η_{ij} فاصلهٔ بین گرهٔ i تا گرهٔ j می‌باشد. q یک متغیر تصادفی با توزیع یکنواخت است. Q_0 یک مقدار آستانه‌ای است که در ابتدای الگوریتم مقدار دهی می‌شود. هنگامی که مقدار Q_0 نزدیک به ۱ باشد، استخراج اطلاعات بر اکتشاف اطلاعات جدید برتری داده می‌شود و اگر مقدار آن برابر صفر قرار داده شود قانون انتخاب گرهٔ بعدی مانند قانون انتخاب مسیر در AS می‌باشد[2]. همچنین در الگوریتم ACS در هر تکرار مورچه‌ای که بهترین دور را تولید کرده است با استفاده از فرمول زیر مقدار فرمون مسیرش را بهنگام می‌کند:

$$\tau_{ij}(t+1) = (1-\rho)\cdot\tau_{ij}(t) + \rho\Delta\tau_{ij}^{best}(t) \qquad (2)$$

که $\Delta\tau_{ij}^{best}(t)$ مقدار فرمونی است که بهترین مورچه در تکرار t بر روی یال (i,j) به جای می‌گذارد. بهترین مورچه می‌تواند بهترین مورچهٔ تکرار جاری و یا بهترین مورچه از آغاز الگوریتم تا تکرار جاری باشد.

۲- اتوماتای یادگیر[7]

اتوماتای یادگیر یک مدل انتزاعی است که تعداد محدودی عمل را می تواند انجام دهد. هر عمل انتخاب شده توسط محیطی احتمالی ارزیابی شده و پاسخی به اتوماتای یادگیر داده می شود. اتوماتای یادگیر از این پاسخ استفاده نموده وعمل خود را برای مرحله بعد انتخاب می کند. شکل ۱ ارتباط بین اتوماتای یادگیر و محیط را نشان می دهد.



شکل ۱: ارتباط بین اتوماتای یادگیر و محیط

محیط[8]: محیط را می توان توسط سه تایی $E \equiv \{\alpha, \beta, c\}$ نشان داد که در آن $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه ورودیها، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_m\}$ مجموعه خروجیها و $c \equiv \{c_1, c_2, ..., c_r\}$ مجموعه احتمالهای جریمه می باشد. هر گاه β مجموعه دو عضوی باشد ، محیط از نوع P می باشد. در چنین محیطی $\beta_1 = 1$ به عنوان جریمه و $\beta_2 = 0$ به عنوان پاداش در

[7] Learning Automata
[8] Environment

نظر گرفته می شود. در محیط از نوع Q ، β(n) می تواند به طور گسسته یک مقدار از مقادیر محدود در فاصله [0,1] و در محیط از نوع S، β(n) متغیر تصادفی در فاصله [0,1] است. c_i احتمال اینکه عمل α_i نتیجه نامطلوب داشته باشد می باشد. در محیط ایستا[9] مقادیر c_i بدون تغییر می مانند، حال آنکه در محیط غیر ایستا[10] این مقادیر در طی زمان تغییر می کنند. اتوماتاهای یادگیربه دو گروه با ساختار ثابت و با ساختار متغیر تقسیم بندی میگردند. در ادامه به شرح مختصری درباره اتوماتای یادگیر با ساختار متغییر می پردازیم.

اتوماتای یادگیر با ساختار متغیر[11]: اتوماتای یادگیر با ساختار متغیر توسط ۴ تائی $\{\alpha, \beta, p, T\}$ نشان داده می شود که در آن $\alpha \equiv \{\alpha_1, \alpha_2, \cdots, \alpha_r\}$ مجموعه عملهای اتوماتا، $\beta \equiv \{\beta_1, \beta_2, \cdots, \beta_m\}$ مجموعه ورودیهای اتوماتا، $p \equiv \{p_1, p_2, \cdots, p_r\}$ بردار احتمال انتخاب هر یک از عملها، و $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ الگوریتم یادگیری می باشد. در این نوع از اتوماتاها، اگر عمل α_i در مرحله n ام انتخاب شود و پاسخ مطلوب از محیط دریافت نماید، احتمال $p_i(n)$ افزایش یافته و سایر احتمالها کاهش می یابند. و برای پاسخ نامطلوب احتمال $p_i(n)$ کاهش یافته و سایر احتمالها افزایش می یابند. در هر حال، تغییرات به گونه ای صورت می گیرد تا حاصل جمع $p_i(n)$ ها همواره ثابت و مساوی یک باقی بماند. الگوریتم زیر یک نمونه از الگوریتمهای یادگیری خطی برای اتوماتای یادگیر با ساختار ثابت است.

الف- پاسخ مطلوب

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

ب- پاسخ نامطلوب

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

در روابط فوق، پارامتر پاداش و a پارامتر پاداش و b پارامتر جریمه می باشد. با توجه به مقادیر a و b سه حالت را می توان در نظر گرفت. زمانیکه a و b با هم برابر باشند، الگوریتم ر، L_{RP} [12] می نامیم. زمانیکه b از a خیلی کوچکتر باشد، الگوریتم را $L_{R\varepsilon P}$ [13] می نامیم. زمانیکه b مساوی صفر باشد، الگوریتم را L_{RI} [14] می نامیم. برای اطلاعات بیشتر در باره اتوماتاهای یادگیر با ساختار متغیر و اتوماتاهای یادگیر با ساختار ثابت مانند $L_{2N,2}$ ، $G_{2N,2}$ ، ***Krinsky*** و ***Krylov*** که در این مقاله از آنها استفاده شده است می توان به [14] مراجعه کرد.

٤- تطبیق پارامترهای ACS با استفاده از اتوماتای یادگیر

در این بخش روشی مبتنی بر اتوماتاهای یادگیر برای تطبیق پارامترهای β، ρ و Q_0 الگوریتم ACS ارایه می گردد. این پارامترها در تصمیم گیری مورچه‌ها و بهنگام‌سازی فرمون دخیل هستند. پارامتر β اهمیت نسبی فرمون، ρ ضریب تبخیر و Q_0 فرمول

[9] Stationary
[10] Non-Stationary
[11] Variable Learning Automata
[12] Linear Reward Pealty
[13] Linear Reward Epsilon Penalty
[14] Linear Reward Inaction

انتخاب مسیر را مشخص می‌کند. بر خلاف الگوریتم ACS که تمام مورچه‌ها پارامترهای یکسانی دارند در الگوریتم پیشنهادی که آنرا ACSLA می‌نامیم هر مورچه پارامترهای خود را دارد. در الگوریتم پیشنهادی هر مورچه به ۳ اتوماتای یادگیر مجهز شده است که هر کدام از آنها وظیفهٔ تطبیق یکی از پارامترها را بر عهده دارد و مورچه برای تصمیم‌گیری در انتخاب گره بعدی و بهنگام سازی فرمون از آنها استفاده می‌کند با توجه با نتایج ارایه شده در [3] برای هر پارامتر مجموعه‌ای از اعداد تعریف کرده‌ایم که آن پارامتر می‌تواند اختیار کند. مجموعهٔ اعداد مجاز هر پارامتر بصورت زیر تعریف شده‌اند :

$\beta \in \{1, 2.5, 4, 5.5, 7, 8.5, 10, 11.5, 13, 14.5\}$

$\rho \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$

$Q_0 \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$

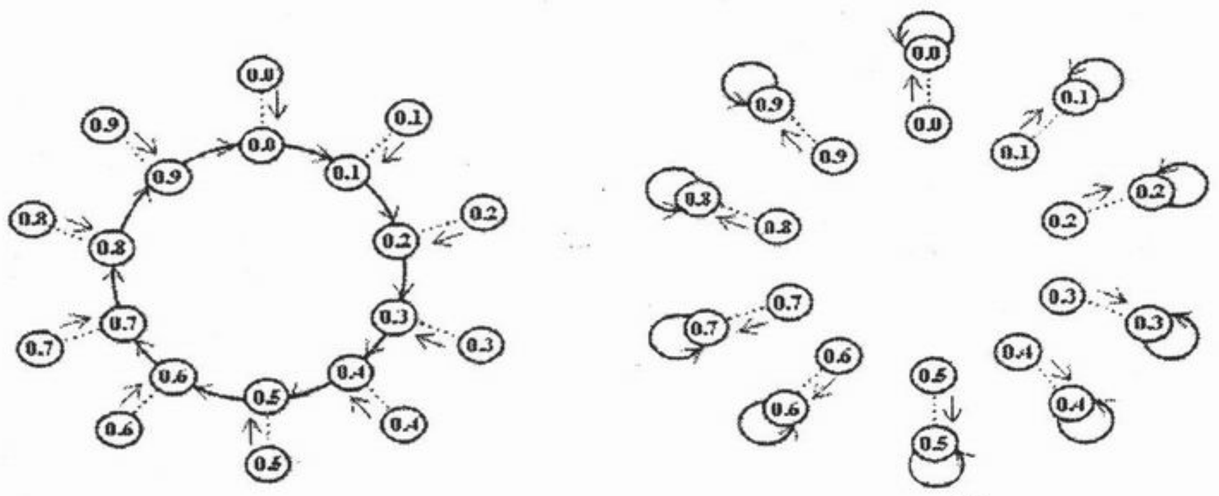
هر عمل در اتوماتای یادگیر متناظر با یک مقدار در مجموعهٔ تعریف شده برای پارامتر مربوطه است. برای مثال اتوماتای یادگیری که ρ را تطبیق می‌دهد دارای ۱۰ عمل است که هر عمل معرف یکی از مقادیری است که در مجموعهٔ $\{0.0,0.1,0.2, \ldots ,0.9\}$ قرار دارد. عمق عمل‌های اتوماتاها در اتوماتاهای ساختار ثابت برابر با ۵ قرار داده شده است. شکل ۲ اتوماتای یادگیر تطبیق پارامتر ρ را نشان می‌دهد. شکل ۲.الف تغییر حالات اتوماتای یادگیر زمانی که اتوماتا پاداش می‌گیرد و ۲.ب تغییر حالات اتوماتای یادگیر زمانی که اتوماتا جریمه می‌شود را نشان می‌دهد. اتوماتاهای یادگیر برای تنظیم پارامترهای دیگر مشابه این اتوماتا است. به بیان دیگر اعمال اتوماتاهای یادگیر مقادیر مختلف پارامترها است که به مورچه داده می‌شود و مورچه بر اساس آنها تصمیم‌گیری و عمل می‌کند و نتیجهٔ کار که همان طول دور تولید شده است برای دادن پاداش یا جریمهٔ به اتوماتاها داده میشود. در واقع مسالهٔ ACS برای اتوماتاهای مورچه نقش محیط را بازی می‌کند. از این طریق هر مورچه در حین اجرای الگوریتم یاد می‌گیرد که بهترین پارامترها برای حل این مساله کدام هستند.

در شروع الگوریتم، عمل انتخابی اتوماتاها (پارامترهای مورچه‌ها) و عمق عمل انتخابی (برای اتوماتاهای ساختار ثابت) بصورت تصادفی انتخاب می‌شود. در بخش بعدی لگوریتم پیشنهادی (ACSLA) و نتایج آزمایشها و مقایسه با الگوریتم Meta ACS ارایه می‌گردد.

٤-١- الگوریتم ACSLA

در این الگوریتم مانند سایر الگوریتم‌های کلونی مورچه‌ها از تعدادی مورچه برای حل مساله استفاده می‌کنیم. هر مورچه به ۳ عدد اتوماتای یادگیر برای تطبیق ۳ پارامتر خود مجهز شده است. در ابتدای الگوریتم عمل انتخابی هر اتوماتای یادگیر که همان پارامتر متناظر اتوماتای یادگیر است بصورت تصادفی انتخاب می‌شود. در هر تکرار تعداد k مورچه انتخاب شده و بصورت تصادفی در گره‌های گراف قرار داده می‌شود و از آنها خواسته می‌شود که یک دور بسازند. مورچه‌ها بصورت تصادفی بگونه‌ای انتخاب می‌شوند که در هر m/k تکرار متوالی (که m تعداد کل مورچه‌ها است) هر مورچه یک بار شرکت کرده باشد.

بر خلاف الگوریتم‌های گزارش شدهٔ در [5] و [6] هر مورچه برای انتخاب گرهٔ بعدی و بهنگام‌سازی فرمون از پارامترهای خود استفاده می‌کند. در پایان هر تکرار اتوماتاهای یادگیر با توجه به نتیجه دورهای تولید شده توسط مورچه‌ها اقدام به تطبیق پارامترهای مورچه‌ها می‌نمایند. اعمال اتوماتاهای یادگیر برای $k/2$ مورچه‌ای که بهترین راه‌حل را یافته‌اند پاداش می‌گیرند و اتوماتاهای یادگیر $k/2$ مورچه‌ای که بدترین راه‌حل را یافته‌اند جریمه می‌شوند.



الف ب

شکل۲: اتوماتای یادگیر از نوع L با ۱۰ عمل برای تطبیق پارامتر ρ. الف) نمودار تغییر وضعیت اتوماتای یادگیر برای پاسخ مطلوب. ب) نمودار تغییر وضعیت اتوماتای یادگیر برای پاسخ نامطلوب.

٤-۲- نتایج شبیه‌سازی

شبیه‌سازی‌ها با استفاده از یک سیستم با مشخصات PIII 500 MHz, 255 MB RAM انجام شده است. در این شبیه‌سازی‌ها مقدار $k=4$ میباشد. در صورتی که از اتوماتاهای ساختار ثابت استفاده کرده باشیم عمل انتخابی و حالت آنرا بصورت تصادفی تعیین می‌شود. تعداد مورچه‌ها را $m=20$ در نظر گرفته‌ایم. تعداد تکرارهای الگوریتم برای هر گراف متفاوت است و مقدار آن در سطر آخر جدول نتایج شبیه‌سازی‌ها آمده است. نتایج گزارش شده میانگین ۱۰ بار اجرای الگوریتم میباشد. انحراف از معیار نتایج میانگیری شده (STD) نیز در جدول آمده است. تعداد تکرارها به گونه‌ای انتخاب شده‌اند که با اجرای الگوریتم‌های ژنتیک مساوی باشد تا بتوان آنها را با هم مقایسه کرد.

در ادامه در آزمایش اول الگوریتم ACSLA را برای اتوماتاهای یادگیر مختلف با الگوریتم Meta ACS مقایسه می‌کنیم. سپس در آزمایشهای بعدی تاثیر عمق اتوماتاهای یادگیر در کارایی الگوریتم (زمانیکه از اتوماتاهای یادگیر با ساختار ثابت استفاده شده است) تعداد تکرارها برای رسیدن به جواب و سرعت رسیدن به جواب را بررسی می‌کنیم.

٤-۲-۱- آزمایش اول: مقایسه الگوریتم ACSLA با الگوریتم Meta ACS

در این آزمایش الگوریتم ACSLA برای اتوماتاهای مختلف شبیه‌سازی گردیده است. شبیه‌سازی را روی ۳ گراف شناخته شده eil51، kro100 و ft70 [7] انجام گرفته است. نتایج شبیه‌سازی‌ها در جدول ۱ آمده است. با توجه به نتایج شبیه‌سازی اتوماتای یادگیر L_{RP} نتایج بهتری را نسبت به سایر اتوماتاهای یادگیر تولید کرده است. نتایج بدست آمده با این اتوماتا را با نتایج الگوریتم Meta ACS [4] در جدول ۲ مقایسه شده است. با مقایسهٔ نتایج شبیه‌سازی می‌توان نتیجه گرفت که الگوریتم ACSLA برای مسایل کوچک و متوسط از هر دو الگوریتم ACS و Meta ACS برتر می‌باشد.

٤-۲-۲- آزمایش دوم: بررسی تاثیر تعداد تکرارها در الگوریتم ACSLA

افزایش تعداد تکرارهای الگوریتم پیشنهادی باعث بهبود در نتایج می‌گردد. نتایج اجرای الگوریتم بر روی گراف rat783 که دارای ۷۸۳ گره می‌باشد در جدول ٤ گزارش شده است. دور بهینه برای این گراف ۸۸۰۶ است. نتایج گزارش شده در جدول ٤ حاصل از میانگین گیری ۳ بار اجرای الگوریتم است. در تکرار ۱۰۰۰۰ ام طول دوری بدست آمده ۹٤٥۰ و در تکرار ۲۱۲۵۷ طول دور بدست آمده برابر با ۹۱۳۹ که با طول دور بهینهٔ ۳۳۳ واحد اختلاف دارد بوده است. نتایج این آزمایش نشان می‌دهد که با افزایش تکرارها نتیجه بدست آمده بهتر می‌شود. البته در تکرارهای بالا سرعت بهبود کاهش می یابد.

جدول ۱: نتایج اجرای الگوریتم ACSLA

Problem Type	eil51		kroA100		ft70	
Automata Type	Average	STD	Average	STD	Average	STD
L	427.6	1.17	21489	42	40395	315
G	429.8	2.89	21527	117	40556	344
Krinsky	428.2	0.79	21698	272	40574	275
Krylov	428.6	2.01	21769	381	40733	287
L_{RP}	427.9	0.57	21520	128	40462	245
$L_{R\varepsilon I}$	428.4	2.76	21434	123	40532	260
L_I	428.4	1.77	21416	110	40728	234
Number of iterations	10000		15000		15000	

جدول ۲: مقایسه نتایج ACSLA با نتایج Meta ACS

Problem Type / Algorithm	eil51	kroA100	ft70
ACSLA (L_{RP})	427.9	21520	40462
Meta ACS	428.5	21513	40569

٤-٢-٣- آزمایش سوم: بررسی تاثیر عمق اتوماتای ثابت در نتایج الگوریتم ACSLA

در این آزمایش تاثیر پارامتر عمق در اتوماتاهای یادگیر با ساختار ثابت را بررسی می‌کنیم. برای این منظور الگوریتم ACSLA را یک بار با اتوماتاهای ثابت که هر یک از اعمال آن دارای عمق ۳ می‌باشد و بار دیگر با اتوماتاهای ثابت که هر یک از اعمال آن دارای عمق ۱۰ می‌باشد بر روی گراف kroA100 آزمایش کرده‌ایم. نتایج آزمایش برای اتوماتاهای ثابت مختلف در جدول ٥ آمده است. این نتایج نشان می‌دهد که افزایش عمق اتوماتا تاثیری مثبت بر نتایج بدست آمده داشته است و هر چه عمق اتوماتا کمتر باشد نتایج بدتر می‌شود. جدول ٥ نشان می‌دهد که برای عمق ۱۰ و ۱۵۰۰ بار تکرار، الگوریتم ACSLA نتایج بهتری در مقایسه با الگوریتم Meta ACS تولید کرده است.

جدول ٤: نتایج تولید شده الگوریتم ACSLA بعد از ۲۰۰۰ تکرار

Problem Type	rat783	
Automata Type	Average	STD
L	10069	100
G	10059	95
Krinsky	10110	55
Krylov	10007	76
L_{RP}	10003	21
$L_{R\varepsilon I}$	9945	58
L_I	10100	151
No. of iterations	20000	

جدول ٥: نتایج ACSLA برای مسالهٔ kroA100 و اتوماتاهای ثابت با عمق‌های مختلف

LA Depth	3		10	
Automata Type	Average	STD	Average	STD
L	21827	317	21423	108
G	21770	403	21373	69
Krinsky	21629	163	21528	195
Krylov	21310	38	21295	12
Meta ACS	21513	---	21513	---
Number of iterations	15000		15000	

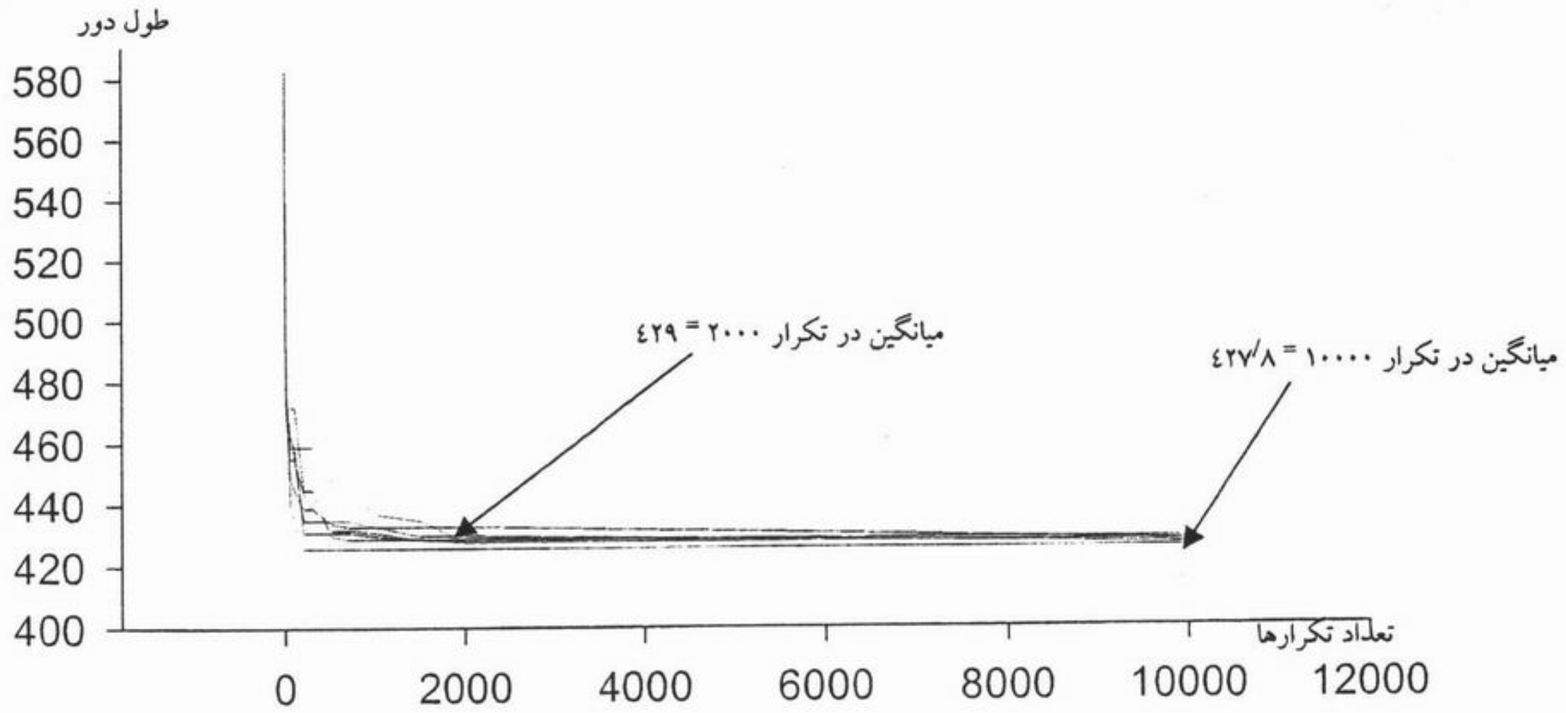
٤-٢-٤- آزمایش چهارم: بررسی سرعت رسیدن به جواب در الگوریتم ACSLA

در این آزمایش سرعت الگوریتم ACSLA را در رسیدن به جواب بررسی می‌کنیم. الگوریتم ACSLA روی گراف eil51 اجرا گردید. نتایج آزمایش نشان می‌دهد که الگوریتم ACSLA برای رسیدن به یک جواب مشخص در مقایسه با الگوریتم ACS تعداد دورهای کمتری نیاز دارد و یا به بیانی دیگر سرعت همگرایی آن بالاتر است. شکل ۳ نتیجه این آزمایش را نشان می‌دهد. در این آزمایش الگوریتم ACSLA را ۸ بار اجرا کرده‌ایم. نتیجهٔ اجرای هر الگوریتم بصورت یک نمودار جداگانه در شکل ۸ آمده است. همانطوریکه در شکل مشخص است الگوریتم ACSLA در تکرار ۲۰۰۰ام به جواب ٤۲۹ رسیده است. تعداد حرکت مورچه‌ها در الگوریتم ACSLA با ۲۰۰۰ تکرار برابر با تعداد حرکت مورچه‌ها در الگوریتم ACS با ٤۰۰ تکرار است. یعنی هزینه الگوریتم ACSLA یک پنجم هزینهٔ اجرای الگوریتم ACS است. همچنین آزمایش نشان می‌دهد میانگین طول دور بدست آمده در ۲۰۰۰ تکرار ACSLA تنها یک واحد با طول دور تولید شده نهایی توسط الگوریتم ACS فاصله دارد. به بیان دیگر با یک پنجم هزینه الگوریتم ACS، الگوریتم ACSLA می‌تواند جوابی به خوبی جواب الگوریتم ACS تولید کند.

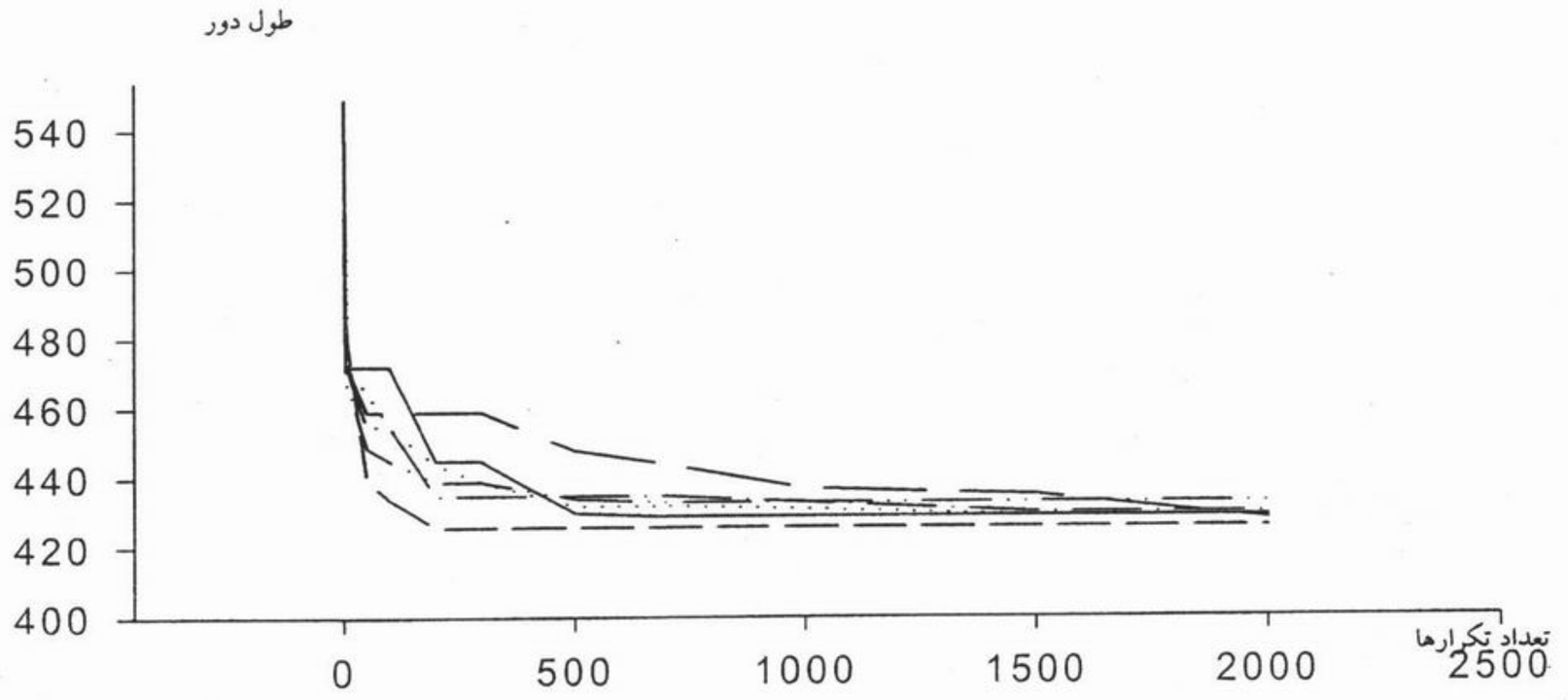
شکل ۴ بازهٔ ۲۰۰۰ تکرار اول را بزرگتر نشان می‌دهد. در این شکل مشخص است که تقریباً در تکرار ۱۰۰۰ام همهٔ نتایج بدست آمده کمتر از ٤٤۰ است. آزمایش‌ها نشان داده اند که سرعت رسیدن به جواب برای الگوریتم ACSLA در گراف‌های بزرگ نیز بالا می‌باشد. برای مثال در گراف rat783 که دارای ۷۸۳ گره است تنها با گذشت حدود ٥ تکرار از الگوریتم ACSLA که هزینه‌ای برابر اجرای ۱ تکرار الگوریتم ACS دارد، نتیجه بدست آمده بهتر از نتیجه نهایی ACS است. اتوماتاهای یادگیر دیگر نیز آزمایش شده‌اند که نتایج مشابهی را برای گراف‌های دیگر تولید کرده است[8].

جدول ٥: نتایج **ACSLA** برای مسالهٔ **kroA100** و اتوماتاهای ثابت با عمق‌های مختلف

LA Depth	3		10	
Automata Type	Average	STD	Average	STD
L	21827	317	21423	108
G	21770	403	21373	69
Krinsky	21629	163	21528	195
Krylov	21310	38	21295	12
Meta ACS	21513	---	21513	---
Number of iterations	15000		15000	



شکل ۳: بررسی سرعت رسیدن به جواب در الگوریتم ACSLA برای مسالهٔ eil51



شکل ۴: بررسی سرعت رسیدن به جواب در ۲۰۰۰ تکرار اول الگوریتم ACSLA

۵- نتیجه گیری

در این مقاله روشی مبتنی بر اتوماتاهای یادگیر برای تطبیق پارامترهای الگوریتم ACS ارایه گردید. با استفاده از شبیه‌سازی نشان داده شد که روش پیشنهادی از کارایی بالایی در ارتباط با سرعت همگرایی و کیفیت جواب‌های تولید شده در مقایسه با یکی از بهترین الگوریتم‌های گزارش شده برخوردار است.

مراجع

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

[2] Dorigo M., Di Caro, G., *The Ant Colony Optimization meta-heuristic, New Ideas in Optimization*, McGraw Hill, London, UK, 1999, pages 11–32.

[3] Coloni, A., Dorigo, M., and Maniezzo, V., *An Investigation of some Property of an Ant Colony, Parallel Problem Solving from Nature*, No. 2, Elsevier Science Publication, 1992.

[4] Marcin L.P., Tony W., *Using Genetic Algorithms to Optimize ACS-TSP*, 2002, http://citeseer.nj.nec.com/, AT: February 2002.

[5] Bullnheimer, B., Hartl, R.F., Strauss, C., *A New Rank-Based Version of the Ant System: A Computational Study*. Tech.Rep.POM-03/97, Institute of Management Science, University of Vienna, 1997.

[6] Stutzle, T., Hoos, H.H., *The Min-Max Ant System and Local Search for the Traveling Salesman Problem, Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)* , 1997, pages 309–314. IEEE Press, Piscataway, NJ.

[7] Reinelt G., *TSPLIB Documentation*. Institut f¨ur Angewandte Mathematik, Universit ¨at Heidelberg. (1995), http://www.research.att.com/~dsj/chtsp/download.html, AT: February 2002.

[8] Abdali, M.F., Meybodi, M.R., *Adaptation of Ant Colony Parameters Using Learning Automata*, Technical Report, Computer Engineering Department, Amirkabir University, 2004

[9] Meybodi, M. R. and Beigy, H., *"A Note on Learning Automata Based Schemes for Adaptation of BP Parameters", Journal of Neurocomputing,* Vol. 48, No. 4, October 2002, pp. 957-974.

[10] Beigy, H. and Meybodi, M. R. *"Backpropagation Algorithm Adaptation Parameters Using Learning Automata", International Journal of Neural System*, Vol. 11, No. 11, No. 3, 2001, PP. 219-228.

[11] Meybodi, M. R. and Beigy, H., *"New Learning Automata Based Algorithms for Adaptation of Backpropagation Algorithm Parameters", International Journal of Neural System*, Vol. 12, No. 1, 2002, PP. 45-67.

[12] Adibi, P., Meybodi, M. R. and R. Safabakhsh, *"Unsupervised Learning of Synaptic Delays based on Learning Automata in an RBF-Like Network of Spiking Neurons for Data Clustering", Journal of Neurocomputing*, Elsevier Publishing Company, Accepted for publication.

[13] Mashoufi, B., Mehaj, M. B., Motamedi, A., and Meybodi, M. R., *"Introducing an Adaptive VLR Algorithm Using Learning Automata for Multilayer Perceptron", IEICE Transactions on Information and Systems*, Vol. E86-D, No. 3, March 2003, pp. 495-609.

[14] Narendra, N and Thathachar, M., "Learning Automata an Introduction", Prentice Hall, 1989.