دهمین
کنفرانس سالانه انجمن کامپیوتر ایران
۲۷ تا ۲۹ بهمن ۱۳۸۳

حل مساله فروشنده دوره‌گرد توسط اتوماتای یادگیر توزیع شده[۱]

محمد علیپور محمدرضا میبدی
دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر
و
پژوهشگاه دانشهای بنیادی، پژوهشکده علوم کامپیوتر
تهران ایران

چکیده: مساله فروشنده دوره گرد از مسائل NP-Complete بوده و بهمین دلیل الگوریتمهای تقریبی متعددی از جمله الگوریتم های مبتنی بر شبکه های عصبی، کولونی مورچه ها و الگوریتم های ژنتیکی که جوابهای قابل قبول در زمان کوتاه تولید می کنند برای حل آن گزارش شده است. اتوماتای یادگیر یک ابزار جستجوی عمومی می باشد و برای حل تعدادی از مسائل NP-Complete بکار برده شده است. در این مقاله با استفاده از اتوماتای یادگیر توزیع شده، یک الگوریتم جدید برای حل مساله فروشنده دوره‌گرد معرفی خواهیم کرد و کارایی الگوریتم ارائه شده را بر روی نمونه مسائل استاندارد مساله فروشنده دوره گرد متقارن و همچنین نامتقارن بررسی کرده و سپس با نتایج بدست آمده توسط الگوریتمهای دیگر مقایسه می کنیم. آزمایشهای انجام گرفته نشان میدهد که الگوریتم پیشنهادی در مقایسه با الگوریتمهای موجود از کارایی خوبی بر خوردار است.

کلمات کلیدی: مساله فروشنده دوره‌گرد، اتوماتای یادگیر، اتوماتای یادگیر توزیع شده

۱- مقدمه

مساله فروشنده دوره‌گرد[۲] یکی از مسائل کلاسیک تئوری گراف در ریاضیات است که البته در علوم مهندسی نیز دارای کاربردهای فراوانی می باشد[۳]. در این مساله یک فروشنده فرضی سعی دارد کوتاهترین مسیر را در عبور از تعدادی شهر انتخاب کند به طوری که می‌بایست از هر شهر تنها یک بار عبور کرده و پس از عبور همه شهرهای مورد نظر به اولین شهر واقع در مسیر بازگردد. در واقع مساله یافتن کوتاه‌ترین گردشی است که N گره یک گراف را به هم متصل می‌کند. این مساله برای اولین بار توسط دانشمندی به نام اولر در سال ۱۷۵۹ مطرح گردید، ولی شهرت واقعی خود را از کتابی که در سال ۱۸۳۲ توسط ویگت نوشته شد، بدست آورد. در سال ۱۹۹٤ گروهی از دانشمندان نمونه مساله‌ای ۷۳۹۷ شهری حل کردند که در سوراخ کاری فیبر مدار چاپی با هدف به حداقل رساندن حرکت دریل بود، بکار می رفت، و همان افراد در سال ۱۹۹۸ مساله‌ای

---

[۱] بخشی از این تحقیق توسط پژوهشگاه دانشهای بنیادی(IPM)، پژوهشکده علوم کامپیوتر حمایت مالی شده است.

[۲] Traveling Salesman Problem (TSP)

[۳] Http://www.tsp.gatech.edu/apps

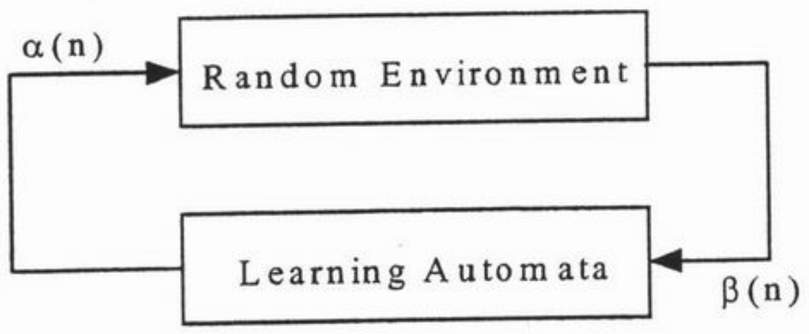
۱۳۵۰۹ شهری را حل کردند [۱]. مساله فروشنده دوره‌گرد با مساله گردش هامیلتونی دارای ارتباط بسیار نزدیکی است، اگر در یک گراف وزن دار، گره‌ها متناظر با شهرها، لبه‌ها متناظر با جاده‌هایی که این شهرها را بهم متصل می سازد و وزن لبه‌ها متناظر با فواصل بین شهرها باشد، مساله فروشنده دوره‌گرد در واقع به مساله یافتن کوتاهترین گردش هامیلتونی تبدیل می‌گردد. مساله TSP به دو گروه TSP متقارن و TSP نامتقارن تقسیم میشود. در TSP متقارن، گراف متناظر با مساله بدون جهت و در TSP نامتقارن گراف متناظر جهت‌دار در نظر گرفته میشود.

الگوریتم های مختلفی برای حل مساله فروشنده دوره گرد گزارش شده است که الگوریتم های کلاسیک جستجوی محلی [۲]، روش سرد شدن فلزات[٤] [۳]، جستجوی تابو[٥] [٤]، شبکه های الاستیک[٥]، شبکه های عصبی[٦]، الگوریتم های ژنتیکی [۷] [۸] و کولونی مورچه ها[٦][۹] از آن جمله می باشند.

در این مقاله یک الگوریتم جدید مبتنی بر اتوماتای یادگیر توزیع شده [۱۰]، برای حل مساله فروشنده دوره‌گرد پیشنهاد می‌شود. این الگوریتم قادر است جوابهای بهینه و یا نزدیک به بهینه را برای هر دو نوع متقارن و نامتقارن مساله تولید نماید. الگوریتم پیشنهادی با الگوریتمهای گزارش شده مبتنی بر الگوریتمهای ژنتیکی، روش سرد شدن فلزات و کولونی مورچه ها مقایسه گردیده است. نتایج این مقایسه سرعت الگوریتم پیشنهادی و کیفیت جوابهای بدست آمده را تائید میکند. ادامه مقاله بشرح زیر سازماندهی شده است. در بخش ۲ مقاله، اتوماتای یادگیر بطور خلاصه معرفی شده و در بخش ۳ اتوماتای یادگیر توزیع شده شرح داده می شود. الگوریتم پیشنهادی در بخش ٤ و نتایج بدست آمده از اجرای الگوریتم را در بخش ٥ نشان می دهیم. بخش پایانی نتیجه گیری میباشد.

۲- اتوماتای یادگیر[۷]

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

۲-۱- محیط[۸]

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 نظر گرفته می شود. در محیط از نوع Q ، $\beta(n)$ می تواند به طور گسسته یک مقدار از مقادیر محدود در فاصله [۰،۱] و در

---

٤ Simulated Annealing
٥ Tabu
٦ Ants Colony
۷ Learning Automata
۸ Environment

محیط از نوع S، $\beta(n)$ متغیر تصادفی در فاصله [۰،۱] است. c_i احتمال اینکه عمل α_i نتیجه نامطلوب داشته باشد می باشد. در محیط ایستا[۹] مقادیر c_i بدون تغییر می مانند، حال آنکه در محیط غیر ایستا[۱۰] 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 که در این مقاله از آنها استفاده شده است می پردازیم.

۲-۲- اتوماتای یادگیر با ساختار متغیر[۱۱]

اتوماتای یادگیر با ساختار متغیر توسط ٤ تائی $\{\alpha, \beta, p, T\}$ نشان داده می شود که در آن $\alpha \equiv \{\alpha_1, \alpha_2, \Lambda, \alpha_r\}$ مجموعه عملهای اتوماتا، $\beta \equiv \{\beta_1, \beta_2, \Lambda, \beta_m\}$ مجموعه ورودیهای اتوماتا، $p \equiv \{p_1, p_2, \Lambda, p_r\}$ 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 متغییرمیباشد

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، 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 را L_{RP}[۱۲]، زمانیکه b از a خیلی کوچکتر باشد، الگوریتم را $L_{R\varepsilon P}$[۱۳] و زمانیکه b مساوی صفر باشد، الگوریتم را L_{RI}[۱٤] می نامیم. برای مطالعه بیشتر در باره اتوماتاهای یادگیر می توان به [۱۱]،[۱۲]،[۱۳]،[۱٦] مراجعه کرد.

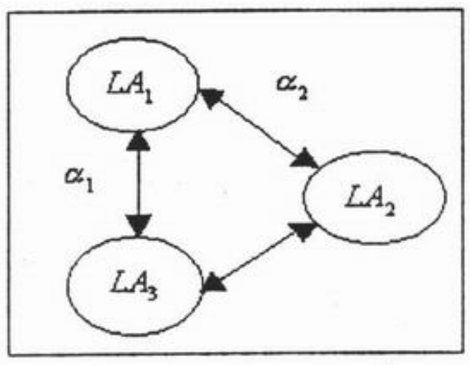
۳- اتوماتای یادگیر توزیع شده[۱۵]

اتوماتای یادگیر توزیع شده (DLA) ، شبکه‌ای از اتوماتاهای یادگیر است که برای حل یک مساله با یکدیگر همکاری می نمایند [۱۰]. تعداد اقدامهای یک اتوماتا در DLA برابر تعداد اتوماتاهای متصل به اتوماتای یادگیر فوق می باشد. انتخاب یک اقدام توسط یک اتوماتا در شبکه، اتوماتای متناظر با این اقدام را فعال می سازد. بعنوان مثال در شکل ۲ هر اتوماتا دارای دو اقدام می باشد. انتخاب اقدام α_2 توسط LA_1، اتوماتا یادگیر LA_3 را فعال خواهد کرد. اتوماتای یادگیر فعال شده (LA_3) بنوبه‌ی خود یکی از اقدامهای خود را انتخاب می کند که در نتیجه آن یکی از اتوماتاهای متصل به آن اتوماتا که متناظر با اقدام

---

[۹] Stationary
[۱۰] Non-Stationary
[۱۱] Variable Learning Automata
[۱۲] Linear Reward Pealty
[۱۳] Linear Reward Epsilon Penalty
[۱٤] Linear Reward Inaction
[۱۵] Distributed Learning Automata

انتخاب شده می باشد فعال می شود. در هر زمان فقط یک اتوماتا در شبکه فعال میباشد. بطور رسمی DLA را میتوان توسط گراف $DLA=(V,E)$ که $V=\{LA_1,LA_2,...,LA_n\}$ مجموعه اتوماتای یادگیر و n تعداد اتوماتاها در DLA و $E\subset V\times V$ مجموعه لبه‌های گراف می باشد، تعریف کرد. لبه (i,j) اقدام j اتوماتا LA_i را نشان می دهد. LA_j زمانی فعال خواهد شد که اقدام j اتاماتون LA_i انتخاب شود. تعداد اقدامهای اتوماتا LA_k ($k=1,2,...,n$) برابر درجه‌ی خروجی آن گره می باشد. برای اطلاعات بیشتر در باره اتوماتای یادگیر توزیع شده میتوان به مراجع [١٤] و [١٠] مراجعه کرد.



شکل ٢: اتوماتای یادگیر توزیع شده (DLA) با ٣ اتوماتای یادگیر

٤- الگوریتم پیشنهادی

ابتدا شبکه‌ای از اتوماتاهای یادگیر که متناظر[١٦] با گراف مساله TSP میباشد ایجاد می شود. در این شبکه هر گره (معادل یک شهر در مساله داده شده)، یک اتوماتا یادگیر با ساختار متغیر بوده و هر لبه‌ی خروجی این گره(جاده ای که توسط آن از این شهر میتوان به شهری دیگری رفت) یکی از اقدامهای آن اتوماتا می باشد. تعداد اقدامهای یک اتوماتا یادگیر معادل تعداد شهرهایی می باشد که میتوان بطور مستقیم از این شهر به آنها مسافرت کرد. خروجی DLA ترتیبی از اقدامهای انتخاب شده توسط اتوماتاها می باشد که مسیر (یا گردش) ویژه‌ای را در گراف نشان می دهد. محیط از طول این مسیر برای تولید خروجی استفاده می کند. این خروجی با توجه به مطلوب یا نامطلوب بودن آن، باعث پاداش و یا پنالتی دادن به اقدامهای واقع در این مسیر (گردش هامیلتونی ایجاد شده البته اگر گردش هامیلتونی در گراف وجود داشته باشد) می شود.

قبل از اینکه به شرح الگوریتم پیشنهادی بپردازیم بایستی به این نکته اشاره نمود که آزمایشهای انجام شده نشان داد که اگر اتوماتای یادگیر فعال فقط از بردار احتمال اقدامها برای انتخاب اقدام خود استفاده کند، جوابهای بدست آمده توسط الگوریتم تقریبا غیر قابل قبول بوده و همچنین نرخ همگرایی الگوریتم بسیار پایین میباشد. جهت رفع این مشکل، به هر اتوماتای یادگیر اجازه داده شد در انتخاب اقدام خود، علاوه بر استفاده از بردار احتمال اقدامها، از مقدار عکس فاصله بین دو گره (عکس فاصله بین گره فعال (اتوماتای فعال) و گره‌های متصل به این لبه نیز استفاده کند. آزمایشها نشان داد که استفاده از این مقدار در انتخاب شهر بعدی بهبود قابل ملاحظه‌ای در کارایی و نرخ همگرایی الگوریتم ایجاد میکند و باعث تولید جوابهای بهینه و یا خیلی نزدیک به جواب بهینه[١٧] میشود. مقدار عکس فاصله بین دو گره j و i در گراف TSP توسط تابع $W^{-1}(j,i)$ نشان داده میشود. برای استفاده از این تابع در انتخاب اقدام اتوماتا فعال، بردار احتمال اقدامهای اتوماتای j، P^j را بطور موقت طبق روابط زیر به بردار P'^j تغییر می دهیم. پس از انتخاب اقدام، بردار احتمال اقدامها مجددا به مقدار قبلی خود P^j برگردانده می شود. این کار در هر تکرار انجام میگیرد. روابط زیر چگونگی محاسبه بردار P'^j را از بردار احتمال اقدام P^j نشان می دهد

بردار احتمال اقدام اتوماتا j:

[١٦] Isomorphic
[١٧] Optimal

$$P^j = [p_1{}^j, p_2{}^j, ..., p_r{}^j]^T$$

بردار احتمال اقدام تغییر یافته اتوماتا j:

$$P'^j = \{p_i'^j \mid p_i'^j = \frac{[p_i{}^j \times W^{-1}(j,i)]^\beta}{\sum_{i=1}^{r}[p_i{}^j \times W^{-1}(j,i)]^\beta} \quad : \quad i = 1,2,...,r\}$$

در این رابطه P_i^j، احتمال انتخاب اقدام i توسط اتوماتا j می باشد و $W^{-1}(j,i)$، عکس فاصله بین دو گره j و i در گراف TSP بوده و $\beta \geq 1$ اهمیت نسبی فاصله بین دو گره در انتخاب یک اقدام را مشخص میکند. بعبارتی دیگر احتمال انتخاب یالهایی (اقدامهایی) که که دارای احتمال انتخاب بیشتر و طول یال کمتر باشند، بالاتر رفته و بالعکس. آزمایشها نشان داده اند که با اعمال تغییرات فوق در بردار احتمال اقدام و با توجه به در نظر گرفتن فاصله بین دو گره، نرخ همگرایی الگوریتم پیشنهادی بمیزان قابل ملاحظه‌ای افزایش می یابد و در عین حال جوابهای نزدیک به بهینه تولید خواهد شد.

اکنون به توصیف الگوریتم پیشنهادی می پردازیم. در گام نخست یکی از شهرها از گراف TSP بصورت تصادفی بعنوان شهر آغازی گردش انتخاب می شود. در مرحله بعد اتوماتای متناظر با این شهر یکی از اقدامهای خود را طبق بردار احتمال اقدامهای تغییر یافته، P'^j، با استفاده از تابع GetNextCity() انتخاب می کند (شکل ٤). اعمال این اقدام، اتوماتا طرف دیگر لبه ای که متناظر با اقدام انتخاب شده میباشد را فعال می سازد. با توجه به اینکه در گردش هامیلتونی هر شهر نبایستی بیش از یکبار ملاقات شود بایستی ترتیبی اتخاذ نمود نا هیچ شهری (گره‌ای) بیش از یکبار انتخاب نشود. برای این منظور اگر اتوماتایی اقدام k را از لیست اقدامهای خود انتخاب کند، همزمان با آن هر اتوماتای غیرفعال اقدام k را در لیست اقدامهای خود را غیر فعال می سازند (اما حذف نمی کنند). در ابتدای هر تکرار تمام اقدامهای غیر فعال شده مجددا فعال خواهند شد. اتوماتای فعال شده با استفاده از بردار احتمال اقدام تغییر یافته‌ی خود، اتوماتا طرف دیگر لبه انتخاب شده را فعال می سازد. فرآیند انتخاب اقدام و فعال سازی اتوماتای متناظر با اقدام انتخاب شده تا ملاقات همه گره‌های (شهرهای) موجود در گراف TSP و برگشت به شهر آغازین و یا بدلیل اینکه امکان انتخاب اقدام بعدی برای اتوماتا فعال وجود نداشته باشد (امکان ایجاد گردش هامیلتونی با توجه به شهرهایی که قبلا انتخاب شده‌اند و یا گراف نمونه مساله دارای گردش هامیلتونی نباشد)، ادامه می یابد.

پس از پیدا کردن یک گردش، در مرحله سوم طول گردش بدست آمده محاسبه شده و با طول بهترین گردشی که تابحال بدست آمده است مقایسه میگردد. بر اساس نتیجه مقایسه، بردار احتمال اقدامهای اتوماتاهای DLA بروز میشود. نحوه‌ی بروزرسانی بردار احتمال اقدامها بدینصورت است که اگر طول گردش ایجاد شده کوچکتر و یا مساوی طول بهترین گردشی که تابحال ایجاد شده است باشد، همه اتوماتاهای DLA، اقدام انتخابی خود را طبق الگوریتم یادگیری L_{R-I}، پاداش می دهند. برای روشن شدن بیشتر این مطلب الگوریتم L_{R-I} ذکر شده در بخش آتوماتای یادگیر را مجددا یادآوری می نمائیم. بعنوان مثال اگر طول گردش ایجاد شده در یک تکرار (t) کوچکتر و یا مساوی طول بهترین گردش ایجاد شده تاکنون باشد و در این تکرار اتوماتا j از مجموعه اقدامهای مجاز خود اقدام i را انتخاب کرده باشد، احتمال انتخاب اقدام i طبق رابطه‌ی زیر افزایش خواهد یافت:

$$p_i(t+1) = p_i(t) + a[1 - p_i(t)]$$

و احتمال انتخاب سایر اقدامهای اتوماتا j بصورت زیر کاهش خواهد یافت:

$$p_k(t+1) = (1-a)p_k(t) \qquad k \neq i \quad k = 1,2,...,r$$

در رابطه‌ی بالا پارامتر a نرخ یادگیری[۱۸] و r تعداد اقدامهای اتوماتا j می باشد. فرآیند ایجاد گردش تا زمانیکه تعداد گردشهای ایجاد شده توسط الگوریتم از یک مقدار از پیش تعیین شده بیشتر ۷۷باشد و یا احتمال مسیر[۱۹] که عبارتست از حاصلضرب احتمال انتخاب لبه‌های موجود در گردش ایجاد شده، از یک مقدار آستانه (برای مثال ۰،۹) فزونی گیرد ادامه می یابد و کوتاهترین گردش هامیلتونی ایجاد شده توسط الگوریتم، گردش تقریبی تولید شده توسط الگوریتم میباشد. الگوریتم پیشنهادی در شکل ۳ نشان داده شده است.

```
Procedure TSP
Begin
  Initialize the probability vector of each automaton
    Repeat                                                    //Phase ۱
    InitialCity := Generate a random number between ۱ and n
    CurrentCity := InitialCity   //(choosing the intial city of tour randomly)
    Disable action 'CurrentCity' of all Unactivated LAs       //Phase ۲
    for i := ۱ to n
      if i < n then
        //choose the next node based on active LA modified action probability vector, P'^j
        NextCity := GetNextCity(Activated Automaton)
        //if there is no edge between the current city and each of the unselected cities, the current path can't be a
        hamiltonian cycle
        Disable action 'NextCity' of all Unactivated LAs
        if i = n-۱ then
          Enable action 'InitialCity' of LA NextCity
        end if
      else
          NextCity := InitialCity
        //if there is no edge between the current city and intial city, current path isn't hamilton cycle
      end if
      CurrentCity := NextCity
    next i                                                    //Phase ۳
    Compute the tour length
    if CurrentTourLength < BestTourLength then
      //Reward the selected actions of all LAs along the tour according to the L_{R-I}
        for j := ۱ to n  //<i> is the selected action of automata <j>
          p_i(t+1) = p_i(t) + a[1 - p_i(t)]
          p_k(t+1) = (1-a) p_k(t)      k ≠ i    k = 1,2,..., r
        Next j
    End if                                                    //Phase ٤
    Enable all the disabled actions of LAs
  Until (stop condition)
End Procedure
```

شکل ۳: الگوریتم حل TSP با استفاده از DLA

```
.Function GetNextCity(j: Integer)
  //modify the action probability vector of <j> ( P^j ) as:

  P'^j = {p'^j_i | p'^j_i = [p_i^j × W^{-1}(j,i)]^β / Σ_{i=1}^{r} [p_i^j × W^{-1}(j,i)]^β      :  i = 1,2,...,r}

  Choose an action based on modified action probability vector, P'^j
  Restore Previous value of P^j
End function
```

شکل ٤: تابع GetNextCity برای انتخاب یکی از اقدامهای مجاز اتوماتای فعال

[۱۸] Learning Rate
[۱۹] Path Probability

۵- نتایج حاصل از اجرای الگوریتم

کارایی الگوریتم پیشنهادی با استفاده از نمونه مسائل موجود در کتابخانه TSPLIB[۲۰] مورد بررسی قرار گرفته است. در این کتابخانه بیش از ۱۰۰ نمونه مساله فروشنده دوره‌گرد بهمراه بهترین جواب یافته شده برای هر کدام موجود است که بزرگترین مساله دارای ۸۵۹۰۰ شهر می باشد. ارزیابیهای انجام گرفته بر روی نمونه مسائلی است که مقدار بهینه آنها شناخته شده است.

در این قسمت نتایج حاصل از اجرای الگوریتم بر روی نمونه‌های مختلف متقارن و نامتقارن TSP برای مقادیر مختلف پارامترهای الگوریتم پیشنهادی مورد بررسی قرار میگیرد. نتایج گزارش شده میانگین حداقل ۵ مرتبه اجرای الگوریتم میباشد. پارامترهای a و b و β بترتیب برابر $\frac{2*(n-2)}{n*(n-1)}$ ، $a/2$ و ۱۰ در نظر گرفته شده است که n تعداد شهرهای موجود در نمونه مساله داده شده می باشد. مقادیر فوق برای پارامترها بصورت تجربی بدست آمده است. آزمایشها با استفاده از کامپیوتر خانگی AMD ۱۳۰۰ MHZ با ۱۲۸ مگابایت حافظه اصلی انجام گرفته است.

جدول ۱ نتایج اجرای الگوریتم، بر روی نمونه‌های متقارن مساله TSP را نشان می دهد. جوابهای بدست آمده توسط الگوریتم دارای خطایی در حدود ۰ الی ٤ درصد می باشند که قابل قبول هستند. در این جدول اولین ستون نام مساله و در داخل پارانتز تعداد شهرهای مساله و در داخل براکت تعداد ارزیابیهای انجام گرفته آمده است. ستون دوم بهترین نتیجه‌ی بدست آمده از اجرای الگوریتم، سومین ستون میانگین نتایج ۵ اجرای مختلف، ستون چهارم جواب بهینه نمونه مساله را نشان میدهد و ستون پنجم زمان لازم برای رسیدن به بهترین نتیجه توسط الگوریتم برحسب دقیقه می باشد و آخرین ستون، درصد خطای الگوریتم از جواب بهینه می باشد که برابر: تفاضل جواب بهینه(N۱) از بهترین نتیجه الگوریتم(N۲) به جواب بهینه است

جدول ۱: نتایج الگوریتم پیشنهادی برای مسائل متقارن TSP

Problem (cities) [#evaluation]	Best Result	Average	Best Known	Time (minute)	Error % (N۲-N۱)/N۱
Square(۲۵) [۲۰۰۰۰]	۳٦ [۱۱]	۳٦	۳٦	۰	۰٪
eil (۵۱) [۲۵۰۰۰]	٤۳ [۲۱۰۰۰]	٤۳۵	٤۲٦	۲	۰/۹٪
Berlin۵۲(۵۲) [۱۰۰۰۰]	۷۵٤۲ [٦۸۰۰]	۷۵٤٤	۷۵٤۲	۱	۰٪
eil(۷٦) [۳۰۰۰۰]	۵٤٦ [۲٤۹۰۰]	۵۵۱	۵۳۸	۵	۱/۱٪
Kroa (۱۰۰) [۵۰۰۰۰]	۲۱۹۵۰ [۵۰۰۰۰]	۲۲۳۰۰	۲۱۲۸۲	۱۰	۳/۱٪
Eil۱۰۱(۱۰۱) [۳۰۰۰۰]	٦۵۰ [۳۰۰۰۰]	٦۷۰	٦۲۹	۱۱	۳٪

نتایج اجرای الگوریتم بر روی نمونه‌های نامتقارن مساله TSP در جدول ۲ نشان داده شده است. با مقایسه نتایج جدول ۱ و۲ میتوان نتیجه گرفت نتایج الگوریتم برای نمونه‌های متقارن، از میزان خطا کمتر و زمان اجرا پایین تر ی برخوردار است.

جدول ۲: نتایج الگوریتم پیشنهادی برای مسائل نا متقارن TSP.

Problem (cities) [#evaluation]	Best Result	Average	Best Known	Time (minute)	Error % (N۲-N۱)/N۱
Br۱۷(۱۷) [۲۰۰۰۰]	۹ [۳]	۳۹	۳۹	۰	۰٪
Ftv۳۳(۳۳) [۲۰۰۰۰]	۲۸٦ [٤٤۵۰]	۱۲۸٦	۱۲۸٦	۱	۰٪
Ftv۳۵ [۲۰۰۰۰]	٤۹۰ [۱۸۰۰۰]	۱۵۱۰	۱٤۷۳	۲	۱/۰٪
Ftv۳۸ [۲۵۰۰۰]	۵٦۱ [۲۱۸۰۰]	۱٦۱۰	۱۵۳۰	۳	۲/۰٪
Ry٤۸p (٤۸) [۲۰۰۰۰]	۵۰۰۰ [۱۳۰۰۰]	۱۵۱۵٤	۱٤٤۲۲	۲	٤/۰٪
Ft۷۰ (۷۰) [۲۰۰۰۰]	۱۰۵۹٤ [۹۲۰۰]	٤۰۹۳۹	۳۸٦۷۳	۳	٤/۰٪

[۲۰] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB۹۵/

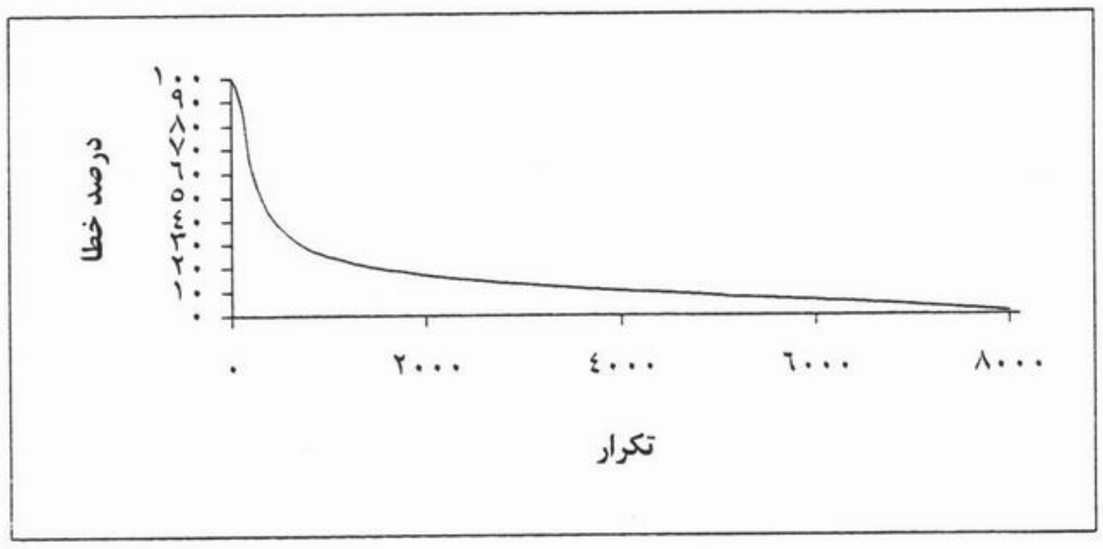
در جدول ۳ نتایج الگوریتم پیشنهادی با الگوریتمهای کولونی مورچه ها (ACS)، الگوریتمهای ژنتیکی (GA)، برنامه نویسی تکاملی (EP)، روش سرد شدن فلزات (SA) و ترکیب الگوریتمهای SA و GA (AG) مقایسه می شود. این جدول نشان میدهد که که جوابهای الگوریتم پیشنهادی در مقایسه با الگوریتمهای ACS، GA و EP بدتر می باشد [۱۵] ولی اگر زمان اجرا و تعداد تکرارهای لازم برای رسیدن به جواب بهینه را در نظر بگیریم، موضوع فرق خواهد کرد. مثلا الگوریتم ACS از ده مورچه برای پیدا کردن جواب بهینه استفاده می کند و برای نمونه، جواب بهینه eil۵۱ با ایجاد ۱۸۳۰ گردش توسط هر مورچه بدست می آید. که در واقع تعداد کل گردشهای انجام گرفته (حاصلضرب تعداد مورچه‌ها در تعداد گردشهای ایجاد شده توسط هر مورچه) توسط این الگوریتم بالغ بر ۱۸۳۰۰ خواهد شد. الگوریتم ACS در شرایط مشابه با الگوریتم پیشنهادی اجرا گردید و مشاهده شد که زمان لازم برای رسیدن به جواب بهینه برای مساله eil۵۱ توسط الگوریتم ACS حدودا ۵ برابر زمان لازم توسط الگوریتم پیشنهادی میباشد.

شکل ٦ همگرایی الگوریتم پیشنهادی به جواب بهینه را برای مساله eil۵۱ نشان میدهد. در این نمودار محور افقی تعداد تکرار و محور عمودی میزان خطای الگوریتم را نشان میدهد. همانگونه که مشاهده میشود بعد از سپری شدن چیزی در حدود ۸۰۰۰ تکرار، الگوریتم پیشنهادی جواب بهینه را پیدا میکند. برای حصول به جوابهای با خطای قابل قبول تعداد تکرارهای مورد نیاز به مراتب کمتر است.

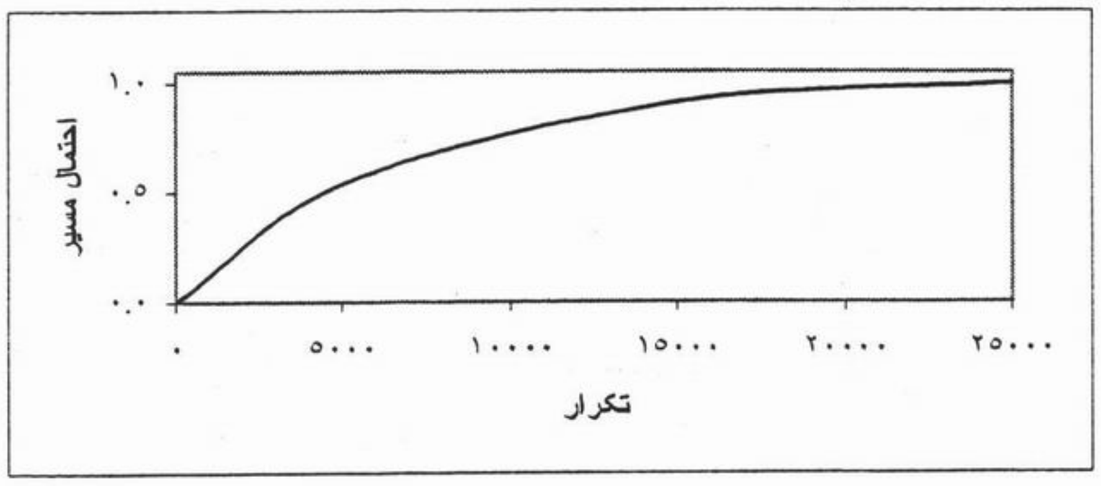
جدول ۳: مقایسه نتایج الگوریتم پیشنهادی با تعدادی از الگوریتمهای مطرح

Problem Name	Propsed Algorithm	ACS	GA	EP	SA	AG	Optimum
Eil۵۱	٤۳۰ [۸۰۵۰]	٤ ۲٦ [۱۸۳۰۰]	٤۲۸ [۲۵۰۰۰]	٤۲٦ [۱۰۰۰۰۰]	٤٤۳ [٦۸۵۱۲]	٤۳٦ [۲۸۱۱۱]	٤ ۲ ٦
Eil۷٦	۵٤٦ [۲٤۹۰۰]	۵ ۳ ۸ [۳٤۸۰۰]	۵٤۵ [۸۰۰۰۰]	۵ ٤ ۲ [۳۲۵۰۰۰]	۵۸۰ [۱۷۳۲۵۰]	۵٦۱ [۹۵۵۰٦]	۵۳۸
Kora۱۰۰	۲۱۹۵۰ [۵۰۰۰۰]	۲۱۲۸۲ [٤۸۲۰۰]	۲۱۷٦۱ [۱۰۳۰۰۰]	N/A [N/A]	N/A [N/A]	N/A [N/A]	۲۱۲۸۲

شکل ٦ نمودار همگرایی الگوریتم به جواب بهینه نمونه مساله Eil۵۱ را نشان می دهد در شکل ۷ نمودار احتمال مسیر که برابر حاصلضرب احتمال انتخاب اقدامهای واقع بر گردش بهینه میباشد، نشان داده شده است، هر چه الگوریتم به جواب بهینه و یا نزدیک به بهینه نزدیکتر شود، احتمال مسیر به یک نزدیکتر خواهد شد.



شکل ٦: نمودار همگرایی الگوریتم به جواب بهینه نمونه eil۵۱.



شکل ۷: نمودار احتمال مسیر

٦- نتیجه گیری

در این مقاله یک الگوریتم تقریبی مبتنی بر اتوماتای یادگیر توزیع شده برای حل مساله فروشنده دوره‌گرد ارائه شد. مزایای این الگوریتم نسبت به الگوریتمهای تقریبی دیگر مانند الگوریتم های ژنتیکی، کولونی مورچه ها و روش سرد شدن فلزات مورد مطالعه قرار گرفت. از طریق شبیه سازی کامپیوتری نشان داده شد که الگوریتم پیشنهادی نسبت به الگوریتمهای فوق الذکر از کارایی بالاتری برخوردار است.

مراجع

[۱] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs", *Springer-Verlag*, ۲nd edition, ۱۹۹٤.

[۲] G. Reinelt, "The Traveling Salesman Problem: Computational Solutions for TSP Applications", *Lecture Notes in Computer Science*, Spring-Verlag, Vol. ۸٤۰, ۱۹۹٤.

[۳] P. van Laarhoven and E. H. L. Aarts, "*Simulated Annealing: Theory and Applications*", Kluwer Academic Publishers, ۱۹۸۷.

[٤] L. Fiechter, "A Parallel Tabu Search Algorithm for Large Traveling Salesman Problems", *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, No. ٥۱, ۱۹۹٤, pp. ۳۳-۲٦۷,

[٥] R. Durbin, R. Szeliski, and A. Yuille, "An Analysis of the Elastic Net Approach to the Traveling Salesman Problem", *Neural Computation*, Vol. ۱, pp. ۳۸٤-۳۸٥, ۱۹۸۹.

[٦] E. H. L. Aarts and H. P.Stehouwer, "Neural Networks and the Traveling Salesman Problem", *Proc. Int. Conf. On Artificial Neural Networks*, Springer-Verlag, ۱۹۹۳, pp. ۹٥۰-۹٥٥.

[۷] D. E. Goldberg, "*Genetic Algorithms in Search, Optimization and Machine Learning*", Addison-Wesley, ۱۹۸۹.

[۸] L. Homaifar, C. Guan, and G. Liepins, "A New Approach to the Traveling Salesman Problem by Genetic Algorithms", *in Proc. ٥th Int. Conf. on Genetic Algorithms*, pp. ٤٦۰-٤٦٦, Morgan Kaufmann Publishers, ۱۹۹۳.

[۹] L. M. Gambardella and M. Darigo, "Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem", *Proc. ۱۲th Int. Conf. On Machine Learning*, pp. ۲٥۲-۲٦۰, Morgan Kaufmann, ۱۹۹٥.

[۱۰] M. R. Meybodi and H. Beigy, "Solving Stochastic Shortest Path Problem Using Distributed Learning Automata", *Proceedings ۲th Annual CSI Computer Conference*, University of Isfahan's Computer Engineering Department, Isfahan, Iran, ۲۰۰۱.

[۱۱] S. Lakshmivarahan, "*Learning Algorithms: Theory and Applications*", New York: Springer-verlag, ۱۹۸۱.

[۱۲] M. R. Meybodi and S. Lakshmivarahan, "On a Class of Learning Algorithms which have Symetric Behavior under Success and Failer", *Lecture Notes in Statistics*, pp. ۱٤٥-۱٥٥., Berlin: Springer-Verlag, ۱۹۸٤.

[۱۳] K. S. Narendra and K. S. Thathachar, "*Learning Automata: An Introduction*", New York: Prentice-Hall, ۱۹۸۹.

[۱٤] H. Beigy and M. R. Meybodi "A New Distributed Learning Automata for Solving Stochastic Shortest Path Problem", *Proceedings of the Sixth International Joint Conference on Information Science*, Durham, USA۲۰۰۲, , pp. ۳۳۹-۳٤۳,

[۱٥] M. Dorigo and L. M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", *IEEE Trans*. Evol. Comp. Vol. ۱, pp. ٥۳-٦٦, ۱۹۹۷.

[۱٦] P. Mars, J. R. Chen, and R. Nambir, "Learning Algorithms: Theory and Applications in Signal Processing", *Control, and Communication*, CRC Press Inc., ۱۹۹٦.