حل مساله فروشنده دوره‌گرد پویا توسط اتوماتاهای یادگیر واکنشی توزیع شده

محمد علیپور[۱] محمدرضا میبدی[۲]

آزمایشگاه سیستمهای نرم افزاری
دانشکده مهندسی کامپیوتر و فناوری اطلاعات
دانشگاه صنعتی امیرکبیر
تهران ایران
mmdalipour@yahoo.com

چکیده

در این مقاله الگوریتم جدیدی برای حل مساله فروشنده دوره‌گرد پویا[۳] با استفاده از اتوماتای یادگیر واکنشی توزیع شده[۴] ارائه میگردد. در مساله فروشنده دوره‌گرد پویا که در این مقاله مورد توجه میباشد هزینه‌ی (زمان) مسافرت بین شهرهای موجود در مساله با زمان تغییر میکند. الگوریتم پیشنهادی از طریق نشان دادن عکس العمل بموقع ومناسب در برابر وقوع تغییرات، راه حلی که با تغییرات تطبیق داشته باشد را پیدا مینماید. برای حصول به این هدف از یک اتوماتای یادگیر به نام اتوماتای یادگیر واکنشی برای بروز رسانی بردار احتمال اقدامهای اتوماتاهای یادگیر در اتوماتای یادگیر توزیع شده استفاده میشود. از طریق شبیه سازی کامپیوتری کارایی الگوریتم پیشنهادی نشان داده میشود.

کلمات کلیدی: مساله فروشنده دوره گرد پویا، اتوماتاهای یادگیر، اتوماتاهای یادگیر توزیع شده، بهینه سازی

۱- مقدمه

مساله فروشنده دوره‌گرد پویا[۵] یکی از انواع مساله فروشنده دوره‌گرد می باشد که بعضی از پارامترهای آن در طی زمان تغییر پیدا میکند. در مساله فروشنده دوره‌گرد پویا ممکن است هزینه‌ی (زمان) مسافرت بین بعضی از شهرهای موجود در مساله در طی زمان تغییر پیدا کند که میتواند ناشی از افزایش یا کاهش در هزینه‌ی زمانی پیمایش بعضی از لبه ها به علت تغییر در تراکم ترافیک آن لبه ها باشد. در نوع دیگری از مساله فروشنده دوره‌گرد پویا ممکن است شهری یا شهرهایی به مساله اضافه و یا از آن حذف شود[9,10].

---

[1] دانشجوي کارشناسي ارشد مهندسي نرم افزار، دانشکده مهندسي کامپيوتر ، دانشگاه صنعتي امير کبير
[2] عضو هيات علمي دانشکده مهندسي کامپيوتر ، دانشگاه صنعتي امير کبير

[3] Dynamic Travelling Salesman Problem
[4] Distributed Responsive Learning Automata
[5] DTSP

مساله فروشنده دوره‌گرد پویا برای اولین بار توسط Psaraftis [1] در سال ۱۹۸۵ معرفی شد و اخیرا Irani و Regan در مقاله‌ای به بررسی نمونه خاصی از این مساله پرداخته‌اند [3]. الگوریتمهای مختلفی برای حل این مساله ارائه شده است که میتوان از آن جمله به الگوریتمهای کولونی مورچه اشاره کرد [8]. یکی از این ویژگیهای مهم الگوریتمهای حل مسائل پویا استفاده از جوابهای بدست آمده قبل از وقوع تغییرات برای پیدا کردن توری است که با تغییرات مطابقت داشته باشد و در عین حال محاسبه آن در مدت زمان قابل قبولی انجام گیرد. ساده‌ترین روش برخورد با تغییرات بوجود آمده، شروع مجدد الگوریتم پس از وقوع تغییرات می باشد. اگر تغییرات صورت گرفته نسبتا کوچک باشند، راه حل جدید به راه حلی که قبل از تغییرات بدست آمده است نزدیک میباشد و الگوریتم میتواند آنرا به سرعت پیدا نماید. در صورتیکه تغییرات انجام گرفته زیاد باشد نرخ همگرایی الگوریتم پایین آمده و به سختی میتواند راه حل جدید را پیدا کند.

در این مقاله یک الگوریتم جدید برای حل مساله فروشنده دوره‌گرد پویا[6] با استفاده از اتوماتای یادگیر توزیع شده (DLA[7]) ارائه میگردد. در مساله فروشنده دوره‌گرد پویا که در این مقاله بررسی خواهد شد هزینه‌ی (زمان) مسافرت بین بعضی از شهرهای موجود در مساله ممکن است در طی زمان تغییر پیدا کند. اتوماتای یادگیر توزیع شده که برای اولین بار در [2] معرفی گردیده است قبلا در حل مساله فروشنده دوره گرد غیر پویا و مساله کوتاه ترین مسیر در گرافهای تصادفی بکار رفته است[2][11].

الگوریتم پیشنهادی با استفاده از نتایج بدست آمده قبل از وقوع تغییرات، تور قابل قبول را محاسبه میکند. برای رسیدن به این هدف راهکارهای مختلفی که تمامی آنها براساس تنظیم مجدد بردار احتمال اقدام[8] اتوماتونهای یادگیر موجود در DLA می باشند، پیشنهاد و با یکدیگر مقایسه شده اند. با توجه به نتایج بدست آمده از آزمایشها میتوان نتیجه گرفت در محیطهایی که تغییرات با سرعت بالایی رخ میدهد استفاده از نتایج مراحل قبل برای محاسبه تور قابل قبول، ضروری بنظر می رسد ولی در محیطهایی که تغییرات به کندی صورت میگیرد حل مساله بدون در نظر گرفتن نتایج مراحل قبل، مقرون بصرفه تر میباشد. همچنین نشان داده شده است که برای مسائلی که از تغییرات زیادی برخوردارند، تنظیم بردار احتمال اقدام تعدادی از اتوماتونهای موجود در DLA که نزدیک به محل وقوع تغییرات می باشد، در مقایسه با تنظیم بردار احتمال اقدام همه اتوماتونهای شبکه DLA، نتایج بهتری تولید می کند. در الگوریتم پیشنهادی از اتوماتاهای یادگیر واکنشی استفاده میگردد. در این گونه از اتوماتاهای یادگیر قانون بروز رسانی بردار احتمال اقدام ها بگونه ای است که از کمتر شدن احتمال اقدامها از یک مقدار معین جلوگیری شده که از این طریق از صفر شدن احتمال انتخاب یک اقدام جلوگیری میشود و بدین وسیله اتوماتای یادگیر توانایی تشخیص تغییرات محیط و عکس العمل در قبال این تغییرات را دارا خواهد شد.

ادامه مقاله بشرح زیر سازماندهی شده است: در بخش ۲ اتوماتاهای یادگیر، یادگیر واکنشی و یادگیر واکنشی توزیع شده مختصرا توضیح داده می شود. در بخش ۳ الگوریتم پیشنهادی شرح داده خواهد شد. بخش ۴ به ارایه نتایج شبیه سازیها اختصاص دارد. بخش نهایی مقاله نتیجه گیری میباشد.

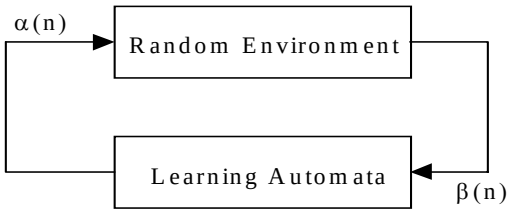
۲- اتوماتاهای یادگیر[9]

اتوماتای یادگیر یک مدل انتزاعی است که تعداد معدودی 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.

محیط[10]: 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

[6] Dynamic Travelling Salesman Problem
[7] Distributed Learning Automata
[8] Action Probability Vector
[9] Learning Automata
[10] Environment

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 [0,1] و در محیط از نوع S، $\beta(n)$ متغیر تصادفی در فاصله [0,1] است. c_i احتمال اینکه عمل α_i نتیجه نامطلوب داشته باشد می باشد. در محیط ایستا[11] مقادیر c_i بدون تغییر می مانند ، حال آنکه در محیط غیر ایستا[12] این مقادیر در طی زمان تغییر می کنند. اتوماتاهای یادگیر به دو گروه با ساختار ثابت و با ساختار متغیر تقسیم بندی میگردند. در ادامه به شرح مختصری در باره‌ی اتوماتاهای یادگیر با ساختار متغییر که در این مقاله از آنها استفاده شده است، می پردازیم.



شکل ۱: ارتباط بین اتوماتای یادگیر و محیط

اتوماتاها ی یادگیر با ساختار متغیر[13]: اتوماتاهای 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 اتوماتای یادگیر، $\beta \equiv \{\beta_1, \beta_2, \cdots, \beta_m\}$ مجموعه ورودیهای اتوماتای یادگیر، $p \equiv \{p_1, p_2, \cdots, p_r\}$ 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 اتوماتاهای یادگیر، 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 در اتوماتای یادگیری 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 را (Linear Reward Pealty)، L_{RP} می نامیم. زمانیکه b از a خیلی کوچکتر باشد، الگوریتم را (Linear Reward Epsilon Penalty)، $L_{R\varepsilon P}$ می نامیم. زمانیکه b مساوی صفر باشد، الگوریتم را (Linear Reward Inaction)، L_{RI} می نامیم. برای مطالعه بیشتر در باره اتوماتاهای یادگیر می توان به [4],[5],[6],[7] مراجعه کرد.

[11] Stationary
[12] Non-Stationary
[13] Variable Learning Automata

اتوماتاهاي يادگير واكنشي: با وجود اينكه اتوماتاهاي يادگير استاندارد توضيح داده شده در بخش قبل قادر است در محيطهاي ايستا كه اطلاعات مربوط به مساله از ابتداي حل آن مشخص مي باشند و در طول حل مساله تغييري در داده هاي مساله بوجود نمي آيد، به جواب بهينه‌اي برسد اما در محيطهاي پويا كه اطلاعات مساله در حين اجراي آن در حال تغيير است مناسب نمي باشند زيرا احتمال انتخاب تعدادي از اقدامها بر اثر انتخاب نشدن آنها بدليل واقع نشدن در مسير بهينه، و در نتيجه جريمه شدن آنها رفته رفته كاهش يافته و به صفر ميل خواهد كرد و در نتيجه چنين اقدامهايي شانس انتخاب مجدد خود را از دست ميدهند. در اين نكته كه چنين اقدامهايي در شرايط جاري، براي رسيدن به جواب بهينه مناسب نمي باشند ترديدي نيست اما ممكن است بر اثر تغييري در محيط به اقدامهاي موثري تبديل شده و قسمتي از جواب بهينه بشوند. پس در محيطهاي پويا بايستي از الگوريتم يادگيري استفاده كرد كه قادر به تشخيص تغييرات محيط بوده و عكس العمل مناسبي در قبال اين تغييرات از خود نشان دهد. براي اين منظور از نوعي اتوماتاي يادگير به نام اتوماتاي يادگير واكنشي استفاده مي كنيم. در اين اتوماتاي يادگير قانون بروز رساني بردار احتمال اقدام ها بگونه اي است كه از كمتر شدن احتمال اقدامها از يك مقدار معين جلوگيري كرده و در نتيجه هر اقدامي به تعداد نامتناهي شانس انتخاب شدن خواهد داشت و الگوريتم قادر به عكس العمل مناسب در برابر تغييرات خواهد بود.اين ويژگي با عملگر جهش در الگوريتمهاي يادگيري تكاملي كه براي گريز از كمينه محلي بكار مي رود، قابل مقايسه است (در حقيقت بسياري از مدلهاي يادگيري تكاملي تقريبا واكنشي مي باشند). مي دانيم اگر در محيط ايستا نرخ يادگيري الگوريتم به a مقداردهي شود، در اينصورت الگوريتم پس از تعدادي تكرار به جواب بهينه همگرا خواهد شد. ولي در محيطهاي پويا نياز به استراتژيي داريم تا احتمال انتخاب هيچ اقدامي از يك مقدار معين كمتر نشود براي اين منظور ترتيبي اتخاذ مي كنيم كه احتمال انتخاب هيچ اقدامي از $a/2$ كمتر نشود در اينصورت هر اقدامي بمقدار نامتناهي شانس انتخاب شدن خواهد داشت. با در نظر گرفتن نكات ذكر شده قانون بروزرساني اتوماتاي يادگير واكنشي به شرح زير تعريف مي شود:

$$p_i(n+1) = p_i(n) + a\sum_{j\neq i}\delta_j(n)p_j(n)$$
$$p_j(n+1) = p_j(n) - a\delta_j(n)p_j(n) \qquad \forall j \neq i$$

و

$$\delta_j(n) = \min\left[1, \frac{p_j(n) - a/2}{ap_j(n)}\right]$$

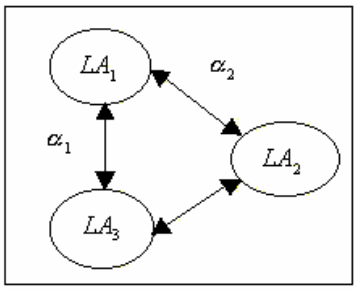
در اين رابطه $a<1$ بوده و هرگاه $p_j \geq a$ براى همه‌ى اقدامهاى اتوماتاى يادگير برقرار باشد در اينصورت قانون بروز رسانى RLA همانند اتوماتاى يادگير استاندارد خواهد شد.

هرگاه $\sum_{i=1}^{r} p_i(n) = 1$ و براى همه‌ى اقدامهاى اتوماتاى يادگير رابطه $p_i(n) \geq a/2$ حفظ شود، گوئيم بردار احتمالى اقدام $p(n)$ معتبر مى باشد. با اندكى بذل توجه ميتوان ديد قانون بروز رسانى الگوريتم يادگير واكنشى، بردارهاى احتمال اقدام معتبر را معتبر نگه مى دارد. در ادامه مقاله حاضر هر جا كه به اتوماتاى يادگير اشاره شود منظور همان اتوماتاى يادگير واكنشى مى باشد.

اتوماتاهاى يادگير واكنشى توزيع شده[14]: يک اتوماتاى يادگير واكنشى توزيع شده (DLA) ، شبكه‌اى از اتوماتاهاى يادگير واكنشى است كه براى حل مساله خاصى با يكديگر همكارى مى نمايند [2]. تعداد اقدامهاى يک اتوماتاى يادگير در DLA برابر تعداد اتوماتاهاى يادگير متصل به اتوماتاى يادگير فوق مى باشد. انتخاب يک اقدام توسط يک اتوماتاى يادگير در شبكه، اتوماتاى يادگير متناظر با اين اقدام را فعال مى سازد. بعنوان مثال در شكل ۲ هر اتوماتاى يادگير داراى دو اقدام مى باشد. انتخاب اقدام α_2 توسط LA_1، اتوماتاى يادگير يادگير LA_3 را فعال خواهد كرد. اتوماتاى يادگير يادگير فعال شده (LA_3)

[14] Distributed Learning Automata

بنوبه‌ی خود یکی از اقدامهای خود را انتخاب می کند که در نتیجه آن یکی از اتوماتاهای یادگیر متصل به آن اتوماتای یادگیر که متناظر با اقدام انتخاب شده می باشد، فعال می شود. در هر زمان فقط یک اتوماتای یادگیر در شبکه فعال خواهد بود. بطور رسمی DLA را میتوان توسط گراف $DLA = (V, E)$ که $V = \{LA_1, LA_2, ..., LA_n\}$ مجموعه اتوماتاهای یادگیر و n تعداد اتوماتاهای یادگیر در DLA و $E \subset V \times V$ مجموعه لبه‌های گراف می باشد، تعریف کرد. لبه (i, j) اقدام j اتوماتای یادگیر LA_i را نشان می دهد. بعبارت دیگر LA_j زمانی فعال خواهد شد که اقدام j اتوماتای یادگیر LA_i انتخاب شود. تعداد اقدامهای اتوماتای یادگیر LA_k ($k = 1,2,...,n$) برابر درجه‌ی خروجی آن گره می باشد[2].



شکل ۲: اتوماتای یادگیر توزیع شده (DLA) با ۳ اتوماتا یادگیر

۳- الگوریتم پیشنهادی

ابتدا شبکه‌ای از اتوماتاهای یادگیر واکنشی که متناظر[15] با گراف ورودی نمونه مساله DTSP داده شده است، ایجاد می شود. در این شبکه هر گره (معادل شهری در مساله داده شده)، یک اتوماتای یادگیر یادگیر واکنشی با ساختار متغیر بوده و هر لبه‌ی خروجی این گره یکی از اقدامهای آن می باشد (لبه‌ای که توسط آن شهر جاری به شهری دیگر متصل می شود). در حقیقت تعداد اقدامهای یک اتوماتا یادگیر معادل تعداد شهرهایی می باشد که میتوان بطور مستقیم از این شهر به آنها رفت (لبه‌ای بین این شهر و شهرهای فوق وجود دارد). خروجی DLA ترتیبی از اقدامهای انتخاب شده توسط اتوماتاهای یادگیر می باشد که مسیر (یا مدار) ویژه‌ای را در گراف نشان می دهد. محیط از طول این مدار برای تولید خروجی استفاده می کند. این خروجی با توجه به مطلوب یا نامطلوب بودن آن، باعث پاداش و یا پنالتی دادن به اقدامهای اتوماتاهای یادگیر واقع بر مدار خاص (مدار هامیلتونی ایجاد شده) می شود.

قبل از اینکه به تشریح الگوریتم بپردازیم نکته‌ای را که در کارایی آن تاثیر بسزایی دارد، توضیح می دهیم. اگر برای انتخاب اقدام اتوماتای یادگیر جاری فقط از بردار احتمال اقدام استفاده شود، جوابهای بدست آمده بر اساس نتایج حاصل از آزمایشها تقریبا غیر قابل قبول بوده و نرخ همگرایی الگوریتم بسیار پایین خواهد بود. جهت رفع این مشکل، یک اتوماتای یادگیر در انتخاب اقدام، علاوه بر استفاده از بردار احتمال اقدام از مقدار عکس فاصله بین دو گره (عکس فاصله بین گره جاری (اتوماتای یادگیری فعال) و گره بعدی، که گره‌های طرفین لبه‌ی انتخاب شده می باشند)، نیز استفاده میکند. استفاده از این مقدار در انتخاب شهر بعدی، بهبود قابل ملاحظه‌ای در کارایی و نرخ همگرایی الگوریتم داشته و جوابهای بهینه و یا خیلی نزدیک به جواب بهینه[16] را تولید میکند. مقدار عکس فاصله بین دو گره j و i در گراف DTSP توسط $W^{-1}(j,i)$ نشان داده میشود.

برای استفاده از این تابع در انتخاب اقدام اتوماتای یادگیر جاری، بردار احتمال اقدام اتوماتای یادگیری j، P^j را بطور موقت طبق روابط زیر به بردار P'^j تغییر می دهیم. و پس از انتخاب اقدام، بردار احتمال اقدام 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 اتوماتای یادگیر j : $P^j = [p_1{}^j, p_2{}^j, ..., p_r{}^j]^T$

[15] Isomorphic
[16] Optimal

بردار احتمال اقدام تغییر یافته اتوماتای یادگیر 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 اتوماتای یادگیر j می باشد و $W^{-1}(j,i)$، عکس فاصله بین دو گره j و i در گراف DTSP بوده و $\beta \geq 1$ اهمیت نسبی فاصله بین دو گره در انتخاب یک اقدام را معین میکند. بعبارتی احتمال انتخاب یالهایی (اقدام) 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 گرفته شدن فاصله بین دو گره، نرخ همگرایی الگوریتم پیشنهادی بمیزان قابل ملاحظه‌ای افزایش می یابد در ضمن اینکه جوابهای نزدیک به بهینه‌ای تولید خواهد شد.

اکنون به توصیف الگوریتم پیشنهادی می پردازیم. در گام نخست یکی از اتوماتاهای یادگیر در شبکه بصورت تصادفی در شبکه DLA انتخاب می شود (بعنوان شهر آغازی تور). این اتوماتای یادگیر یکی از اقدامهای خود را طبق بردار احتمال اقدام تغییر یافته، P'^{j}، انتخاب می کند. انجام این اقدام، اتوماتای یادگیر طرف دیگر لبه که متناظر با اقدام انتخاب شده میباشد را فعال می سازد. با توجه به اینکه در مدار هامیلتونی هر شهر فقط یکبار ملاقات شود بایستی ترتیبی اتخاذ نمود نا الگوریتم هیچ شهری (گره‌ای) را بیش از یکبار انتخاب نکند. برای این منظور اگر اتوماتای یادگیری اقدام k را از لیست اقدامهای خود انتخاب کند، همزمان با آن اتوماتاهای یادگیرغیرفعال اقدام k را در لیست اقدامهای خود غیر فعال[17] می سازند (اما حذف نمی کنند). در ابتدای هر تکرار تمام اقدامهای غیر فعال شده مجددا فعال خواهند شد.

اتوماتای یادگیر فعال شده با استفاده از بردار احتمال اقدام تغییر یافته‌ی خود، اتوماتای یادگیر طرف دیگر لبه انتخاب شده را فعال می سازد. فرآیند انتخاب اقدام و فعال سازی اتوماتای یادگیر متناظر با اقدام انتخاب شده تا ملاقات همه گره‌های (شهرهای) موجود در گراف DTSP و برگشت به شهر آغازین و یا بنابر برخی دلایل که امکان انتخاب اقدام بعدی برای اتوماتای یادگیر جاری وجود نداشته باشد (امکان ایجاد مدار هامیلتونی با توجه به شهرهایی که قبلا انتخاب شده‌اند و یا ساختار گراف نمونه مساله داده شده، وجود نداشته باشد)، تکرار می شود.

پس از پیدا کردن یک تور، طول آن محاسبه شده و با طول بهترین توری که تابحال بدست آمده است مقایسه میگردد. بر اساس نتیجه مقایسه، بردار احتمال اقدام اتوماتاهای یادگیر در DLA بروزرسانی خواهد شد. نحوه‌ی بروزرسانی بردار احتمال اقدام بدینصورت است که اگر طول تور ایجاد شده کوچکتر و یا مساوی طول بهترین توری که تابحال ایجاد شده است باشد، همه اتوماتاهای یادگیر در 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 اتوماتای یادگیر را مجددا یادآوری می نمائیم. بعنوان مثال اگر طول تور ایجاد شده در یک تکرار (t) کوچکتر و یا مساوی طول بهترین توری که تابحال ایجاد شده است، باشد و در این تکرار اتوماتای یادگیر j از مجموعه اقدامهای مجاز خود اقدام i را انتخاب کرده باشد، احتمال انتخاب اقدام i طبق رابطه‌ی زیر افزایش خواهد یافت:

$$p_i(t+1) = p_i(t) + a\sum_{k \neq i}\delta_k(t)p_k(t)$$

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

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

و

$$\delta_k(t) = \min\left[1, \frac{p_k(t) - a/2}{ap_k(t)}\right]$$

[17] disable

در رابطه‌ی بالا پارامتر a نرخ یادگیری[۱۸] و r تعداد اقدامهای اتوماتای یادگیر j می باشد. فرآیند ایجاد تور تا رسیدن به شرط پایانی ادامه میباید. آخرین تور هامیلتونی ایجاد شده (اگر گراف مورد نظر دارای مدار هامیلتونی باشد) تور تقریبی تولید شده توسط الگوریتم میباشد. الگوریتم پیشنهادی در شکل ۴ دیده می شود.

Procedure DTSP
Begin
Initialize probability vector of each automaton
repeat
//Phase 1
InitialCity := Generate a random number between 1 and n (choosing the intial city of tour randomly)
CurrentCity := InitialCity
Disable action 'CurrentCity' of all Unactivated LAs
//Phase 2
for i := 1 to n
if i < n **then**
//choose next node as a sample realization of active LA modified action probability vector, P'^{j}
NextCity := GetNextCity()
//if there's no edge between current city and each of the unselected cities, so current path //could not be hamilton cycle
Disable action 'NextCity' of all Unactivated LAs
if i = n-1 **then**
Enable action 'InitialCity' of LA NextCity
end if
else
NextCity := InitialCity
//if there's no edge between current city and intial city, so current path could not be //hamilton cycle
end if
CurrentCity := NextCity
next i
//Phase 3
compute tour length
if CurrentTourLength < BestTourLength **then**
//Reward selected actions of all LAs along the Tour according to L_{R-I} learning algorithm
for j := 1 to n
// <i> is the selected action of automata <j>

$$p_i(t+1) = p_i(t) + a\sum_{k\neq i}\delta_k(t)p_k(t)$$

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

$$\delta_k(t) = \min\left[1, \frac{p_k(t) - a/2}{ap_k(t)}\right]$$

next j
end if
//Phase4
If the distance (cost) between 2 cities (a , b) changed **then**
For every edge (j , i) **do**
// W is the function that returns distance between two passed arguments
If (W(a,i) < q . MaxDist) Or (W(b,i) < q . MaxDist) Or
(W(a,j) < q . MaxDist) Or (W(b,j) < q . MaxDist) **then**
//smoothing the probability vector values of automata <j>

[18] Learning Rate

$$p_i^j = \begin{cases} p_0.(1+\log(p_i^j / p_0)) & p_i^j > p_0 \\ p_0 \quad otherwise \end{cases}$$

//Adjust probabilities such that $\sum_{i=1}^{r} p_i^j = 1$

$$p_i^j = \frac{p_i^j}{\sum_{k=1}^{r} p_k^j \qquad k \neq i}$$

End If
End For
End If
Enable all the disabled actions of LAs
until (stop condition)
end procedure

شکل ۴: الگوریتم حل TSP با استفاده DLA

این الگوریتم را میتوان بشرح زیر توصیف کرد: ابتدا بردار احتمال اقدام اتوماتاهای یادگیر در DLA که متناظر با گراف DTSP است، مقدار دهی اولیه می شوند. در ابتدای اجرای الگوریتم احتمال انتخاب اقدامهای هر اتوماتای یادگیر با یکدیگر برابر بوده مساوی (تعداد اقدامهای اتوماتاهای یادگیر)/۱ می باشد.

مرحله۱: یکی از گره‌های گراف بصورت تصادفی بعنوان شهر آغازین تور (InitialCity) انتخاب شده و بعنوان گره جاری (CurrentCity) الگوریتم در نظر گرفته می شود و در ضمن این اقدام (CurrentCity) درهمه اتوماتاهای یادگیر موجود در شبکه غیر فعال خواهد شد.

مرحله ۲: در هر تکرار از این مرحله یکی از اقدامهای مجاز [۱۹] اتوماتای یادگیری جاری انتخاب می شود. برای این منظور از تابع ()GetNextCity استفاد شده است (شکل ۵) این تابع با استفاده از بردار احتمال P'^j، یکی از اقدامهای مجاز اتوماتای یادگیری جاری را انتخاب می کند. اگر این تابع قادر به تعیین شهر بعدی برای حرکت نباشد در اینصورت مسیر پیمایش شده تا این مرحله، قابل تبدیل به مدار هامیلتونی نبوده و از آن صرفنظر شده و مجددا از ابتدا اقدام به ایجاد یک گردش جدید خواهیم کرد. اگر الگوریتم قادر به برگشت به شهر آغازین بشود در این صورت یک تور هامیلتونی توسط الگوریتم پیدا شده و به مرحله ۳ میرویم و در غیر اینصورت اگر اتوماتای یادگیری جاری امکان انتخاب اقدام مجاز و یا از آخرین اتوماتای یادگیر امکان بازگشت به گره آغازی وجود نداشته باشد، در اینصورت الگوریتم موفق به ایجاد تور مجاز (مدار هامیلتونی) نشده و به مرحله ۴ می رویم.

مرحله ۳: در این مرحله طول تور ایجاد شده محاسبه و با طول بهترین توری که تابحال بدست آمده است مقایسه می شود و در صورتیکه کوچکتر یا مساوی آن باشد، به اقدامهای انتخاب شده‌ی اتوماتاهای یادگیر در DLA طبق الگوریتم یادگیری L_{R-I} پاداش داده می شود.

مرحله۴: اقدام‌های غیر فعال شده در حین اجرای مرحله ۲، مجددا فعال شده و با تست شرط خاتمه الگوریتم در صورتیکه شرط پایان الگوریتم برقرار نباشد مبادرت به ایجاد توری جدید مینماییم. الگوریتم خاتمه میابد اگر تعداد تورهای ایجاد شده از تعداد از معینی فراتر رفته و یا احتمال مسیر [۲۰] که عبارتست از "حاصلضرب احتمال لبه‌های موجود در مدار انتخاب شده"، از یک حد مشخصی (برای مثال ۰,۹) بیشتر شود.

[19] اقدامهاي غير فعاليكه مجاز به انتخاب آنها نيستيم، اقدامهايي مي باشند كه براي اتوماتاي يادگيرجاري به دلايلي امكان انتخاب آنها وجود ندارد چه در اينصورت تور حاصل داراي شهرهاي تكراري خواهد شد و حائز شرايط براي ايجاد مدار هاميلتوني نخواهد بود.

[20] Path Probability

Function GetNextCity(j: Integer)
// j is the Current Automata
//modify Action Probability vector of <j> (P^j) as:

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

Choose a action as a sample realization of j's modified action probability vector, P'^j
Restore Previous value of P^j
End function

شکل ۵: تابع GetNextCity برای انتخاب یکی از اقدامهای مجاز اتوماتای یادگیری جاری

۴- ارزیابی الگوریتم پیشنهادی

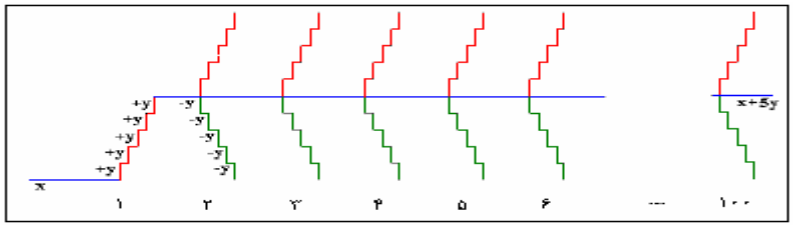
در آزمایشهای انجام گرفته، تراکم ترافیک در لبه‌های گراف مساله در گامهای معین و هر بار فقط در یک لبه بوقوع می پیوندد. الگوریتم پیشنهادی با استفاده از دو نمونه مساله ارزیابی میگردد. یکی از این نمونه‌ها ۲۵ شهری و دیگری یک نمونه ۱۰۰ شهری می باشد.

۴-۱- نمونه مساله ۲۵ شهری

این نمونه شامل ۲۵ شهر می باشد که در شبکه‌ای ۱۰۰*۱۰۰ قرار گرفته‌اند. همه شهرها بصورت تصادفی در تقاطع خطوط شبکه قرار دارند و بنابراین دارای مختصات صحیح می باشند [8]. تراکم ترافیک بین شهرهایی که در مسیر بهینه واقع شده‌اند، بوجود می آید. بنابرین لبه هایی که در آنها ترافیک بوجود خواهد آمد از قبل معین می باشند. وقوع ترافیک بصورت تدریجی بوده و در مجموع ۱۰ واحد به هزینه لبه افزوده خواهد شد که در هر ۵ تکرار، ترافیک یک واحد افزایش خواهد یافت و نیز بهمین صورت ترافیک بوجود آمده در لبه ای، ناپدید خواهد شد. مزیت این روش بالا رفتن شانس الگوریتم برای پاسخگویی به تغییرات کوچک می باشد. زیرا اگر ترافیک در لبه‌ای یک مرتبه بوجود آید، الگوریتم قادر به نمایش قابلیت خود در انتخاب مسیری جدید هنگام وقوع تراکم ترافیک نخواهد بود.

۴-۲- نمونه مساله ۱۰۰ شهری

نمونه‌ی مساله ۱۰۰ شهری با قرار دادن ۱۰۰ شهر بصورت تصادفی در شبکه‌ای ۱۰۰*۱۰۰ ایجاد شده است [8]. پس از تعدادی تکرار الگوریتم، تراکم ترافیک در هر ۵۰ تکرار بوجود خواهد آمد که هزینه لبه را ۵ واحد که هر واحد در یک گام افزوده شده، تغییر خواهد داد. برای مثال فرض می کنیم پس از تعدادی تکرار بدون در نظر گرفتن تراکم ترافیک به جواب بهینه مساله رسیده‌ایم و طول این مدار برابر x می باشد. پس از آن در هر ۵۰ تکرار یک تراکم ترافیک بوجود خواهیم آورد و همزمان با وقوع هر افزایش ترافیک جدید افزایش ترافیک قبلی حذف خواهد شد. در نتیجه اگر هیچ گونه بهینه سازی در هزینه مدار بهینه اولیه انجام نگیرد، طول مدار نمونه مساله پس از وقوع افزایش ترافیک همواره برابر x بعلاوه هزینه تراکم ترافیک بوجود آمده، خواهد شد و چون هر افزایش ترافیک شامل ۵ گام y واحدی است، بجز چند تکرار اولیه الگوریتم برای یافتن مدار بهینه مساله، همیشه $5*y$ افزایش ترافیک وجود خواهد داشت. فرآیند وقوع و ناپدید شدن افزایش ترافیک در شکل ۶ نشان داده شده است.

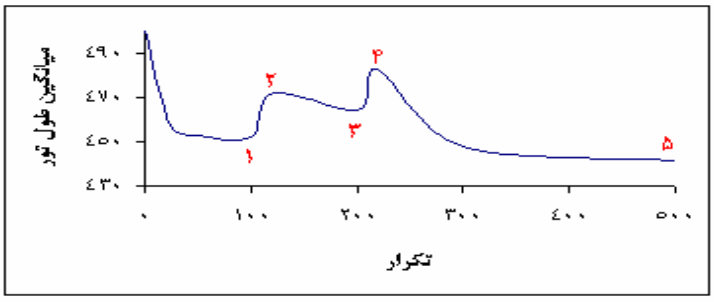


شکل۶: وقوع و ناپدید شدن ترافیک در نمونه مساله ۱۰۰ شهری. طول تور ثابت در طی حل مساله بجز برای دوره کوتاه ابتدایی الگوریتم، برابر x + 5y می باشد.

براساس ترتیب در نظر گرفته شده، اولین وقوع تراکم ترافیک در لبه خروجی شهر ۱، دومین تراکم ترافیک در لبه خروجی شهر ۲ و بهمین ترتیب بوجود آمده و ناپدید خواهند شد. هر آزمایش شامل ۵۰۰۰ تکرار بعلاوه تعدادی تکرار اولیه برای یافتن مدار بهینه، می باشد، از اینرو ۱۰۰ تراکم ترافیک ایجاد خواهد شد. نتایج نشان داده شده میانگین ۱۰ مرتبه اجرای الگوریتم می باشند. واکنشهای الگوریتم پیشنهادی در طی اجرا، نشان دهنده کارایی آن خواهد بود. نتایج با کیفیت بالا، نیازمند واکنشهای مناسب در مقابل تغییرات می باشند. برای مثال لبه‌ای که هزینه آن در حال افزایش است بتدریج با لبه‌ی دیگری که با انتخاب آن هزینه مدار پایین خواهد آمد، جایگزین خواهد شد و لبه‌ای که طول آن در حال کاهش است (در اثر از بین رفتن تراکم ترافیک) رفته رفته برای قرار گرفتن مجدد در تور، مناسبتر می شود.

۴-۳- ارزیابی نمونه مساله ۲۵ شهری

متوسط طول تور مساله ۲۵ شهری با استفاده از الگوریتم پیشنهادی در شکل ۷ دیده می شود. در این شکل تا نقطه ۱، تراکم ترافیکی بوجود نیامده است. دو تراکم ترافیک پس از تکرارهای ۱۰۰ و ۲۰۰ بوجود آمده است که قسمت پویای مساله می باشند. در هر یک از دو نقطه‌ای که تراکم ترافیک بوقوع می پیوندد، در گراف نشان داده شده، نقطه بیشینه‌ای دیده می شود. نقاط بیشینه‌ای که دارای ارتفاع کمتری می باشند، بازیابی سریع مدار بهینه پس از ناپدید شدن تراکم ترافیک را نشان می دهند. هرگاه مساله بدون هیچ تراکم ترافیکی اجرا شود، طول متوسط مدار بهینه پس از ۵۰۰ تکرار برابر ۴۴۱,۴۰ خواهد بود. اگر پس از ایجاد ترافیک، الگوریتم پیشنهادی به چنین نتیجه‌ای برسد، جوابی مطلوب بدست آمده است. پس از وقوع آخرین ترافیک، در حدود ۲۵۰ تکرار دیگر برای رسیدن به چنین جوابی نیاز میباشد.



شکل ۷: میانگین طول تور برای ۱۰ اجرای مختلف در طی ۵۰۰ تکرار(نقاط وقوع ترافیک و ناپدید شدن آنها با شماره‌های ۲ تا ۵ علامت گذاری شده‌اند.)

نتایج الگوریتم پیشنهادی در ۴ نقطه وقوع تراکم ترافیک و ناپدید شدن آن بهمراه نتایج بدست آمده از الگوریتمهای براساس کولونی مورچه ها برای حل مساله فروشنده دوره‌گرد احتمالی [8] در جدول ۱ گزارش شده است. همانطور که نتایج نشان میدهد، الگوریتم پیشنهادی دارای نتایج قابل قبولی در مقایسه با الگوریتمهای کولونی مورچه ها می باشد.

جدول ۱: نتایج الگوریتم پیشنهادی و الگوریتمهای براساس کولونی مورچه، در نقاط وقوع و ناپدید شدن ترافیک برای نمونه مساله ۲۵ شهری مساله فروشنده دوره گرد احتمالی.

نام الگوریتم	۲	۳	۴	۵
الگوریتم پیشنهادی	۴۷۳,۳	۴۶۲,۱	۴۸۴,۰	**۴۴۱,۲**
الگوریتم اصلی ACS[21]	۴۸۵,۲	۴۶۵,۴	۴۹۸,۷	**۴۴۰,۱**
هموار سازی ACS ،q=0.1	۴۷۷,۷	۴۶۳,۰	۴۸۹,۰	۴۴۳,۴
هموار سازی ACS ،q=0.25	۴۷۹,۵	۴۶۵,۰	۴۸۵,۰	۴۴۰,۶
هموار سازی ACS ،q=1	**۴۷۲,۰**	۴۶۱,۸	۴۸۱,۸	۴۴۱,۹
مقدار دهی مجدد ACS	۴۸۳,۶	**۴۶۱,۲**	**۴۸۰,۰**	۴۴۱,۶

[21] Ant Colony Systems

۴-۴- ارزیابی نمونه مساله ۱۰۰ شهری

میانگین طول تور بدست آمده برای مساله ۱۰۰ شهری توسط الگوریتم پیشنهادی بهمراه نتایج گزارش شده برای الگوریتمهای کولونی مورچه برای حل مساله فروشنده دوره‌گرد احتمالی [8]، در جدول ۲ نشان داده شده است.

جدول ۲: میانگین طول تور نمونه ۱۰۰ شهری مساله فروشنده دوره‌گرد احتمالی برای الگوریتم پیشنهادی و الگوریتمهای کولونی مورچه .

نام الگوریتم	میانگین طول تور
الگوریتم پیشنهادی	۸۱۲,۵
الگوریتم اصلی ACS	۸۱۳,۹
هموار سازی q=0.1، ACS	۸۱۱,۸
هموار سازی q=0.25، ACS	۸۱۲,۷
هموار سازی q=1، ACS	۸۱۵,۶
مقدار دهی مجدد ACS	۸۱۵,۵

با توجه به نتایج ارایه شده در این جدول همه الگوریتمها قادر به پاسخگویی به شرایط پویا می باشند. بهترین نتیجه مربوط به الگوریتم هموارسازی محلی می باشند. کارایی الگوریتم پیشنهادی در مقایسه با سایر الگوریتمها در رتبه دوم قرار دارد که با توجه به سرعت نسبتا بالای آن قابل قبول بنظر می رسد.

۵- نتیجه گیری

در این مقاله الگوریتمی براساس اتوماتای یادگیر واکنشی توزیع شده برای حل مساله فروشنده دوره‌گرد پویا ارائه گردید و کارایی آن با استفاده از دو نمونه ۲۵ و ۱۰۰ شهری مورد بررسی قرار گرفت. اتوماتای یادگیر واکنشی توزیع شده بکار گرفته برای حل این مساله، جهت پاسخگویی به پویایی محیط از روش ویژه‌ای برای بروز رسانی بردار احتمال اقدام اتوماتاهای یادگیر، استفاده می کند که در آن از کمتر شدن احتمال انتخاب اقدامهای اتوماتای یادگیر از یک مقدار مشخص جلوگیری می شود. نتایج بدست آمده بر کارایی قابل قبول الگوریتم پیشنهادی دلالت داشته و تاییدی دیگری است بر به کارگیری اتوماتاهای یادگیر برای حل مسایل بهینه سازی در محیطهای پویا.

مراجع

[1] H. N. Psarafis, , “Dynamic vehicle routing problems,” in Vehicle Routing: Methods and Studies, B.L. Golden and A.A. Assad (eds), 223-248, Elsevier Science Publishers 1988.
[2] M. R. Meybodi and H. Beigy, “Solving Stochastic Shortest Path Problem Using Distributed Learning Automata”, Proceedings 6th Annual CSI Computer Conference, University of Isfahan’s Computer Engineering Department, 2001.
[۳] X. Lu, A.C. Regan and S. Irani, “The M/G/1 queue with switchover costs: An examination of alternative heuristics,” Queueing Systems, under review (2001).
[4] S. Lakshmivarahan, Learning Algorithms: Theory and Applications. New York: Springer-verlag, 1981.
[5] M. R. Meybodi and S. Lakshmivarahan, “On a Class of Learning Algorithms which have Symetric Behavior under Success and Failer”, pp. 145-155. Lecture Notes in Statistics, Berlin: Springer-Verlag, 1984.
[6] P. Mars, J. R. Chen, and R. Nambir, Learning Algorithms: Theory and Applications in Signal Processing, Control, and Communication. CRC Press Inc., 1996.
[7] K. S. Narendra and K. S. Thathachar, Learning Automata: An Introduction. New York: Prentice-Hall, 1989.
[8] J. Zwiers, A. Nijholt, M. Poel, M. Snoek, “Ant Systems for Dynamic Problems: The TSP Case – Ants caught in a traffic jam,” Master’s thesis, University of Twente, The Netherlands, August 2001
[9] M. Guntsch, J. Branke, M. Middendorf , H. Schmeck, “ACO Strategies for Dynamic Optimisation Problems,” In [To be published], 2001
[10] M. Guntsch, M. Middendorf , H. Schmeck, “An Ant Colony Optimization Approach to Dynamic TSP,” Proceedings of GECCO 2001, 2001