یک الگوریتم تکاملی تخمین توزیع جدید با استفاده از اتوماتای یادگیر

رضا رستگار و محمدرضا میبدی

چکیده: در سالهای اخیر رویکرد جدیدی به منظور حل مشکلات الگوریتمهای تکاملی به ویژه الگوریتم‌های ژنتیکی مورد توجه محققین قرار گرفته است. این رویکرد مبتنی برایجاد مدلهای احتمالاتی از ژنوم‌ها و اجزای سازنده آنها می‌باشد. تاکنون الگوریتم‌های متنوعی بر این اساس ارائه شده‌اند که اگر چه برخی از سادگی الگوریتم‌های ژنتیکی برخوردار نیستند، اما در حل مسائل با موفقیت بیشتری روبرو بوده‌اند. در این مقاله رهیافت دیگری از این الگوریتمها را بر اساس اتوماتای یادگیر معرفی و مورد بررسی قرار می‌دهیم. در این رهیافت مدل احتمالاتی اجزای سازنده مسئله به وسیله اتوماتای یادگیر و بر اساس ژنوم‌های نسل تولید شده تخمین زده می‌شود. الگوریتم پیشنهادی بسیار ساده و برای مسائل مورد بررسی در این مقاله دارای کارایی خوبی می‌باشد.

کلید واژه: **الگوریتم تکاملی، الگوریتم تخمین توزیع، اتوماتای یادگیر.**

۱- مقدمه

نیاز به جستجو در حل مسائل کاربردی امری غیر قابل اجتناب و در عین حال دشوار است. به همین جهت تعداد زیادی از الگوریتم‌های جستجو با فلسفه‌ها و دامنه استفاده متفاوت بوجود آمده‌اند. این الگوریتم‌های جستجو را می‌توان به دو دسته کلی جستجوهای کامل و جستجوهای مکاشفه‌ای تقسیم کرد. فرق اساسی بین الگوریتم‌های این دو دسته به این صورت است که در جستجوهای کامل، تمام فضای جستجو به طور کامل مورد جستجو و ارزیابی قرار می‌گیرد تا جواب مورد نظر یافته شود، در حالیکه در جستجوهای مکاشفه‌ای تنها بخشی از فضا که احتمال یافتن جواب در آن بیشتر است، مورد توجه قرار می‌گیرد. جستجوهای مکاشفه‌ای به دو دسته الگوریتم‌های قطعی و غیرقطعی تقسیم پذیرند. ویژگی اصلی الگوریتم‌های قطعی در این است که تحت شرایط یکسان، جوابهای یکسان می‌دهند. از جمله این الگوریتم‌ها می‌توان به تپه‌نوردی[1] اشاره کرد. ایراد اساسی این الگوریتم‌ها احتمال گیر افتادن در مینیمم‌های محلی می‌باشد. در مقابل، الگوریتم‌های مکاشفه‌ای غیر قطعی با استفاده از احتمالات و جستجوهای تصادفی، در شرایط یکسان جوابهای متفاوتی بدست می‌آورند. همینطور در صورت افتادن در مینیمم‌های محلی، از آنها می‌گریزند. الگوریتم‌های مکاشفه‌ای غیر قطعی را بر اساس تعداد جوابهایی که در هر تکرار بررسی و ذخیره می‌کنند به دو دسته تقسیم می‌کنند. بعضی مانند تابکاری فلزات تنها یک جواب را در هر تکرار مورد بررسی قرار داده و ذخیره می‌کنند. بعضی دیگر در هر تکرار دسته‌ای از جوابها را ذخیره می‌کنند، که به این الگوریتم‌ها، الگوریتم‌های مبتنی بر جمعیت[2] می‌گویند. دسته بزرگی از این الگوریتم‌ها را الگوریتم‌های ژنتیکی تشکیل می‌دهد. الگوریتم‌های ژنتیکی کلاسیک به دلیل سادگی و عدم نیاز به معادلات دیفرانسیل پیچیده در حل مسائل پیچیده با فضای جستجوی غیر هموار مورد استفاده قرار می‌گیرند. این الگوریتم‌ها بر پایه انتخاب و ترکیب مجدد[3] جوابهای مجاز مسئله کار می‌کنند. مجموعه جوابهای تولید شده در هر دوره الگوریتم را یک نسل و هر کدامها از جوابها را یک ژنوم یا کروموزوم می‌نامند. هر ژنوم ترکیبی از متغیرهای مسئله است. در بعضی مسائل متغیرهای مسئله به هم وابسته و در بعضی دیگر از هم مستقل هستند. ولی همواره ارتباط و تبادل اطلاعات بین ژنوم‌ها از طریق انتخاب و ترکیب مجدد ژنوم‌ها در یک نسل صورت می‌پذیرد. این جابجایی اطلاعات سبب می‌شود تا جوابهای جزئی[4] با یکدیگر ترکیب و احتمالاً جوابهایی با کیفیت بالاتر بدست آیند. اما با تمام ویژگیهای مثبتی که در الگوریتم ژنتیکی استاندارد وجود دارد، این الگوریتم تنها در مواقعی که متغیرها از هم مستقل و یا در فاصله کمی از هم در ژنوم قرار گرفته باشند، کارایی مناسبی دارد [۱]. به عبارت دیگر رفتار الگوریتم ژنتیکی وابستگی شدیدی به پارامترهایی از جمله نحوه تعریف عملگرهای جهش و تولید نسل، احتمال جهش و تولید نسل، اندازه جمعیتها و تعداد نسلهای تولید شده دارد. به همین دلیل گاهی در ترکیب ژنوم‌ها با یکدیگر نه تنها بهبودی در کیفیت در جوابها حاصل نمی‌شود، بلکه الگوریتم در نقاط بهینه محلی به دام می‌افتد. به منظور رفع این مشکل نسخه‌های متعددی از الگوریتهای ژنتیکی بوجود آمده‌اند، که می‌توانیم آنها را به سه دسته تقسیم کنیم. دسته اول الگوریتم‌هایی هستند که در آنها عملگرهای ژنتیکی همزمان با فرایند تکامل به صورت خودکار بهبود می‌یابند [۲]. در دسته دوم به طور مشابه نحوه بازنمایی مسئله همزمان با تکامل متحول می‌شود [۲]. دسته سوم الگوریتم‌ها نیز با ساخت مدلهای احتمالاتی از متغیرها سعی در افزایش سرعت همگرایی به سمت جواب مناسب می‌کنند [۱] تا [۱۳]. دسته سوم با نام الگوریتم‌های تخمین توزیع شناخته می‌شوند که در آن ژنوم‌های جدید به جای استفاده از عملگرهای مرسوم در الگوریتم ژنتیکی با استفاده از توزیع احتمالاتی تمام راه حل‌های مجاز نسل قبل ساخته می‌شوند. الگوریتم تخمین توزیع برای اولین بار در [۲] در رده الگوریتم‌های تکاملی مطرح شد و از زمان طرح آن الگوریتم‌های متنوعی از آن بوجود آمده‌اند.

در مراجع [۱۴] و [۱۵] ترکیبی از الگوریتم‌های ژنتیکی با اتوماتای یادگیر ارائه شده‌اند. در الگوریتم ارائه شده در [۱۵] با نام stGA[5] تمام جمعیت ژنوم به عنوان یک اتوماتای یادگیر تصادفی و هر عضو فضای ژنومی به عنوان یک عمل اتوماتای یادگیر در نظر گرفته می‌شود

این مقاله در تاریخ ۳۰ آذر ماه ۱۳۸۲ دریافت و در تاریخ ۱۲ آذر ماه ۱۳۸۳ بازنگری شد.

رضا رستگار، دانش آموخته کارشناسی ارشد مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر، تهران.

محمدرضا میبدی، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر، تهران (e-mail: meybodi@ce.aut.ac.ir).

1. Hill Climbing
2. Population-Based Algorithm
3. Recombination
4. Partial Solution
5. Stochastic Genetic Algorithm

به طوری که ارزش هر ژنوم با احتمال آن سنجیده می‌شود. ارزیابی هر ژنوم معادل انتخاب عمل توسط اتوماتای یادگیر و دریافت پاسخ محیط آن می‌باشد. هنگامی که یک عملگر تولید مثل بین دو ژنوم اتفاق می‌افتد، احتمال فرزند به دست آمده معادل میانگین احتمال والدین قرار می‌گیرد. این الگوریتم دارای یک مکانیزم برای حذف ژنوم‌های تکراری در جمعیت ژنوم‌ها می‌باشد. این الگوریتم تنها به مسائل ساده با پیچیدگی کم و دارای ژنوم‌های با طول کم، قابل اعمال است. در الگوریتم GLA[1] که در [۱۴] ارائه گردیده است، رهیافت دیگری دنبال شده است. به این ترتیب که در این الگوریتم، جمعیت ژنوم‌ها با جمعیت بردارهای احتمالاتی جایگزین شده است. در هر تولید نسل بر اساس هر کدام از بردار احتمالها یک رشته تولید می‌شود و ارزیابی رشته تولید شده با استفاده از تابع ارزیابی انجام می‌شود. عمل جفت‌گیری بین دو بردار احتمال مانند الگوریتم ژنتیکی استاندارد صورت می‌پذیرد. با این تفاوت که به جای جابجا شدن بیتها بین ژنوم‌ها، احتمالها جابجا می‌شوند. عمل جهش ژنتیکی نیز با جایگزینی مقدار احتمال بیت ژنوم به مقدار اولیه (در حالتی که از اتوماتایی با دو عمل استفاده شود این مقدار ۰/۵ است.) انجام می‌پذیرد. این الگوریتم به دلیل جابجا کردن مقادیر احتمالها که مقدار حقیقی دارند، سربار محاسباتی بیشتری را نسبت به الگوریتم ژنتیکی وارد می‌کند.

در این مقاله یک الگوریتم جدید از الگوریتم‌های تخمین توزیع که مبتنی بر اتوماتای یادگیر می‌باشد، معرفی می‌گردد. در این الگوریتم به منظور ایجاد مدل احتمالاتی متغیرهای تشکیل دهنده ژنوم، از اتوماتای یادگیر استفاده می‌کنیم. این الگوریتم بسیار ساده و پیاده‌سازی نرم افزاری و سخت افزاری آن سهل الوصول است. همچنین بر خلاف الگوریتم GLA و بسیاری از الگوریتم‌های تخمین توزیع به سادگی در فضاهای غیر باینری قابل استفاده است.

ادامه مقاله به این صورت سازماندهی شده است. در بخش ۲، فرم کلی الگوریتم‌های تخمین توزیع بررسی می‌گردد و انواع متفاوت از الگوریتم‌های تخمین توزیع ارائه شده در حوزه فضای گسسته را در قالب یک دسته‌بندی کلی و به صورت خلاصه بیان می‌گردد. بخش ۳ به مروری خلاصه بر اتوماتای یادگیر و به خصوص اتوماتای یادگیر با ساختار متغیر اختصاص دارد. سپس در بخش ۴، الگوریتم پیشنهادی را با تفضیل بیشتری بیان خواهیم کرد. در ادامه در قسمت ۵ چند مسئله خطی و ترکیبی را به منظور بررسی کارایی الگوریتم پیشنهادی معرفی می‌کنیم. در قسمت ۶، نتایج به دست آمده و مقایسه آنها با نتایج الگوریتم ژنتیکی ساده و الگوریتم تخمین توزیع UMDA[2] و PBIL[3] ارائه می‌شود. بخش پایانی نیز نتیجه‌گیری می‌باشد.

۲- الگوریتم تخمین توزیع

در [۲] دسته‌ای از الگوریتم‌های مکاشفه‌ای غیر قطعی مبتنی بر جمعیت با نام الگوریتم‌های تخمین توزیع ارائه شده‌اند که همانند الگوریتم‌های ژنتیکی نیازمند به فضای جستجوی هموار و معادلات دیفرانسیل پیچیده نمی‌باشند و علاوه بر این بسیاری از مشکلات الگوریتم‌های ژنتیکی را حل نموده‌اند. در الگوریتم‌های تخمین توزیع با ساخت یک مدل احتمالاتی از اجزای سازنده ژنوم، سرعت پیشروی به سوی جواب بهینه مسئله افزایش می‌یابد. در این الگوریتم‌ها جمعیتهای جدید با استفاده از عملگرهای جهش و تولید نسل به وجود نمی‌آیند. بلکه ژنوم‌های جدید، بر اساس توزیع احتمالاتی تخمین زده شده بر اساس ژنوم‌های انتخاب شده از نسلهای قبل، نمونه برداری و ساخته می‌شوند. در واقع تخمین همین تابع توزیع مشکل‌ترین بخش الگوریتم‌های تخمین توزیع است. قالب کلی همه الگوریتم‌های تخمین توزیع به صورت زیر است :

قدم ۱- در ابتدا نخستین جمعیت، D_0، با تعداد N ژنوم تشکیل می‌شود. معمولا تولید N ژنوم مذکور، با فرض توزیع یکنواخت به روی هریک از متغیرها صورت می‌پذیرد. سپس هر یک از ژنوم‌ها ارزیابی می‌شوند.

قدم ۲- یک تعداد مشخص، $Se \le N$، ژنوم از میان جمعیت حاضر، D_l، بر اساس یک معیار تعریف شده (معمولاً بهترین‌ها بر اساس تابع ارزیابی مسئله مورد نظر) انتخاب می‌شوند که با D_l^{Se} مشخص می‌شود.

قدم ۳- یک مدل احتمالاتی n بعدی که نشان دهنده توزیع n متغیر موجود راه حل می‌باشد، تخمین زده می‌شود.

قدم ۴- در پایان یک جمعیت جدید متشکل از N ژنوم جدید با استفاده از توزیع احتمالاتی به دست آمده در مراحل قبل، ایجاد می‌شود.

قدم‌های ۲ تا ۴ الگوریتم فوق تا برقراری شرط خاتمه تکرار می‌شوند. شرایط خاتمه متفاوتی مانند تولید تعداد معینی نسل، انجام تعداد معینی ارزیابی، یکنواختی در چندین نسل پایانی و ثابت ماندن تقریبی ارزش ژنوم‌ها در چند نسل پیاپی را می‌توان در نظر گرفت. در شکل ۱ نمایی از نحوه عملکرد الگوریتم تخمین توزیع داده شده است [۱۶].

اگر فرض کنید $X_i, i = 1,...,n$ یک متغیر تصادفی و x_i یک مقدار برای X_i باشد. در این صورت $X = (X_1,...,X_n)$ یک متغیر تصادفی n بعدی است که می‌تواند $x = (x_1,...,x_n)$ را به عنوان مقدار خود پذیرد. مدل گرافیکی X دارای دو جزء تشکیل دهنده ساختار و یک مجموعه از چگالی‌های احتمالاتی محلی است. به طوری که ساختار S به صورت یک گراف جهت دار بدون دور تعریف می‌شود. این گراف نشان دهنده ساختار وابستگی بین متغیرهای X_i است. اگر در این ساختار، Pa_i^S به معنای متغیرهای والد متغیر X_i باشد، آنگاه متغیر X_i از تمام متغیرهای $\{X_1,...,X_n\} \setminus Pa_i^S$ مستقل بوده و بنابراین می‌توانیم تابع چگالی عمومی توام[4] $\rho(X = x)$ را به صورت زیر بنویسیم:

$$\rho(X = x) = \rho(x_1,...x_n) = \prod_{i=1}^{n} \rho(x_i \mid Pa_i^S) . \qquad (۱)$$

از طرفی با فرض این که چگالی‌های احتمالاتی محلی به مجموعه پارامترهای $\theta_S \in \Theta_S$ وابسته است می‌توانیم معادله (۱) را به صورت زیر بنویسیم:

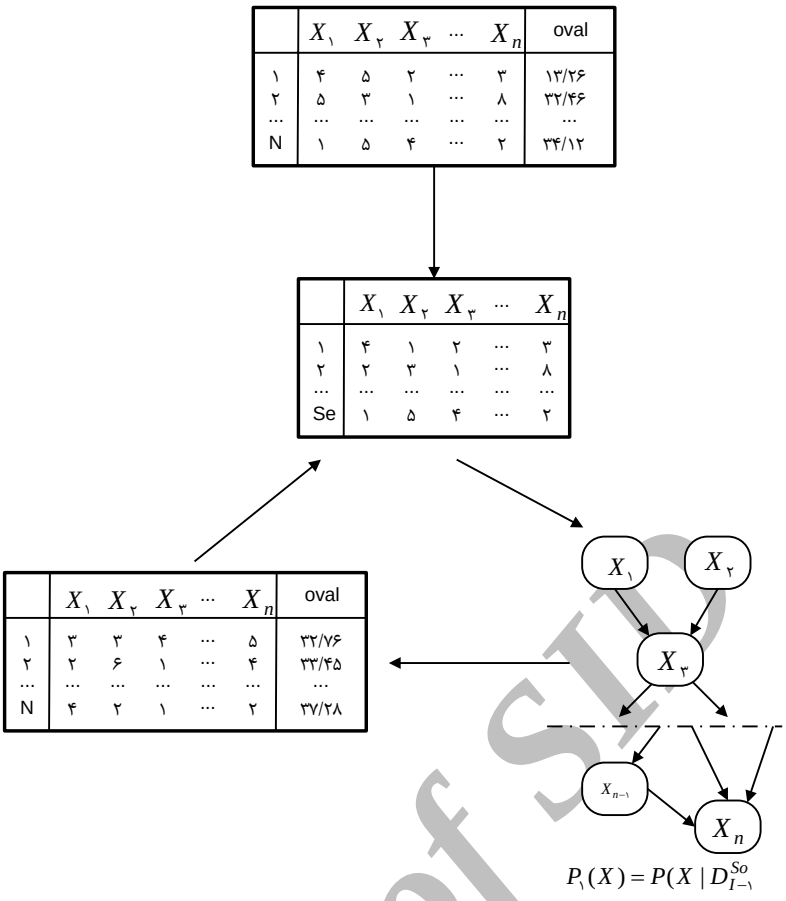
$$\rho(X \mid \theta_S) = \prod_{i=1}^{n} \rho(x_i \mid Pa_i^S, \theta_i) \qquad (۲)$$

که $\theta_S = (\theta_1,...,\theta_n)$ می‌باشد. بنابراین مدل گرافیکی احتمالاتی X را می‌توان به صورت $M = (S, \theta_S)$ بیان کرد [۱۷]. بر اساس نوع ارتباطات بین متغیرها مدل‌های گرافیکی احتمالاتی به سه گروه مستقل، با ارتباطات دوتایی و با ارتباطات چندتایی تقسیم می‌شوند.

در گروه اول، متغیرها از هم مستقل هستند و بنابراین احتمال یک نمونه X، برابر حاصلضرب احتمال متغیرهای آن است. به عبارت دیگر داریم:

$$P(X = x) = \prod_{i=1}^{n} P(X_i = x_i) \qquad (۳)$$

1. Genetic Learning Automata
2. Univariate Marginal Distribution Algorithm
3. Population-Based Incremental Learning
4. Joint Generalized Probability Density Function



شکل ۱: نحوه عملکرد الگوریتم تخمین توزیع.

این مدل، ساده‌ترین مدل ممکن است و با قبول درصدی از خطا می‌توان از آن برای حل مسائل بهینه‌سازی غیرخطی استفاده کرد. از الگوریتم‌های این دسته می‌توان به UMDA [۸]، PBIL[۱] [۱] و [۴] و CGA[۲] [۶] و BSC[۳] [۱۸] اشاره کرد. دو الگوریتم آخری فقط برای ژنوم‌های با متغیرهای دودویی (که تنها مقادیر صفر و یک را می‌پذیرند) به کار می‌روند.

نادیده گرفتن وابستگی بین متغیرها در بسیاری از مسائل بهینه‌سازی از واقعیت دور است. با در نظر گرفتن وابستگی بین متغیرها، اولین و ساده‌ترین نوع ارتباطات، ارتباطات دوتایی است. الگوریتم‌هایی که در این دسته قرار می‌گیرند، اغلب با استفاده از الگوریتم‌های حریصانه، مدلهای احتمالاتی محلی را استخراج می‌کنند. از الگوریتم‌های این گروه می‌توان به MIMIC[۴] [۵]، COMIT[۵] [۳] و BMDA[۶] [۹] اشاره کرد.

در نهایت مدل سوم که در واقع عمومی‌ترین مدل گسسته است، قادر به تخمین صحیح توزیع متغیرها می‌باشد. چند نکته در این مدل حائز اهمیت است. از آنجا که این مدل احتمالاتی پیچیده است، تخمین آن کاری زمان‌بر است. بنابراین باید به گونه‌ای کارایی فرایند استخراج مدل بهبود بیابد. از طرفی باید آستانه مناسبی برای خلاصی از ارتباطات غیر ضروری بین متغیرها، به کار گرفت. از جمله الگوریتم‌هایی که در این گروه جای می‌گیرند، می‌توان به FDA[۷] [۷]، EBNA[۸] [۱۶]، BOA[۹] [۱۱] و ECGA[۱۰] [۱۳] اشاره کرد. از آنجایی که الگوریتم ارائه شده در این مقاله جزو گروه الگوریتم‌های مدل اول محسوب می‌شود، در ادامه این بخش به توضیح الگوریتم‌های مدل اول می‌پردازیم.

در PBIL جمعیت ژنوم‌ها با استفاده از یک بردار احتمال $P_l(x) = (P_l(x_1),...,P_l(x_n))$ بازنمایی می‌شود. به این ترتیب که $P_l(x_i)$ احتمال قرار گرفتن مقدار ۱ در متغیر i–ام نسل l–ام می‌باشد. در هر نسل از بردار احتمال N ژنوم نمونه‌گیری شده و Se تا از بهترین آنها انتخاب می‌شوند. بردار احتمال بر اساس ژنوم‌های انتخابی و با استفاده از یک قانون مشابه قانون هب[۱۱] به روز درآورده می‌شود [۱]. یعنی:

$$P_{l+1}(x_i) = (1-\alpha)P_l(x_i) + \alpha\left(\sum_{k=1}^{Se} x_{i,k:N}^{l} \Big/ Se\right) \qquad (۴)$$

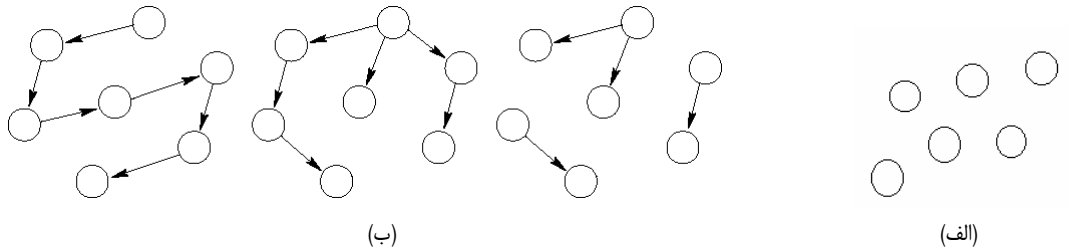
به صورتی که $\alpha \in (0,1]$ و $x_{i,k:N}^{l}$ برابر با مقدار x_i در k–امین ژنوم انتخاب شده در نسل l–ام می‌باشد.

در UMDA توزیع احتمالاتی توأم[۱۲] ژنوم با استفاده از فراوانی حاشیه‌ای[۱۳] هر کدام از متغیرهای ژنوم به صورت زیر بدست می‌آید.

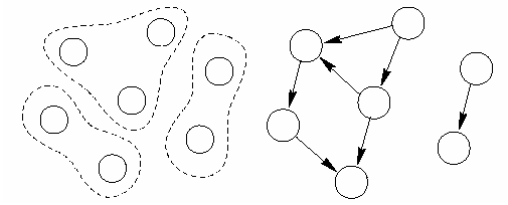
$$P_l(x_i) = \sum_{j=1}^{Se} \delta_j(X_i = x_i \mid D_{l-1}^{Se}) \Big/ Se \qquad (۵)$$

به طوری که اگر متغیر X_i ژنوم j–ام مجموعه D_{l-1}^{Se} مقدار x_i داشته باشد، مقدار $\delta_j(X_i = x_i \mid D_{l-1}^{Se})$ یک و در غیر این صورت مقدار آن صفر است. نشان داده شده است که UMDA رفتاری همانند الگوریتم ژنتیکی ساده را از خود نشان می‌دهد [۱۲].

1. Population-Based Incremental Learning
2. Compact Genetic Algorithm
3. Bit Based Simulated Crossover
4. Mutual Information Maximization for Input Clustering
5. Combining Optimizers with Mutual Information Tress
6. Bivariate Marginal Distribution Algorithm
7. Factorized Distribution Algorithm
8. Estimation of Bayesian Network Algorithm
9. Bayesian Optimization Algorithm
10. Extended Compact Genetic Algorithm
11. Hebbian law
12. Joint Probability Distribution
13. Marginal Frequency

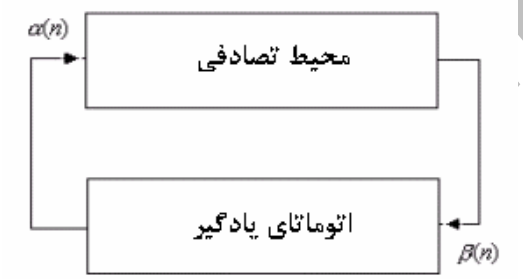


(الف) (ب)



(ج)

شکل ۲: مدل گرافیکی متغیرها ،(الف) مستقل، (ب) با ارتباط دوتایی، (ج) با وابستگی چندتایی.



شکل ۳: ارتباط بین اتوماتای یادگیر و محیط.

در الگوریتم CGA بردار احتمال ژنوم‌ها، $P_l(x) = (P_l(x_1),..., P_l(x_n))$، را برابر (۰/۵،...،۰/۵) قرار می‌دهیم. دو ژنوم بر اساس بردار ساخته می‌شود و بر اساس تابع ارزیابی مرتب می‌کنیم. ژنوم با ارزش]تر را برنده[1] و ژنوم دیگر را بازنده[2] می‌نامیم. سپس بر اساس رابطه زیر بردار احتمال را به روز در آورده و این فرایند را تا همگرا شدن بردار احتمال ادامه می‌دهیم [۶].

$$P_{l+1}(x_i) = \begin{cases} P_l(x_i) + 1/n & loser[i] \neq winner[i] \ \&\ winner[i] = 1 \\ P_l(x_i) - 1/n & loser[i] \neq winner[i] \ \&\ winner[i] = 0 \\ P_l(x_i) & loser[i] = winner[i] \end{cases} \quad (۶)$$

درالگوریتم BSC برای هر مقدار ممکن متغیر X_i یک احتمال در نظر گرفته می‌شود. برای یک مسئله در فضای $\{0و1\}$، $p_l(x_i)$ دارای یک توزیع برنولی می‌باشد.

$$p_l(x_i) \propto \sum_{r=1}^{N} \delta(I_{rl}^i) e(I_{rl}^i) \quad (۷)$$

به طوری که $\delta(.)$ تابع ضربه کرانکر، I_{rl}^i نشان دهنده مقدار i–امین ژن در ژنوم r–ام در جمعیت D_1 و $e(I_{rl}^i)$ ارزش r–امین ژنوم جمعیت D_1 می‌باشد [۱۸] و [۱۹]. این الگوریتم‌ها در مواقعی که مسائل مورد نظر دارای متغیرهای کاملاً مستقل باشند، سریعاً به جواب بهینه دست پیدا می‌کنند. اما هر چه به وابستگی متغیرها افزوده شود از کارایی این الگوریتم‌ها کاسته می‌شود.

۳- اتوماتای یادگیر

اتوماتای یادگیر یک ماشین با حالات محدود[3] [۲۰] است که تعداد محدودی عمل را می‌تواند انجام دهد. هر عمل انتخاب شده توسط محیطی احتمالی ارزیابی می‌گردد و پاسخی به اتوماتای یادگیر داده می‌شود. اتوماتای یادگیر از این پاسخ استفاده نموده و عمل بعدی را انتخاب می‌کند. در طی این فرآیند، اتوماتای یادگیر یاد می‌گیرد که چگونه بهترین عمل را انتخاب نماید. شکل ۳ ارتباط بین اتوماتای یادگیر و محیط را نشان می‌دهد.

محیط را می‌توان توسط سه تایی $E \equiv \{\alpha, \beta, c\}$ نشان داد که در آن $\alpha = \{\alpha_1,..., \alpha_r\}$ مجموعه ورودیها، $\beta = \{\beta_1,..., \beta_m\}$ مجموعه خروجیها و $c = \{c_1,..., c_r\}$ مجموعه احتمالهای جریمه می‌باشد. هرگاه β مجموعه دو عضوی باشد محیط از نوع P می‌باشد. در چنین محیطی $\beta_1 = 1$ به عنوان جریمه و $\beta_2 = 0$ به عنوان پاداش در نظر گرفته می‌شود. در محیط Q Q، $\beta(n)$ می‌تواند به طور گسسته یک مقدار از مقادیر محدود در فاصله [۰،۱] و در محیط از نوع S، متغیر تصادفی در فاصله [۰،۱] باشد. c_i احتمال اینکه عمل α_i نتیجه نامطلوب داشته باشد، می‌باشد. در محیط ایستا مقادیر c_i بدون تغییر می‌مانند، حال آنکه در محیط غیر ایستا این مقادیر در طی زمان تغییر می‌کنند. اتوماتای یادگیر به دو دسته با ساختار ثابت و ساختار متغیر تقسیم می‌شوند. از آنجا که در این تحقیق بیشتر تمرکز بروی اتوماتای با ساختار متغیر است، در ادامه این بخش به این نوع خواهیم پرداخت.

1. Winner
2. Loser
3. Finite State Machine

با توجه به مدل مارکوفی[1] اتوماتای یادگیر، این یادگیرنده‌ها به دو دسته ارگودیک[2] و جذب شونده تقسیم می‌شوند. اتوماتای ارگودیک به یک توزیع مستقل از حالت اولیه همگرا می‌شود. به همین دلیل در محیط‌های غیر ایستا این اتوماتاها کاربرد بیشتری دارند. اتوماتای با نقاط جذب شونده بعد از چندین مرحله در یک حالت قفل شده و از آن بیرون نمی‌آید. و همین ویژگی سبب می‌شود تا کاربرد آنها به محیط‌های ایستا یا ناایستا با تغییرات بسیار کم، محدود شود. اتوماتای یادگیر با ساختار متغیر توسط چهارتایی $\{\alpha,\beta,p,T\}$ نشان داده می‌شود که در آن $\alpha=\{\alpha_1,...,\alpha_r\}$ مجموعه عملهای اتوماتا، $\beta=\{\beta_1,...,\beta_m\}$ مجموعه ورودیهای اتوماتا، $p=\{p_1,..,p_r\}$ بردار احتمال انتخاب هریک از عملها و $p(n+1)=T[\alpha(n),\beta(n),p(n)]$ الگوریتم یادگیری می‌باشد. الگوریتم زیر یک نمونه از الگوریتم‌های یادگیری خطی است. فرض کنید عمل α_i در مرحله nام انتخاب شود. – پاسخ مطلوب

$$\begin{aligned} p_i(n+1) &= p_i(n)+a[1-p_i(n)] \\ p_j(n+1) &= (1-a)p_j(n) \quad \forall j \ \ j\neq i \end{aligned} \tag{۸}$$

– پاسخ نامطلوب

$$\begin{aligned} p_i(n+1) &= (1-b)p_i(n) \\ p_j(n+1) &= (b/r-1)+(1-b)p_j(n) \quad \forall j \ \ j\neq i \end{aligned} \tag{۹}$$

در روابط (۵) و (۶)، a پارامتر پاداش و b پارامتر جریمه می‌باشند. با توجه به مقادیر a و b سه حالت زیر را می‌توان در نظر گرفت.

حالت ۱ زمانی که a و b با هم برابر باشند، الگوریتم را L_{RP} می‌نامیم. این اتوماتا از نوع اتوماتای ارگودیک می‌باشد. حالت ۲ زمانی که b از a خیلی کوچکتر باشد، الگوریتم را $L_{R\varepsilon P}$ می‌نامیم. حالت ۳ زمانی که b مساوی صفر باشد. الگوریتم را L_{RI} می‌نامند که از دسته اتوماتاهای جذب شونده به یک حالت خاص می‌باشد.

نوع دیگری از اتوماتای یادگیر با ساختار متغیرکه از آن برای تست کارایی الگوریتم خود بهره خواهیم برد، pursuit است. این اتوماتای یادگیر یک اتوماتای غیر خطی می‌باشد که با سرعت بیشتری نسبت به L_{RI} به نقاط جذب همگرا می‌شود [۲۱] و [۲۲]. اتوماتای pursuit برخلاف L_{RI}، بردار احتمال خود را در جهت افزایش احتمال عملی که بیشترین پاداش تخمینی را داشته است، به روز رسانی می‌کند. الگوریتم pursuit به شرح زیر است:

– پاسخ مطلوب

$$p_j(n+1)=(1-a)p_j(n) \quad j\neq i \tag{۱۰}$$

$$p_k(n+1)=1-\sum_{j\neq i}^{r} p_i(n+1) \tag{۱۱}$$

– پاسخ نامطلوب

$$p_j(n+1)=p_j(n) \ \ \forall j \tag{۱۲}$$

که $p_k(n)$ احتمال عملی است که بیشترین مقدار $k_i'(n)$ را داراست. برای هر دو نوع پاسخ، تخمینهایی که از پاداشهای گرفته شده توسط همه عملها زده می‌شود، به صورت زیر به هنگام در می‌آید.

$$W_i(n+1)=W_i(n)+(1-\beta(n)) \tag{۱۳}$$

$$Z_i(n+1)=Z_i(n)+1 \tag{۱۴}$$

$$k_i(n+1)=W_i(n+1)/Z_i(n+1) \tag{۱۵}$$

1. Markovian Model
2. Ergodic Automata

الگوریتم-۱: LAEDA-L_{RI}-P

ورودی: n متغیر، تابع ارزیابی f ، مجموعه مقادیر مجاز برای متغیرها، محدودیتهای مسئله

خروجی: ژنومی که تابع ارزیابی f را بهینه می‌کند.

قدم ۱- در ابتدا مجموعه M متشکل از n اتوماتای یادگیر L_{RI} مقدار دهی اولیه می‌شود. سپس نخستین جمعیت، D_l که $l=0$، با تعداد N ژنوم و با استفاده از مجموعه M تشکیل می‌شود. سپس هر یک از ژنومها با استفاده از f ارزیابی می‌شوند.

قدم ۲- اگر شرایط پایانی الگوریتم بدست آمده باشد، اجرای الگوریتم خاتمه می‌یابد و الا Se ژنوم را از بهترین ژنومهای مجموعه D_l،که دارای ارزش بالاتری هستند انتخاب و این مجموعه D_l^{Se} نامیده می‌شود.

قدم ۳- برای هر متغیر X_i ، $1\le i\le n$، و به ازای همه مقادیر k، $1\le k\le r_i$، $N_i(x_k)$ را به صورت زیر محاسبه می‌کنیم.

$$N_i(x_k)=\sum_{j=1}^{Se}\delta_j(X_i=x_k\,|\,D_l^{Se})$$

به طوریکه اگر متغیر X_i در j امین ژنوم، مقدار k-ام خود را داشته باشد مقدار $\delta_j(X_i=x_k\,|\,D_l^{Se})$ برابر یک و در غیر این صورت برابر صفر است.

قدم ۴- برای هر متغیر X_i ، $1\le i\le n$، مقدار $BA_i^{\max}$ به صورت زیر محاسبه می‌شود.

$$BA_i^{\max}=\arg\max_{1\le k\le r_i} N_i(x_k)$$

قدم ۵- به ازای تمام مقادیر $1\le i\le n$ به عمل $BA_i^{\max}$ اتوماتای یادگیر i-ام مجموعه M، پاداش داده می‌شود.

قدم ۶- مقدار l یکی افزایش یافته و یک جمعیت جدید متشکل از N ژنوم جدید با استفاده از مجموعه M ایجاد می‌شود. سپس کنترل الگوریتم به قدم (۲) منتقل می‌شود.

شکل ۴: LAEDA در حالتی که از اتوماتای یادگیر مبتنی بر پاداش مانند اتوماتای L_{RI} استفاده شود.

که $W_i(n)$ و $Z_i(n)$ به ترتیب تعداد پاداشها و انتخاب شدن عمل i–ام را نشان می‌دهند. می‌توانید اطلاعات بیشتر درباره اتوماتاهای یادگیر را در [۲۳] بیابید.

٤- الگوریتم تخمین توزیع با استفاده از اتوماتای یادگیر

الگوریتم[3] LAEDA از جمله الگوریتم‌های قابل اعمال به مدل اول (متغیرهای مستقل) می‌باشد [۲۴]. در این الگوریتم فرض بر مستقل بودن متغیرهای ژنوم‌ها می‌باشد و به ازای هر متغیر در ژنوم یک اتوماتای یادگیر استفاده می‌شود. تعداد عملهای اتوماتای یادگیر برابر با تعداد مقادیر مجاز برای متغیر متناظر با آن می‌باشد. برای ساختن هر نمونه ژنوم در ابتدا از اتوماتای یادگیر هر متغیر در خواست می‌شود تا عمل مورد نظر خود را انتخاب کند، سپس به متغیر متناظر آن، مقدار متناظر عمل انتخاب شده را انتساب می‌دهیم. بنابراین احتمال ساخته شدن یک ژنوم $x=(x_1,...,x_n)$ از رابطه زیر به دست می‌آید.

$$P(X=x)=\prod_{i=1}^{n}P(X_i=x_i)=\prod_{i=1}^{n}Grad_i^j \tag{۱۶}$$

که $1\le j\le r_i$ و $Grad_i^j$ احتمال انتخاب عمل j متناظر با مقدار x_i، توسط اتوماتای یادگیر iام می‌باشد. در هر مرحله با استفاده از اتوماتاهای یادگیر به تعداد جمعیت، N، ژنوم ساخته می‌شود. سپس جمعیت جدید

3. Learning Automata based Estimation of Distribution Algorithm

//الگوریتم-۲: LAEDA-L_{RP}-P

ورودی: n متغیر، تابع ارزیابی f ، مجموعه مقادیر مجاز برای متغیرها، محدودیتهای مسئله

خروجی: ژنومی که تابع ارزیابی f را بهینه می‌کند.

قدم ۱- در ابتدا مجموعه M متشکل از n اتوماتای یادگیر L_{RP} مقدار دهی اولیه می‌شود. سپس نخستین جمعیت، D_l که $l=0$، با تعداد N ژنوم و با استفاده از مجموعه M تشکیل و سپس هر یک از ژنومها با استفاده *از f ارزیابی* می‌شوند.

قدم ۲- اگر شرایط پایانی الگوریتم بدست آمده باشد، الگوریتم خاتمه می‌یابد و الا Se ژنوم از بین بهترین ژنومهای مجموعه D_l که دارای ارزش بالاتری هستند انتخاب شده و مجموعه D_l^{Se} نامیده می‌شود.

قدم ۳- با احتمال μ به قدم ۷ می‌رویم.

قدم ۴- برای هر متغیر X_i ، $1 \le i \le n$، و به ازای همه مقادیر k ، $1 \le k \le r_i$، $N_i(x_k)$ به صورت زیر محاسبه می‌شود.

$$N_i(x_k) = \sum_{j=1}^{N-Se} \delta_j(X_i = x_k \mid D_l - D_l^{Se})$$

به طوری که اگر متغیر X_i در j امین ژنوم مجموعه $D_l - D_l^{Se}$، مقدار kام خود را داشته باشد مقدار $\delta_j(X_i = x_k | D_l - D_l^{Se})$ برابر یک و در غیر این صورت برابر صفر است.

قدم ۵- برای هر متغیر X_i ، $1 \le i \le n$، مقدار $WA_i^{\max}$ به صورت زیر محاسبه می‌شود.

$$N_i^{\max} = \max_{1 \le k \le r_i} N_i(x_k)$$
$$WA_i^{\max} = \arg \max_{1 \le k \le r_i} N_i(x_k)$$

قدم ۶- به ازای تمام مقادیر $1 \le i \le n$ عمل $WA_i^{\max}$ اتوماتای یادگیر iام مجموعه M، جریمه شده و کنترل الگوریتم به قدم ۱۰ منتقل می‌شود.

قدم ۷- برای هر متغیر X_i ، $1 \le i \le n$ و به ازای همه مقادیر k ، $1 \le k \le r_i$، $N_i(x_k)$ به صورت زیر محاسبه می‌شود.

$$N_i(x_k) = \sum_{j=1}^{Se} \delta_j(X_i = x_k \mid D_l^{Se})$$

به طوریکه اگر متغیر X_i در j امین ژنوم مجموعه D_l^{Se}، مقدار k-ام خود را داشته باشد مقدار $\delta_j(X_i = x_k | D_l^{Se})$ برابر یک و در غیر این صورت برابر صفر است.

قدم ۸- برای هر متغیر X_i ، $1 \le i \le n$، مقدار $BA_i^{\max}$ به صورت زیر محاسبه می‌شود.

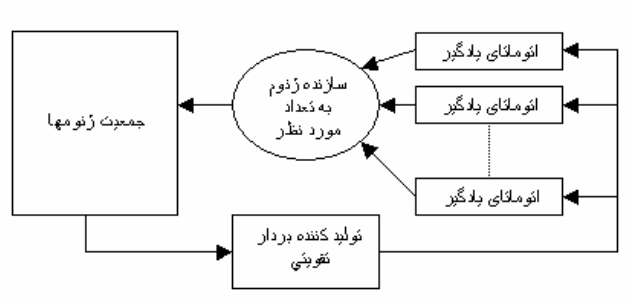
$$N_i^{\max} = \max_{1 \le k \le r_i} N_i(x_k)$$
$$BA_i^{\max} = \arg \max_{1 \le k \le r_i} N_i(x_k)$$

قدم ۹- به ازای تمام مقادیر $1 \le i \le n$ به عمل $BA_i^{\max}$ اتوماتای یادگیر i-ام مجموعه M، پاداش داده می‌شود.

قدم ۱۰- مقدار l یکی افزایش یافته و یک جمعیت جدید متشکل از N ژنوم جدید با استفاده از M ایجاد می‌شود. سپس کنترل الگوریتم به قدم (۲) منتقل می‌شود.

شکل ۵: LAEDA در حالتیکه از اتوماتای یادگیر مبتنی بر پاداش–تنبیه استفاده شود.

توسط تابع ارزیابی مورد ارزیابی قرار می‌گیرد و از میان آنها Se ژنوم، که از بهترین ژنومها هستند انتخاب می‌شوند. بعد از اعمال مکانیزمی که وابسته به مدل محیط اتوماتای یادگیرنده، (مدلهای P ، S و Q) می‌باشد، بردار سیگنال تقویتی ساخته شده و سپس هر اتوماتای یادگیر بردار احتمالات خود را بروز می‌کند. پس از آن، نسل جدید ساخته شده و مراحل فوق تکرار می‌شود. شرط پایان الگوریتم می‌تواند همگرا شدن اتوماتاهای یادگیر، تولید تعداد معینی نسل، انجام تعداد معینی ارزیابی و یا یکنواختی ارزش ژنومها در چندین نسل پایانی باشد. به دلیل ماهیت اتوماتای یادگیر، ژنومهایی که در مراحل اولیه ساخته می‌شوند تصادفی بوده و سمت و سوی خاصی ندارند و به همین دلیل جمعیت اولیه دارای پراکندگی ژنتیکی بالایی می‌باشد. با گذشت زمان و تغییر بردار احتمال



شکل ۶: نمای گرافیکی از الگوریتم LAEDA.

اتوماتاها، این تنوع جمعیتی در جهت بهبود ارزش ژنومها کاهش می‌یابد.

برای گریز از شرایطی که الگوریتم در دام مینیمم‌های محلی گرفتار می‌شود، از یک مکانیزم ثانویه تولید پاداش استفاده می‌شود. به این منظور در هر بار تولید نسل با یک احتمال از پیش تعریف شده به جای انتخاب Se ژنوم برتر جمعیت، $N-Se$ ژنوم از کم ارزشترین ژنومهای جمعیت انتخاب شده و بر اساس آنها بردار سیگنال تقویتی بر اساس مکانیزمی متفاوت ساخته می‌شود.

طبق آنچه گفته شد یک الگوریتم LAEDA را می‌توان به صورت یک هفت تایی $< N, Se, \mu, f, M, \Phi, \Psi >$ نشان داد که N تعداد ژنومها در هر نسل تولید شده، Se تعداد ژنومهای انتخاب شده، μ احتمال جریمه کردن بدترین مقادیر، f تابع ارزیابی، M مجموعه اتوماتاهای یادگیر متناظر با متغیرهای سازنده ژنوم، Φ مکانیزم پاداش دهی به اتوماتاهای یادگیر و بلاخره Ψ تابع نگاشت عملهای اتوماتاهای یادگیر به مقادیر متغیرها است.

از مسایل مهمی که باید به آن توجه شود، انتخاب نوع اتوماتاهای یادگیر است. برای سادگی فرض می‌کنیم همه اتوماتاهای یادگیر مجموعه M از یک نوعند. همچنین در صورت استفاده از اتوماتاهایی مانند L_{RI} که بر پایه پاداش کار می‌کنند و از مکانیزم جریمه کردن در آنها استفاده نمی‌شود، مقدار μ را برابر صفر قرار می‌دهیم. انتخاب نوع اتوماتاهای یادگیر به کار رفته در M و نوع مکانیزم پیاده‌سازی Φ بستگی به مسئله مورد مطالعه دارد.

در شکل ۴ برای روشنتر شدن بیشتر نحوه کار الگوریتم، یک نمونه از الگوریتم LAEDA مبتنی بر مدل P که از الگوریتم یادگیری L_{RI} استفاده می‌کند، ارائه شده است. همانطور که دیده می‌شود با ایجاد تغییر بروی قدمهای ۳، ۴ و ۵ می‌توان از اتوماتاهای یادگیر متفاوت با مدل Q و S و سیستم‌های پاداش دهی متفاوت استفاده کرد.

در الگوریتم ۱، مکانیزم پاداش‌دهی بر پایه پاداش در نظر گرفته شده است. در صورت استفاده از اتوماتاهای یادگیری مانند L_{RP} می‌توانیم، در هر مرحله با در نظر گرفتن یک احتمال، به جای انتخاب بهترین مقادیر متغیرها و ارائه پاداش به اعمال متناظر با آنها، با استفاده از ژنومهای $D_l - D_l^{Se}$ بدترین مقادیر را برای متغیرهای ژنوم یافته و اعمال متناظر با آنها را جریمه کرد. در شکل ۵ الگوریتم پیشنهادی در حالتی که الگوریتم‌های مبتنی بر پاداش و جزا مانند الگوریتم یادگیری L_{RP} استفاده شود، ارائه شده است. یک نمایی کلی از اجزای الگوریتم و ارتباط آنها با یکدیگر در شکل ۶ نمایش داده شده است.

۵- حل چند مسئله با استفاده از LAEDA

به منظور مطالعه کارایی الگوریتم LAEDA، چند مسئله توسط

الگوریتم پیشنهادی حل گردید. فضای همه این مسائل $\{۰,۱\}^n$ می‌باشد. به عبارت دیگر ژنوم‌هایی که برای بازنمایی این مسائل مورد استفاده قرار می‌گیرند، ژنوم‌های دودویی با طول n می‌باشند. OneMax و SubsetSum از مسائل خطی و CheckerBoard و EqualProducts غیر خطی می‌باشند. همچنین به منظور بررسی کارایی الگوریتم در حل مسائل بغرنج دو مسئله Knapsack0/1 و TSP نیز با استفاده از الگوریتم پیشنهادی حل گردید.

مسئله ۱- OneMax : مسئله OneMax یک مسئله ساده خطی می‌باشد. که می‌توان آنرا به صورت ماکزیمم کردن تابع زیر تعریف کرد.

$$F_{OneMax}(x) = \sum_{i=۱}^{n} x_i \qquad (۱۷)$$

این تابع فوق در نقطه $x^* = (۱,...,۱)$ دارای حداکثر مقدار است. این مسئله در بیشتر مقالات الگوریتم‌های تکاملی بعنوان یک مسئله تست مورد استفاده گرفته است.

مسئله ۲- Subset Sum : در این مسئله هدف یافتن یک زیر مجموعه B از مجموعه اعداد A است. به نحوی که مجموع اعضای B برابر یک مقدار مشخص c باشد. به زبان ریاضی می توان گفت:

$$B = \arg\min_{B'' \subseteq A} \left| c - \sum_{x \in B''} x \right| \qquad (۱۸)$$

این مسئله یک مسئله خطی است که می‌تواند به وسیله برنامه‌سازی پویا حل گردد. اما در شرایطی که مجموعه A بزرگ باشد، حل آن بسیار زمانگیر خواهد بود.

مسئله ۳- CheckerBoard: در این مسئله یک شبکه $s \times s$ داده شده است. هر گره از شبکه فوق می‌تواند مقدار صفر یا یک را اختیار کند. هدف ساختن یک الگوی CheckerBoard از مقادیر صفر و یک به روی شبکه می‌باشد به طوریکه هر گره شبکه با مقدار صفر (یک) باید در تمام جهات اصلی توسط گره‌های با مقادیر یک (صفر) احاطه شود. ژنوم‌ها در این مسئله $n = s \times s$ متغیر دارند.

اگر شبکه مورد نظر را با یک ماتریس $C = [c_{ij}]_{i,j=۱,...,s}$ نشان دهیم و $\delta(.,.)$ تابع دلتای Kronecker باشد، آنگاه تابع CheckerBoard را می‌توان به صورت زیر تعریف کرد.

$$F_{Checker\,Board}(C) = ۴(s-۲)^۲ - \sum_{i=۲}^{n-۱}\sum_{j=۲}^{n-۱} \delta(c_{ij}, c_{i-۱\,j}) + \delta(c_{ij}, c_{i+۱\,j}) + \delta(c_{ij}, c_{i\,j-۱}) + \delta(c_{ij}, c_{i\,j+۱}) \qquad (۱۹)$$

مسئله ۴- EqualProducts : این مسئله، یکی از مسائل کلاسیک است. مجموعه‌ای از n عدد حقیقی $\{b_۱, b_۲, ..., b_n\}$ داده شده است. یک زیر مجموعه از آن انتخاب می‌شود. هدف مینیمم کردن تفاوت بین حاصل‌ضرب اعداد انتخاب شده و انتخاب نشده می‌باشد. به زبان ریاضی می‌توان مسئله را به صورت مینیمم کردن تابع زیر تعریف کرد.

$$F_{Equal\,Products}(x) = \left| \prod_{i=۱}^{n} x_i b_i - \prod_{i=۱}^{n} (۱ - x_i) b_i \right| \qquad (۲۰)$$

برای مسائل EqualProducts و SubsetSum مجموعه اعداد مورد نیاز به صورت تصادفی از بازه (۰,۴) انتخاب شده‌اند.

مسئله ۵- کوله پشتی (Knapsack 0/1): یک کوله‌پشتی و تعداد n بسته مفروضند. هر بسته i دارای حجم c_i و ارزش v_i می‌باشد. از طرفی کوله‌پشتی نیز ظرفیتی معین برابر با C دارد، یعنی مجموع حجم بسته‌هایی که می‌تواند در آن قرار بگیرد می‌بایست از مقدار C تجاوز نکند. مسئله از این قرار است که می‌خواهیم طوری بسته‌ها را برای قرار دادن در کوله‌پشتی انتخاب کنیم که در نهایت ارزش بسته‌های درون کوله پشتی حداکثر شود. در پیاده‌سازی این مسئله، هر راه‌حل (یک ترکیب از بسته‌ها)، یک ژنوم خواهد بود. هر ژنوم با فرض اینکه n بسته داشته باشیم، توسط رشته‌ای بیتی، $x = (x_۱,...,x_n)$، به طول n نمایش داده می‌شود. ۱ بودن بیت iام در این رشته نشان دهنده حضور بستهٔ iام در کوله پشتی خواهد بود. نحوه تعیین مقدار تابع ارزش، با مجموع ارزش بسته‌های انتخاب شده برای قرار گرفتن در کوله پشتی، نسبت مستقیم دارد. شرط ظرفیت کوله‌پشتی نیز بدین صورت است که اگر مجموع حجمها از ظرفیت کوله‌پشتی بیشتر شد، ترکیب غیر معتبر خواهد بود. از آنجایی که احتمال ایجاد هر ترکیبی از بیتها- شامل حالت نامعتبر- وجود دارد، ممکن است نتیجه نامعتبر باشد. با راه‌حلهای غیر معتبر می‌توان به دو روش برخورد کرد:

الف) ترکیبهای نامعتبر حذف شوند. در این حالت باید به صورت اتفاقی و یا حریصانه جواب غیر مجاز را تغییر دهیم تا به یک جواب مجاز تبدیل شود. در چنین حالتی تابع ارزش به صورت زیر تعریف می‌شود.

$$f_{Knapsack}(x) = \sum_{i=۱}^{n} (x_i\, v_i) \qquad (۲۱)$$

ب) ترکیبهای نامعتبر از جمعیت حذف نشوند، ولی به آنها امتیاز منفی داده شود و یا به بیان دیگر جریمه شوند

$$f_{Knapsack}(x) = \sum_{i=۱}^{n} (x_i\, v_i) + \eta(C - \sum_{i=۱}^{n} (x_i c_i))\; u(\sum_{i=۱}^{n} (x_i c_i) - C) \qquad (۲۲)$$

که $u(.)$ تابع پله و η ضریب لاگرانژ است. از آنجا که احتمال تولید جوابهای مناسب از جوابهای غیر معتبر، وجود دارد و اینگونه جوابها با اینکه خود غیرقابل قبول هستند ولی می‌توانند راهی برای رسیدن به جوابهای بهینه باشند از روش دوم استفاده می‌کنیم، با این توضیح که پس از محاسبه مقدار شایستگی هر جواب، اگر حاصل نامعتبر بود متناسب با اضافه ظرفیت جریمه انجام می‌شود. دراین مقاله ضریب منظور شده برای جریمه یک در نظر گرفته شده است. نکته مهم دیگری که باید به آن توجه شود وابستگی پیچیدگی مسئله کوله پشتی به توزیع آماری ارزش و حجم بسته ها می‌باشد. مشخصات بسته‌های مورد استفاده در این مقاله، به صورت تصادفی از اعداد صحیح ۱ تا ۳۰ انتخاب شده‌اند.

مسئله ۶- فروشنده دوره گرد (TSP): مسئله فروشنده دوره گرد یکی از مشهورترین مسائل بغرنج[1] می‌باشد. L شهر با مختصات جغرافیایی بر روی یک صفحه مسطح مشخص شده است. هدف پیدا کردن مینیمم توری است که از یک شهر آغاز شده، از سایر شهرها عبور کرده و سپس به شهر مبدأ ختم شود. ژنوم‌هایی که برای بازنمایی تورها بکار می‌روند رشته‌هایی به طول $L\log_۲^L$ بیت می‌باشند که هر شهر با یک زیر رشته به طول $\log_۲^L$ بیت نشان داده می‌شود. این زیررشته در واقع نماینده عدد دودویی شهر می‌باشد. اولین زیر رشته در ژنوم نماینده اولین شهر، دومین زیر رشته نماینده دومین شهر ملاقات شده در تور و به همین صورت بقیه زیر رشته‌ها نماینده سایر شهرها می‌باشند [۱۸].

$$f_{TSP}(x) = d(city_L, city_۱) + \sum_{i=۱}^{L-۱} d(city_i, city_{i+۱})$$
$$city_i = \sum_{j=۰}^{\log_۲^L - ۱} ۲^j\, x_{(i-۱)L+(\log_۲^L - j)} \qquad (۲۳)$$

به طوری که $d(.,.)$ فاصله اقلیدسی بین دو شهر را نشان می‌دهد.

1. NP-Complete

جدول ۱: مشخصات مسائل استفاده شده در شبیه سازی

مساله	F_{OneMax}	$F_{SubsetSum}$	$F_{CheckerBoard}$	$F_{EqualProducts}$	F_{TSP}	$F_{Knapsack}$
تعداد متغیرها	۱۲۸	۱۲۸	۱۰۰	۵۰	۱۲۸	۱۰۰
ماکزیمم تعداد ارزیابی	۱۰۰۰۰۰	۱۰۰۰۰۰	۱۰۰۰۰۰	۳۰۰۰۰۰	۳۰۰۰۰۰	۳۰۰۰۰۰
نوع مساله	ماکزیمم سازی	مینیمم سازی	ماکزیمم سازی	مینیمم سازی	ماکزیمم سازی	ماکزیمم سازی
مقدار بهینه	۱۲۸	نامشخص	۲۵۶	نامشخص	نامشخص	۱۱۴۷*

* با استفاده از برنامه سازی پویا بدست می آید.

جدول ۲: نتایج بدست آمده از الگوریتمهای *LAEDA-L_{RI}-P*، *LAEDA-PURSUIT-P* و *LAEDA-L_{RP}-P* برای مسائل مطرح شده در جدول (۱)

مساله		LAEDA-L_{RI}-P	LAEDA-Pursuit-P	LAEDA-L_{RP}-P
F_{OneMax}	ارزش بهترین جواب بدست آمده	**۱۲۸**	**۱۲۸**	**۱۲۸**
	تعداد ارزیابیها	۳۲۳۹	۲۶۷۰	۵۲۰۰
$F_{SubsetSum}$	ارزش بهترین جواب بدست آمده	۰/۰۰۳۸۴	۰/۰۴۸۳۵	**۰/۰۰۳۲۱**
	تعداد ارزیابیها	۴۰۵۰	۳۸۰	۷۵۳۰
$F_{CheckerBoard}$	ارزش بهترین جواب بدست آمده	۱۸۶	۱۶۶	**۲۱۰**
	تعداد ارزیابیها	۱۰۰۰۰	۵۰۰	۴۴۰۰۰
$F_{EqualProducts}$	ارزش بهترین جواب بدست آمده	۱/۱۶	۲/۹۷	**۱/۱۰۳**
	تعداد ارزیابیها	۱۷۰۰۰	۴۰۰۰	۲۲۳۴۰
F_{TSP}	ارزش بهترین جواب بدست آمده	۱۰۹۸	۹۱۰	**۱۱۴۱**
	تعداد ارزیابیها	۲۷۳۰۰	۸۳۱۰	۳۲۸۵۰
$F_{Knapsack}$	ارزش بهترین جواب بدست آمده	۱۸۹۳	۲۳۲۴	**۱۷۱۰**
	تعداد ارزیابیها	۵۲۸۰۰	۷۹۳۰	۵۷۲۰۰

جدول ۳: نتایج بدست آمده از الگوریتم‌های UMDA، SGA و PBIL برای مسائل مطرح شده در جدول (۱).

مساله		UMDA	PBIL	SGA
F_{OneMax}	ارزش بهترین جواب بدست آمده	**۱۲۸**	**۱۲۸**	۱۲۵
	تعداد ارزیابیها	۷۳۵۰	۴۲۴۰	۳۶۲۱۰
$F_{SubsetSum}$	ارزش بهترین جواب بدست آمده	/۰۰۰۳۳۲	**۰/۰۰۳۲۰**	۰/۰۰۳۴۴
	تعداد ارزیابیها	۶۴۳۰	۴۰۵۰	۱۲۴۳۰
$F_{CheckerBoard}$	ارزش بهترین جواب بدست آمده	۲۴۱	۲۱۰	**۲۴۶**
	تعداد ارزیابیها	۵۳۰۰۰	۱۳۲۴۰	۵۲۰۰۰
$F_{EqualProducts}$	ارزش بهترین جواب بدست آمده	۱/۹۵	**۱/۱**	۳/۳۵
	تعداد ارزیابیها	۳۶۳۲۰	۱۵۳۲۰	۱۹۳۴۳۰
F_{TSP}	ارزش بهترین جواب بدست آمده	**۱۱۴۱**	۱۱۲۵	۱۰۲۴
	تعداد ارزیابیها	۳۴۱۲۰	۲۷۳۰۰	۳۰۰۰۰۰
$F_{Knapsack}$	ارزش بهترین جواب بدست آمده	**۱۷۹۴**	۱۹۲۶	۱۸۷۳
	تعداد ارزیابیها	۸۹۷۴۰	۵۲۸۰۰	۳۰۰۰۰۰

مختصات شهرهای استفاده شده در این مقاله به صورت تصادفی تولید شده است.

٦- نتایج شبیه‌سازی

در این قسمت نتایج حاصل برای الگوریتم پیشنهادی با نتایج الگوریتم‌های ژنتیک استاندارد (SGA)، UMDA وPBIL مقایسه خواهیم کرد. در تمام موارد دو پارامتر کیفیت جواب و تعداد ارزیابی‌ها مورد توجه قرار می‌گیرند. تمام نتایج میانگین ۱۰ بار اجرای الگوریتم‌ها است. در همه شبیه‌سازی از هر سه اتوماتای یادگیر L_{RP}، L_{RI} و Pursuit استفاده کردیم. برای مقایسه کارایی الگوریتم‌های ذکر شده پارامترهای مشترک بین آنها مانند تعداد ژنوم‌های هر نسل و تعداد ژنوم‌های انتخاب شده در هر نسل (در LAEDA، PBIL و UMDA) را یکسان در نظر گرفتیم. تعداد ژنوم‌های هر نسل برای مسائل ۱، ۲، ۳، ۴، ۵ و ۶ به ترتیب برابر ۱۰، ۱۰ ،۲۰، ۱۰ ، ۱۰۰، ۱۰۰ در نظر گرفته شد. نرخ یادگیری در LAEDA و PBIL و μ (برای الگوریتم LAEDA-L_{RP}) به ترتیب در همه جا برابر ۰/۰۱، ۰/۰۱ و ۰/۰۵ در نظر گرفته شده است. برای توقف الگوریتم‌ها از چند شرط استفاده شده است. شرط اول که تنها در مورد مسائل OneMax، CheckerBoard و Knapsack مورد استفاده قرار گرفته است، رسیدن به مقدار بهینه می‌باشد. شرط دوم رسیدن تعداد نسل‌های تولید شده به عدد خاصی می‌باشد. که این اعداد بر اساس پیچیدگی مسئله متفاوت در نظر گرفته شد. شرط پایانی دیگر همگن شدن جمعیت نسلها به لحاظ ارزش آنها می‌باشد. در مورد LAEDA وPBIL پس از تولید چند نسل بردارهای احتمال تقریبا همگرا می‌شوند و دیگر نیازی به تولید نسلهای دیگر نمی‌باشد. در الگوریتم‌های UMDA، PBIL و LAEDA ، Se ژنوم با ارزش بالاتر برای به روز رسانی مدل احتمالاتی ژنوم انتخاب شدند. مقدار Se در همه آزمایشها برابر نصف جمعیت هر نسل قرار دادیم. و نکته آخر این که در تمام الگوریتم‌ها به منظور از دست ندادن بهترین ژنوم‌های تولید شده فرض کردیم در فرایند جستجو همواره بهترین ژنوم نسل فعلی به نسل بعد منتقل می‌شود. در جدول ۱ مشخصات مسائل بیان شده است. در جدول ۲ نتایج حاصل از شبیه‌سازی برای مسائل مطرح شده در جدول ۱ آورده شده‌اند. در مورد هر الگوریتم دو مقدار در جدول ذکر شده است. اولین مقدار، ارزش بهترین جواب بدست آمده قبل از توقف الگوریتم می‌باشد. مقدار دوم، تعداد ارزیابی‌ها می‌باشد که نشان‌دهنده تعداد ژنوم‌های تولید شده توسط هر الگوریتم است.

همانطور که دیده می‌شود، از میان الگوریتم‌های LAEDA-L_{RP}-P، LAEDA-L_{RI}-P و LAEDA-Pursuit-P الگوریتم LAEDA-L_{RP}-P در اکثر موارد جواب بهتری برخوردار است. که دلیل این امر استفاده از مکانیزم ثانویه تنبیه در آن می‌باشد. الگوریتم LAEDA-Pursuit-P در مسائلی که وابستگی بین متغیرهای آن زیاد می‌باشد، افت کارایی بسیاری دارد. دلیل این موضوع را می‌توان این گونه توجیح کرد که در مسائل با متغیرهای وابسته، محیط فعالیت اتوماتاهای یادگیر دارای دینامیک بالایی می‌باشد و به همین دلیل اتوماتاهای Pursuit که دارای تعصب‌گرایی زیادی نسبت یادگرفته‌های خود می‌باشند، نمی‌توانند تغییرات محیط را دنبال کنند. اما در مورد مسئله OneMax که متغیرهای مسئله از یکدیگر مستقل می‌باشند، به دلیل عدم وابستگی رفتار اتوماتاهای یادگیر به یکدیگر، LAEDA-Pursuit-P با سرعتی بالاتر از سایر الگوریتم‌ها به جواب بهینه می‌رسد.

در جدول ۳ نتایج الگویتم‌های SGA، UMDA و PBIL برای مسائل مورد مطالعه آورده شده‌اند. با توجه به نتایج به دست آمده، برای مسئله OneMax بهترین نتیجه متعلق به الگوریتم LAEDA-Pursuit-P و برای مسائل TSP و Knapsack بهترین نتایج متعلق به الگوریتم LAEDA-L_{RP}-P و بدترین نتایج متعلق به الگوریتم LAEDA-Pursuit-P می‌باشد. الگوریتم PBIL نتایج بهتری را برای مسئله EqualProducts تولید می‌کند. در مورد مسئله CheckerBoard الگوریتم SGA، بسیار بهتر از سایر الگوریتم‌ها عمل می‌کند. الگوریتم پیشنهادی نتایج جالبی برای مسئله CheckerBoard تولید نمی‌کند. در مورد مسئله SubsetSum نتایج حاصل از الگوریتم‌های PBIL و LAEDA-L_{RP}-P تقریباً برابر می‌باشد، اما الگوریتم PBIL با سرعت بیشتری به جواب دست پیدا می‌کند.

۷- نتیجه‌گیری

در این مقاله الگوریتم تکاملی جدیدی از دسته الگوریتم‌های تخمین توزیع بر اساس اتوماتای یادگیر معرفی شد. در این الگوریتم با فرض استقلال متغیرهای مسئله بهینه‌سازی، از اتوماتای یادگیر برای تخمین توزیع احتمالاتی متغیرها استفاده شد. اگر چه فرض مستقل بودن متغیرها در اکثر مسائل به خصوص مسائل بهینه‌سازی ترکیبی نقض می‌شود ولی آزمایشهای انجام شده بروی تعدادی از مسائل مطرح شده نشان‌دهنده کارایی خوب الگوریتم می‌باشد. از مزایای این الگوریتم امکان استفاده از تعداد متنوعی از اتوماتاهای یادگیر می‌باشد. از طرفی بر خلاف الگوریتم‌های مشابه، به راحتی در فضاهای جستجوی غیر دودویی می‌توان از آن استفاده کرد. مطالعات نشان می‌دهد، الگوریتم LAEDA-LRP-P می‌تواند به عنوان یک ابزار مناسب برای بهینه‌سازی در کنار الگوریتم‌های رایج دیگر مورد استفاده قرار گیرد.

مراجع

[1] S. Baluja and R. Caruana, "Removing the genetics from the standard genetic algorithm", in *Proc. of ICML'95*, pp. 38-46, Morgan Kaufmann Publishers, Palo Alto, CA, 1995.

[2] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. binary parameters", *Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature-PPSN IV*, pp. 178-187, 1996.

[3] S. Baluja and S. Davies, "Using optimal dependency trees for combinatorial optimization: learning the structure of search space", *Technical Report CMU-CS-97-107, Carnegie Mellon University*, Pittsburgh, Pennsylvania, 1997.

[4] S. Baluja, "Population based incremental learning: a method for integrating genetic search based function optimization and competitive learning", *Technical Report CMU-CS-94-163, Carnegie Mellon University*, Pittsburgh, Pennsylvania, 1994.

[5] J. S. De Bonet, C. L. Isbell, and P. Viola, "MIMIC: finding optima by estimating probability densities", *In Proc. of NIPS'97*, pp. 424–431, MIT Press, Cambridge, MA, 1997.

[6] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm", *IEEE Trans. on Evolutionary Computing*, vol. 3, no. 4, pp. 287–297, Nov. 1999.

[7] H. Mühlenbein and T. Mahnig, "The factorized distribution algorithm for additively decomposed functions", in *Proc. of the 1999 Congress on Evolutionary Computation*, IEEE Press, pp. 752-759, 1999.

[8] H. Mühlenbein, "The equation for response to selection and its use for prediction", *Evolutionary Computation*, vol. 5, no. 3, pp. 303-346, Fall 1998.

[9] H. Mühlenbein and M. Pelikan, *The Bivariate Marginal Distribution Algorithm*, Advances in Soft Computing-Engineering Design and Manufacturing, pp. 521-535, 1999.

[10] M. Pelikan, D. E. Goldberg, and F. Lobo, "A survey of optimization by building and using probabilistic model", *Illinois Genetic*

Algorithm Report, No. 99018, Illinois University, Illinois, USA, Sep. 1999.
[11] M. Pelikan, D. E. Goldberg, and E. Cant-Paz, "Linkage problem, distribution estimation and Bayesian networks", *Evolutionary Computation*, vol. 8, no. 3, pp. 311-340, Sep. 2000.
[12] H. Mühlenbein and T. Mahnig, *Evolutionary Algorithms: From Recombination to Search Distributions, Theoretical Aspects of Evolutionary Computing*, Springer Publication, 2001.
[13] G. Harik, "Linkage learning via probabilistic modeling in the ECGA", *Illinois Genetic Algorithm Report, no. 99010, Illinois University*, Illinois, USA, 1999.
[14] M. N. Howell, T. J. Gordon, and F. V. Brandao, "Genetic learning automata for function optimization", *IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 32, no. 6, pp. 804-815, Dec. 2002.
[15] M. Munetomi, Y. Takai, and Y. Sato, "stGA: An application of genetic algorithm to stochastic learning automata", *Syst., Computer. Jpn.*, vol 27, pp. 67-78, Sep. 1996.
[16] P. Larranaga, R. Etxeberria, J. A. Lozano, and J. M. Pena, "Optimization by learning and simulation of Bayesian and Gaussian networks", *Technical Report EHU-KZAA-IK-4/99, Department of Computer Science and Artificial Intelligence*, University of Basque Country, Dec. 1999.
[17] D. Heckerman, "A tutorial on learning with Bayesian networks", *Technical Report, MSR-TR-95-06, Advanced Technology Division, Microsoft Cooperation*, Redmond, WA, Nov. 1996.
[18] G. Syswerda, "Simulated crossover in genetic algorithms", in *Proc. Second Workshop on Foundation of Genetic Algorithms, Morgan Kaufmann Publishers*, San Mateo California, pp. 239-255, 1993.
[19] N. Monmarch´e, E. Ramat, G. Dromel, M. Slimane, and G. Venturini, "On the similarities between AS, BSC and PBIL: toward the birth of a new Meta-heuristic", *Technical Report 215, Laboratoire d'Informatique, Universit´e de Tours*, 1999.
[20] G. Rosenberg, and A. Salomaa (Editors), *Handbook of Formal Language*, Springer-Verlag, Berlin, 1993.
[21] B. J. Oommen and M. Agache, "Continuous and discredited pursuit learning schemes: various algorithms and their comparison", *IEEE Trans. on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 31, no. 3, pp. 277-287, Jun. 2001.
[22] M. A. L. Thathachar and P. S. Sastry, "Estimator algorithms for learning automata", *Proc. Platinum Jubilee Conferences on Systems and Signal Processing, Electrical Eng. Department, Indian Institute of Science*, Bangalore, India, Dec. 1986.
[23] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*, Printice-Hall Inc, 1989.
[24] R. Rastegar and M. R. Meybodi, "LAEDA: a new evolutionary algorithm using learning automata", *Accepted in 10th Annual International Computer Society of Iran Computer Conference CSICC2004, Sharif university*, Tehran, Iran, 2004.

رضا رستگار تحصیلات خود را در مقاطع کارشناسی و کارشناسی ارشد مهندسی کامپیوتر بترتیب در سالهای ۱۳۸۱ و ۱۳۸۳ در دانشگاه صنعتی امیرکبیر به پایان رسانده است. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: یادگیری ماشین، شبکه های عصبی مصنوعی و الگوریتمهای جستجوی تصادفی.

محمدرضا میبدی تحصیلات خود را در مقاطع کارشناسی و کارشناسی ارشد اقتصاد بترتیب در سالهای ۱۳۵۲ و ۱۳۵۶ از دانشگاه شهید بهشتی و در مقاطع کارشناسی ارشد و دکتری علوم کامپیوتر بترتیب در سالهای ۱۳۵۹ و ۱۳۶۲ از دانشگاه اوکلاهمای آمریکا به پایان رسانده است و هم اکنون استاد دانشکده مهندسی کامپیوتر دانشگاه صنعتی امیرکبیر می‌باشد. نامبرده قبل از پیوستنش به دانشگاه صنعتی امیرکبیر در سالهای ۱۳۶۲ الی ۱۳۶۴ استادیار دانشگاه میشیگان غربی و در سالهای ۱۳۶۴ الی ۱۳۷۰ دانشیار دانشگاه اوهایو در ایالات متحده آمریکا بوده است. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: الگوریتمهای موازی، پردازش موازی، محاسبات نرم و کاربردهای آن، شبکه های کامپیوتری و مهندسی نرم افزار.