PSO-LA: یک مدل جدید برای بهینه سازی

محمد شیبانی* محمد رضا میبدی†

چکیده

حرکت جمعی ذرات یک تکنیک بهینه سازی است که بر اساس قوانین احتمال کار میکند و از حرکت گروهی پرندگان و ماهیها هنگامی که دنبال غذا میگردند الهام گرفته شده است. در این روش هر یک از ذرات سعی میکنند به سمتی حرکت کنند که بهترین تجربه های شخصی و گروهی در آن نقاط روی داده است. مشکل اصلی این مدل که در بسیاری از مسایل به خصوص مسایل چند قله ای بروز میکند مساله گیر افتادن در بهینه های محلی است. در این مقاله یک مدل جدید بر اساس PSO به نام PSO-LA پیشنهاد میشود که در آن از یک اتوماتای یادگیر برای تنظیم رفتار ذرات و برقراری موازنه بین جستجوی سراسری و جستجوی محلی استفاده میشود. نتایج آزمایشها بر روی مسایل نمونه نشان میدهد که روش ارائه شده از عملکرد بهتری در مقایسه با مدل PSO استاندارد برخوردار است.

کلمات کلیدی

حرکت دسته جمعی ذرات، اتوماتاهای یادگیر، بهینه سازی

PSO-LA: A New Model for Optimization

Mohammad Sheybani Mohammad Reza Meybodi

Abstract

Particle swarm optimization (PSO) is a population based statistical optimization technique which inspired by social behavior of bird flocking or fish schooling. PSO starts with a random population of particles. Each particle of the population tries to move toward the points of search space in which the best experience of the particle itself and also the best experience of the population occurred. The main weakness of PSO especially in multi modal problems is trapping in local optimums. In this paper a new PSO model called PSO-LA is proposed in which a learning automaton takes the role of configuring the behavior of particles and also creating a balance between the process of global and local search. The results of experiments conducted on some standard problems show that the proposed algorithm produces better results than the standard PSO.

Keywords

Particle Swarm, Learning Automata, Optimization

* دانشجوي كارشناسي ارشد دانشكده مهندسي كامپيوتر و فناوري اطلاعات دانشگاه صنعتي اميركبير، sheybani@ce.aut.ac.ir

† عضو هيئت علمي دانشكده مهندسي كامپيوتر و فناوري اطلاعات دانشگاه صنعتي اميركبير، meybodi@ce.aut.ac.ir

۱- مقدمه

حرکت دسته جمعی ذرات[1] یک تکنیک کارا برای حل مسایل بهینه سازی است که بر مبنای قوانین احتمال و بر اساس جمعیت کار میکند. در این روش هر یک از اعضای جمعیت که ذره نام دارند سعی میکنند با تنظیم مسیر خود و حرکت به سمت بهترین تجربه شخصی و بهترین تجربه جمعی گروه، به سوی راه حل نهایی حرکت کنند. از آنجا که در این الگوریتم ذرات به تدریج به سمت بهترین راه حل پیدا شده تا به حال میل میکنند اگر این راه حل، یک بهینه محلی باشد ذرات همگی به سمت آن میروند و الگوریتم استاندارد PSO راهکاری برای خروج از این بهینه محلی ارائه نمیدهد. این بزرگترین مشکل PSO استاندارد است که سبب میشود در حل مسایل چند قله ای مخصوصا با فضای حالت بزرگ ناتوان باشد.

یکی از اعمال صورت گرفته برای مقابله با مشکل بهینه های محلی در الگوریتم حرکت دسته جمعی ذرات، استفاده از جهش[2] است [۶][۷][۸]. در [۶] و [۷] از جهش گاوسی در PSO استاندارد و نیز نسخه هایی تغییر یافته از آن استفاده شده است. در این مقالات مدل PSO با جهش ارائه شده با استفاده از توابعی با حد اکثر دو بعد آزمایش شده اند و جوابهای مناسبی داشته اند. در [۸] برای جهش از توزیع کوشی استفاده شده است. به این ترتیب که هر ذره با احتمال P_{mutate} میتواند جهش پیدا کنند. هرگاه ذره ای برای جهش انتخاب شد، هر یک از اجزای بردار آن ذره به احتمال $1/d$ که d تعداد ابعاد مساله است، ممکن است دچار جهش شوند. برای جهش دادن هر جزء از یک ذره یک عدد به صورت تصادفی از توزیع کوشی انتخاب میشود و با مقدار جزء مربوطه از بردار ذره جمع میشود. روش ارائه شده برای برخی مسایل با فضای حالتهای بزرگ، جواب مناسبی داشته است.

در الگوریتم حرکت دسته جمعی ذرات استاندارد، در محاسبه سرعت ذره در گام بعد، کل سرعت فعلی ذره محاسبه میشود. در واقع سرعت ذره در هر گام از دو قسمت تشکیل میشود که قسمت اول سرعت فعلی ذره و قسمت دوم مربوط به دنبال کردن بهترین تجربه شخصی و بهترین تجربه گروه است. بدون قسمت دوم الگوریتم حالت یک جستجوی سراسری کورکورانه را خواهد داشت و بدون قسمت اول، الگوریتم به جستجویی محلی در نزدیکی بهترین ذره تبدیل خواهد شد که در رسیدن به قسمتهای زیادی از فضای جستجو ناتوان خواهد بود. الگوریتم حرکت دسته جمعی ذرات با ترکیب این دو قسمت، سعی میکند که به نوعی تعادل را بین جستجوی سراسری و محلی ایجاد نماید.

در [۵] پارامتری با نام وزن میانی تعریف شده است. وزن میانی در واقع ضریبی از سرعت فعلی ذره است که در تعیین سرعت ذره در گام بعد مورد استفاده قرار میگیرد. هدف از معرفی این پارامتر این بوده است که برقراری تعادل بین جستجوی محلی و جستجوی سراسری، بسته به نوع مساله بتواند قابل تنظیم باشد. در برخی از کاربردها از یک مقدار وزن میانی ثابت استفاده شده است ولی معمولا وزن میانی در ابتدا مقدار بالاتری دارد و با گذر زمان مقدار آن به صورت خطی کاهش می یابد و به این ترتیب ذرات در ابتدا بیشتر سعی در شناسایی محلهای جدید از فضای جستجو دارند و با گذشت زمان بیشتر میل به دنباله روی پیدا میکنند. کاهش مقدار وزن میانی به صورت خطی روشی بسیار ساده است که توانایی بکارگیری کامل دانش اولیه راجع به مساله و نیز استفاده از بازخورد سیستم را ندارد.

برخی از محققان سعی کرده اند تا با افزودن نویز به حرکات ذرات در الگوریتم استاندارد PSO، باعث حرکات متنوعتری شوند و بدین صورت مشکل بهینه های محلی را حل کنند[۹][۱۰]. برخی دیگر از راه حلها برای حل مشکل بهینه های محلی از طریق ترکیب الگوریتم PSO با الگوریتمهای دیگر مانند الگوریتمهای ژنتیک[3] و روش بالا رفتن در امتداد شیب[4] و الگوریتم سرد کردن تدریجی[5] حاصل شده است[۱۱][۱۸].

یک اتوماتای یادگیر[۳]، ماشینی است که میتواند تعدادی متناهی عمل را انجام دهد. هر عمل انتخاب شده توسط یک محیط احتمالی ارزیابی میشود و نتیجه ارزیابی در قالب سیگنالی مثبت یا منفی به اتوماتای یادگیر داده میشود و اتوماتا از این پاسخ در انتخاب عمل بعدی استفاده میکند و بدین ترتیب به سمت انتخاب عملی که بیشترین پاداش را از محیط میگیرد، میل میکند و به یا عبارتی دیگر عملی را که بیشترین پاداش را از محیط دریافت میکند یاد میگیرد. اتوماتای یادگیر برای بهبود قدرت یادگیری بسیاری از الگوریتمها مورد استفاده قرار گرفته است که از آن جمله میتوان به شبکه های عصبی [۱۴][۱۵] و الگوریتمهای ژنتیک [۱۶][۱۷] و همچنین حرکت دسته جمعی ذرات باینری[۱۲] اشاره نمود. در [۱۲] مدل PSO باینری بر پایه اتوماتای یادگیر ارائه شده است. در این روش از یک اتوماتون یادگیر در هر بعد ذره استفاده شده است. هر کدام از این اتوماتاهای یادگیر دارای دو عمل ۰ و ۱ میباشد و به عنوان مغز متفکر ذره عمل میکند و حرکت آن را در فضای حالت محدود به ۰ و ۱ کنترل و راهبری مینماید. این مدل نتایج بهتری در مقایسه با مدل باینری استاندارد[۲] در حل مسایل نمونه داشته است.

در این مقاله مدل جدیدی به نام PSO-LA پیشنهاد میشود. در مدل پیشنهادی، از یک اتوماتای یادگیر برای تنظیم رفتار ذرات استفاده شده است. اتوماتای یادگیر در هر گام تعیین میکند که ذرات به مسیر فعلی ادامه دهند و یا به دنباله روی از بهترین ذرات پیدا شده

تاکنون بپردازند. در الگوریتم پیشنهادی وظیفه برقراری تعادل بین جستجوی سراسری و جستجوی محلی که در [۵] بر عهده ضریب میانی گذاشته شده است توسط اتوماتای یادگیر انجام میشود. استفاده از اتوماتای یادگیردارای دو مزیت عمده میباشد: اولا میتوان از دانش موجود در تعیین روند تغییرات وزن میانی استفاده نمود و ثانیا این روند با گرفتن بازخورد از اجرای الگوریتم اصلاح گردد.

ادامه این گزارش به صورت زیر سازماندهی شده است. بخش ۲ به معرفی مدل PSO میپردازد. در بخش سوم اتوماتاهای یادگیر به اختصار معرفی شده است. بخش ۴ مدل پیشنهادی شرح داده میشود. بخش پنجم نتایج شبیه سازیها را ارائه میکند و بخش پایانی نتیجه گیری میباشد

۲- حرکت دسته جمعی ذرات

حرکت جمعی ذرات(PSO)، یک تکنیک بهینه سازی احتمالی است که بر مبنای جمعیت کار میکند. این روش در سال ۱۹۹۵ توسط دکتر ابرهارت و دکتر کندی ارائه شد [۱] و ایده اصلی آن از رفتار دسته جمعی ماهیها یا پرندگان به هنگام جستجوی غذا الهام گرفته شده است. گروهی از پرندگان در فضایی به صورت تصادفی دنبال غذا میگردند. تنها یک تکه غذا در فضای مورد بحث وجود دارد. هیچ یک از پرندگان محل غذا را نمیدانند. یکی از بهترین استراتژیها میتواند دنبال کردن پرنده ای باشد که کمترین فاصله را تا غذا داشته باشد. این استراتژی در واقع جانمایه الگوریتم PSO است.

در الگوریتم PSO، هر راه حل که به آن یک ذره گفته میشود، معادل یک پرنده در الگوری حرکت جمعی پرندگان میباشد. هر ذره یک مقدار شایستگی دارد که توسط یک تابع شایستگی محاسبه میشود. هر چه ذره در فضای جستجو به هدف- غذا در مدل حرکت پرندگان- نزدکتر باشد، شایستگی بیشتری دارد.. همچنین هر ذره دارای یک سرعت است که هدایت حرکت ذره را بر عهده دارد. هر ذره با دنبال کردن ذرات بهینه در حالت فعلی، به حرکت خود در فضای مساله ادامه میدهد.

آغاز کار PSO به این شکل است که گروهی از ذرات(راه حلها) به صورت تصادفی به وجود می آیند و با به روز کردن نسلها سعی در یافتن راه حل بهینه مینمایند. در هر گام، هر ذره با استفاده از دو بهترین مقدار به روز میشود. اولین مورد، بهترین موقعیتی است که تا کنون ذره موفق به رسیدن به آن شده است. موقعیت مذکور با نام pbest شناخته و نگهداری میشود. بهترین مقدار دیگری که توسط الگوریتم مورد استفاده قرار میگیرد، بهترین موقعیتی است که تا کنون توسط جمعیت ذرات بدست آمده است. این موقعیت با gbest نمایش داده میشود. در برخی ویرایشهای PSO، ذره، قسمتهایی از جمعیت را که همسایگان توپولوژیکی اش هستند، انتخاب و تنها آنها را در اعمال خود دخیل میکند که در این حالت بهترین راه حل محلی که با lbest نشان داده میشود، به جای gbest مورد استفاده قرار میگیرد. پس از یافتن بهترین مقادیر، سرعت و مکان هر ذره با استفاده از معادلات (۱) و (۲) به روز میشود.

v[] = v[] + c1 * rand() * (pbest[] - position []) + c2 * rand() * (gbest[] - position[])	(۱)
position[] = position[] + v[]	(۲)

در معادلات (۱) و (۲)، v[] سرعت ذره و position [] محل فعلی ذره هستند که هر دو آرایه هایی به طول تعداد ابعاد مساله میباشند. Rand() یک عدد تصادفی در بازه (0,1) است. C1 و C2 نیز پارامترهای یادگیری هستند و معمولا C1=C2=2 در نظر گرفته میشود. سرعت ذرات در هر بعد به یک مقدار Vmax محدود میشوند. اگر مجموع شتابها باعث شوند که سرعت در یک بعد از Vmax بیشتر شود، مقدار سرعت در آن بعد برابر با Vmax قرار میگیرد. شبه کد الگوریتم PSO در شکل (۱) مشاهده میشود.

سمت راست معادله (۱) از سه قسمت تشکیل شده است که قسمت اول، سرعت فعلی ذره است و قسمتهای دوم و سوم تغییر سرعت ذره و چرخش آن به سمت بهترین تجربه شخصی و بهترین تجربه گروه را به عهده دارند. اگر قسمت اول را در این معادله در نظر نگیریم، آنگاه سرعت ذرات تنها با توجه به موقعیت فعلی و بهترین تجربه ذره و بهترین تجربه جمع تعیین میشود. به این ترتیب، بهترین ذره جمع، در مکان خود ثابت باقی میماند و سایرین به سمت آن ذره حرکت میکنند. در واقع حرکت دسته جمعی ذرات بدون قسمت اول معادله(۱)، پروسه ای خواهد بود که طی آن فضای جستجو به تدریج کوچک میشود و جستجویی محلی حول بهترین ذره شکل میگیرد. در مقابل اگر فقط قسمت اول معادله (۱) را در نظر بگیریم، ذرات راه عادی خود را میروند تا به دیواره محدوده برسند و به نوعی جستجویی سراسری را انجام میدهند[۵].

```
For each particle
   Initialize particle
End For
Do
   For each particle
      Calculate fitness value of the particle fp
      /*updating  particle's best fitness value so far)*/
       If fp is better than pBest
          set current value as the new pBest
   End For
    /*updating  population's best fitness value so far)*/
   Set gBest to the best fitness value of all particles
```

```
For each particle
   Calculate particle velocity according equation (1)
   Update particle position according equation (2)
End For
While maximum iterations OR
      minimum error criteria is not attained
```

شکل ۱- شبه کد الگوریتم PSO

در معادله (۱) با ترکیب این دو عامل سعی شده است تا موازنه ای بین جستجوی محلی و سراسری برقرار گردد. در [۵]، یک پارامتر جدید توسط ارائه دهندگان الگوریتم اولیه PSO بود، ارائه گردید. این پارامتر جدید وزن میانی نامیده و با w نشان داده شد. هدف از معرفی این پارامتر برقراری موازنه بهتر بین جستجوی محلی و جستجوی سراسری است. به این منظور در معادله (۱)، این ضریب در سرعت اولیه ذره ضرب میشود و به عبارتی تنها قسمتی از سرعت اولیه به سرعت آینده ذره منتقل میشود و به این ترتیب معادله (۱) به شکل معادله (۳) تغییر میکنند.

```
V[] = w * v[] +
      c1 * rand() * (pbest[] – position[]) +
      c2 * rand() * (gbest[] – position[])
```
(۳)

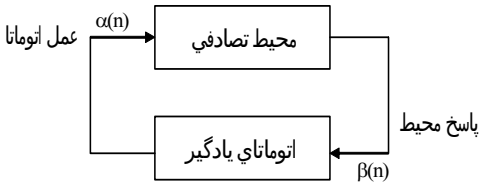
وزن میانی میتواند یک ضریب ثابت، یک تابع خطی با زمان و یا حتی یک تابع غیرخطی با زمان نیز باشد. در بسیاری از کاربردها وزن میانی به صورت تابعی خطی با زمان در نظر گرفته میشود که تابعی نزولی است. به این ترتیب در ابتدا قسمت بیشتری از سرعت فعلی ذره در سرعت آینده اش دخیل میشود و با گذر زمان، این میزان کاهش می یابد. به عبارت بهتر، در ابتدا ذرات میل بیشتری به حرکات انفجاری و تجربه های تازه دارند و با گذشت زمان این میل جای خود را به دنباله روی بیشتر از بهترینها میدهد. این روش در بسیاری موارد میتواند کاهش دهنده مشکل گیر افتادن در مینیممهای محلی باشد.

مدل باینری PSO برای حل مسایل گسسته، در سال ۱۹۹۷ توسط ارائه دهندگان الگوریتم استاندارد، معرفی شد[۲]. در این مدل موقعیت هر ذره در هر بعد با یکی از دو مقدار ۰ یا ۱ مشخص میگردد. به این ترتیب، ذره در یک فضای محدود به ۰ و ۱ حرکت میکند و سرعت ذره در هر بعد برابر با احتمال یک بودن موقعیت ذره در آن بعد خواهد بود. بروز رسانی سرعت همچنان مطابق با معادله (۱) انجام میشود. سپس ابتدا سرعت بدست آمده در هر بعد با استفاده از تابع سیگموید به بازه [۰,۱] منتقل میگردد و آنگاه موقعیت جدید ذره در هر بعد مطابق با معادله (۴) محاسبه میگردد. rand() یک مقدار تصادفی در بازه [۰,۱] است.

```
If (rand() < sigmoid(v_id(t+1)) then
    x_id(t+1) = 1
else
    xid(t+1) = 0
```
(۴)

۳- اتوماتاهای یادگیر[۶]

اتوماتای یادگیر[۳]، ماشینی است که میتواند تعدادی متناهی عمل را انجام دهد. هر عمل انتخاب شده توسط یک محیط احتمالی ارزیابی میشود و نتیجه ارزیابی در قالب سیگنالی مثبت یا منفی به اتوماتا داده میشود و اتوماتا از این پاسخ در انتخاب عمل بعدی تاثیر میگیرد. هدف نهایی این است که اتوماتا یاد بگیرد تا از بین اعمال خود بهترین عمل را انتخاب کند. بهترین عمل، عملی است که احتمال دریافت پاداش از محیط را به حداکثر برساند. کارکرد اتوماتای یادگیر در تعامل با محیط، در شکل(۲) مشاهده میشود.



شکل (۲) : ارتباط بین اتوماتای یادگیر و محیط

محیط را می‌توان 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 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 از نوع Q، $\beta(n)$ می‌تواند به طور گسسته یک مقدار از مقادیر محدود در فاصله [۰,۱] و در محیط از نوع S، $\beta(n)$ متغیر تصادفی در فاصله [۰,۱] است. 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 با ساختار متغیر را میتوان توسط چهارتایی $\{\alpha, \beta, p, T\}$ نشان داد که $\alpha = \{\alpha_1, ..., \alpha_r\}$ 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 می باشد. الگوریتم زیریک نمونه از الگوریتمهای یادگیری خطی است. فرض کنید عمل α_i در مرحله nام انتخاب شود.

- پاسخ مطلوب

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)]$$
$$p_j(n+1) = (1-a)p_j(n) \qquad \forall j \quad j \neq i \qquad (۵)$$

- پاسخ نامطلوب

$$p_i(n+1)=(1-b)p_i(n)$$
$$p_j(n+1)=(b/r-1)+(1-b)p_j(n) \quad \forall j \;\; j\neq i \qquad (۶)$$

در روابط (۵) و (۶)، a پارامتر پاداش و b پارامتر جریمه می باشند. با توجه به مقادیر a و b سه حالت زیر را می توان در نظر گرفت. زمانیکه a و b با هم برابر باشند، الگوریتم را L_{RP} می نامیم، زمانیکه b از a خیلی کوچکتر باشد، الگوریتم را $L_{R\varepsilon P}$ می نامیم. و زمانیکه b مساوی صفر باشد الگوریتم را L_{RI} مینامیم[۴]. شکل (۳) شبه کد اتوماتای یادگیر با ساختار متغیر را نشان میدهد.

```
Initialize p to [1/s,1/s,…,1/s]  /*s is the number of
actions*/
While not done
   Select an action i based on the probability vector p
   Evaluate action and return a reinforcement signal β
   Update probability vector using learning rule
End While
```

شکل (۳) : اتوماتای یادگیر با سختار متغیر

۴- مدل پیشنهادی (PSO-LA)

در این بخش یک الگوریتم حرکت دسته جمعی ذرات بر پایه اتوماتای یادگیر که انرا PSO-LA مینامیم پیشنهاد میگردد. مانند الگوریتم حرکت دسته جمعی ذرات، در این الگوریتم نیز یک جمعیت از ذرات وجود دارد و هر یک از ذرات دارای یک موقعیت و سرعت اولیه هستند. تفاوت الگوریتم پیشنهادی با الگوریتم استاندارد این است که الگوریتم پیشنهادی از یک اتوماتای یادگیر برای کنترل رفتار ذرات استفاده میکند. این اتوماتای یادگیر دارای دو عمل «دنباله روی» و «ادامه مسیر فعلی» میباشد. در ابتدا، موقعیت و سرعت ذرات و همچنین بردار احتمالات انتخاب اعمال اتوماتای یادگیر مقداردهی اولیه میشوند. سپس تا زمانیکه حداکثر تعداد گامها انجام گردد و یا هدف مورد نظر حاصل شود، مراحل زیر تکرار میشوند:

۱- اتوماتای یادگیر یکی از اعمالش را بر طبق بردار احتمال اعمالش، انتخاب میکند.

۲- با توجه به عمل انتخاب شده، نحوه بروزرسانی سرعت ذرات تعیین میشود و سپس ذرات، سرعت و موقعیت خود را بروز میکنند.

۳- بر اساس نتایج به روزرسانی موقعیت ذرات، عمل اتوماتای یادگیر، ارزیابی میشود و بردار احتمال انتخاب اعمال اتوماتای یادگیر اصلاح میشود.

عملی که اتوماتای یادگیر در هر گام برمیگزیند، تعیین کننده شیوه بروز کردن سرعت ذرات در آن گام میباشد. در صورت انتخاب عمل «دنباله روی»، تنها دنبال کردن بهترین تجربه شخصی و بهترین تجربه گروهی، در به روز نمودن سرعت ذرات مد نظر قرار خواهند گرفت و از سرعت فعلی ذرات صرفنظر میشود که در این صورت بروزرسانی سرعت ذرات مطابق با فرمول (۷) انجام میگردد. در صورت انتخاب عمل «ادامه مسیر فعلی»، سرعت جدید ذرات برابر با سرعت فعلی آنها خواهد بود و ذره همچنان مسیری فعلی را ادامه خواهد داد.

```
V[] = c1 * rand() * (pbest[] – position[]) +
      c2 * rand() * (gbest[] – position[])
```
(۷)

در واقع انتخاب عمل «دنباله روی»، انجام یک جستجوی محلی را در پی خواهد داشت و انتخاب عمل «ادامه مسیر فعلی»، باعث جستجوی سراسری و کشف قسمتهایی ناشناخته از فضای جستجو میگردد. وظیفه اتوماتای یادگیر، یادگیری بردار احتمال مطلوب و برقراری تعادل بین جستجوی سراسری و جستجوی محلی در فرایند جستجو میباشد.

شیوه ارزیابی عمل انتخاب شده، به این صورت است که موقعیت فعلی هر ذره با موقعیت قبلی ان مقایسه میشود. چنانچه c_{imp} درصد ازجمعیت موقعیتشان بهبود یافته باشد، عمل انتخاب شده مثبت و در غیر اینصورت منفی ارزیابی میشود. c_{imp} از پارامترهای الگوریتم پیشنهادی است که بایستی با توجه به نوع مساله و اتوماتای یادگیر بکار گرفته شده، تنظیم گردد. شکل (۴) شبه کد الگوریتم پیشنهادی را نشان میدهد.

```
Initialize the LA
For each particle
   Initialize particle
End For
Do
   The LA selects an action ac
   For each particle
      Calculate fitness value of the particle fp
      /*updating  particle's best fitness value so far)*/
       If fp is better than pBest
          set current value as the new pBest
   End For
    /*updating  population's best fitness value so far)*/
   Set gBest to the best fitness value of all particles
   For each particle
      if  ac is "follow the best"
           Calculate particle velocity according equation (7)
      Update particle position according equation (2)
   End For
While maximum iterations OR
         minimum error criteria is not attained
```

شکل (۴)- شبه کد الگوریتم PSO-LA

۵- نتایج شبیه سازیها

آزمایشات بر روی چهار تابع استاندارد صورت گرفته است که معمولا به عنوان معیار سنجش الگوریتمهای بهینه سازی مورد استفاده قرار میگیرند. توابع استفاده شده عبارتند از توابع آکلی، روزنبراک،

رستریجین و اسفیر که به ترتیب توسط معادلات (۸) تا (۱۱) تعریف شده اند. این توابع همگی دارای بهینه سراسری با مقدار ۰ هستند. آزمایشات با مقادیر n برابر با ۱۰، ۲۰ و ۳۰ صورت گرفته است و پارامتر c_{imp} برابر با ۷۵ قرار داده شده است.

$f(x) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$	(۸)
$f(x) = \sum_{i=1}^{n-1} (100\ (x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	(۹)
$f(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	(۱۰)
$f(x) = \sum_{i=1}^{n} x_i^2$	(۱۱)

تعداد جمعیت ۲۵ و تعداد گامها به ترتیب ۲۵ و ۱۰۰۰ در نظر گرفته شده اند. ضرایب پاداش و جریمه برای اتوماتای یادگیر L_{RP} و نیز ضریب پاداش در اتوماتای یادگیر L_{RI} برابر با ۰٫۰۱ در نظر گرفته شده است. آزمایشات ۳۰ بار تکرار شده اند و بهترین و میانگین نتایج سی بار اجرا ارائه شده است. نتایج بدست آمده در جدولهای۱ تا ۴ مشاهده میشوند.

جدول ۱- نتایج مقایسه دو الگوریتم PSO و PSO-LA برای تابع آکلی

ابعاد	الگوریتم	بهترین	متوسط
۱۰	PSO-Std	۰٫۵۹	۲٫۵
	PSO-I.W	۰٫۰۸	۲٫۴۹
	PSOLA-L_{RP}	۰٫۰۱	۱٫۷۸
	PSOLA-L_{RI}	۰٫۰۱	۱٫۵۹
۲۰	PSO-Std	۴٫۳۲	۶٫۰۹
	PSO-I.W	۳٫۷۶	۵٫۳۶
	PSOLA-L_{RP}	۲٫۵۸	۴٫۸۷
	PSOLA-L_{RI}	۲٫۵۸	۴٫۷۹
۳۰	PSO-Std	۷٫۴۶	۹٫۱۱
	PSO-I.W	۶٫۵۱	۸٫۳۴
	PSOLA-L_{RP}	۶٫۲۷	۷٫۷۷
	PSOLA-L_{RI}	۶٫۱۵	۷٫۶۴

جدول ۲- نتایج مقایسه دو الگوریتم PSO و PSO-LA برای تابع روزنبراک

ابعاد	الگوریتم	بهترین	متوسط
۱۰	PSO-Std	۱۴٫۳۹	۳۵۲٫۸۶
	PSO-I.W	۰٫۴۳	۱۱٫۳۳
	PSOLA-L_{RP}	۰٫۳۶	۱۰٫۹۹
	PSOLA-L_{RI}	۰٫۷۳	۷٫۱۱
۲۰	PSO-Std	۵۸۶۴٫۰۶	۳۱۶۷۴٫۲۶
	PSO-I.W	۱۴٫۷۲	۳۵٫۲۴
	PSOLA-L_{RP}	۱۱٫۴۸	۲۹٫۳۵
	PSOLA-L_{RI}	۱۴٫۶۲	۳۱٫۹۸
۳۰	PSO-Std	۵۲۰۵۷٫۶۴	۴۱۲۵۱۵٫۰۴
	PSO-I.W	۲۹٫۱۵	۸۹٫۰۰
	PSOLA-L_{RP}	۲۳٫۶۹	۶۷٫۰۵
	PSOLA-L_{RI}	۲۲٫۹۸	۶۷٫۸۴

جدول ۳- نتایج مقایسه دو الگوریتم PSO و PSO-LA برای تابع رستریجین

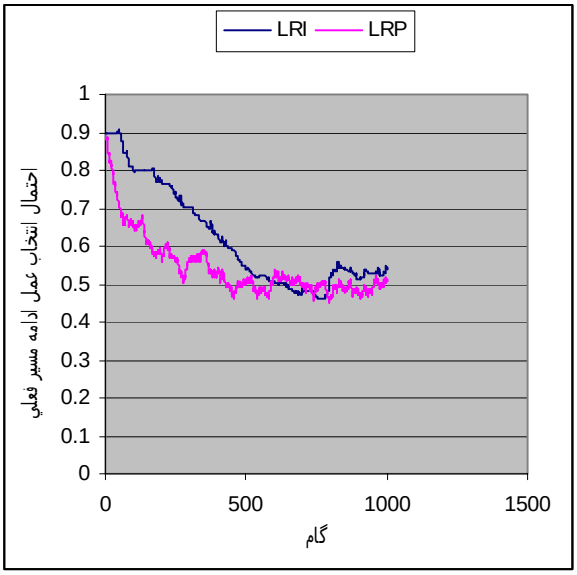
ابعاد	الگوریتم	بهترین	متوسط
۱۰	PSO-Std	۴٫۹۷	۱۰٫۱۴
	PSO-I.W	۳٫۶۵	۸٫۶۹
	PSOLA-L_{RP}	۳٫۰۸	۹٫۳۹
	PSOLA-L_{RI}	۴٫۰۱	۹٫۵۳
۲۰	PSO-Std	۱۰٫۰۵	۲۸٫۲۱
	PSO-I.W	۱۰٫۹۹	۲۸٫۹۳
	PSOLA-L_{RP}	۹٫۲۳	۲۱٫۸۴
	PSOLA-L_{RI}	۱۱٫۶	۲۲٫۷۲
۳۰	PSO-Std	۲۸٫۴۹	۵۷٫۱۰
	PSO-I.W	۲۴٫۲۵	۳۴٫۱۶
	PSOLA-L_{RP}	۱۶٫۹۲	۳۱٫۸۶
	PSOLA-L_{RI}	۱۳٫۷۲	۳۲٫۲۱

جدول ۴- نتایج مقایسه دو الگوریتم PSO و PSO-LA برای تابع اسفیر

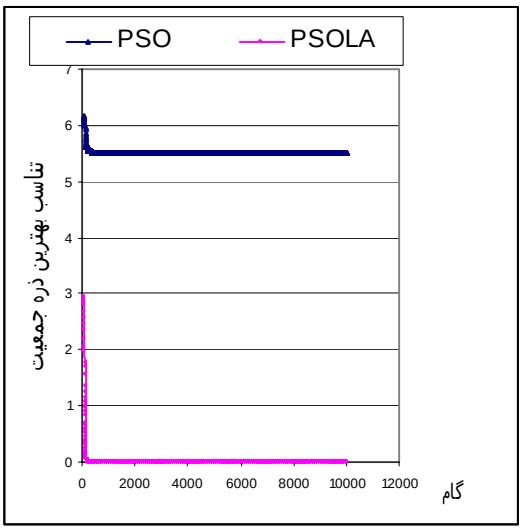
ابعاد	الگوریتم	بهترین	متوسط
۱۰	PSO-Std	۰٫۰۰۹	۰٫۳۳
	PSO-I.W	۰٫۰۰۰۴	۰٫۰۰۲
	PSOLA-L_{RP}	۰٫۰۰۰۱	۰٫۰۰۱
	PSOLA-L_{RI}	۹٫۱۳e-۵	۰٫۰۰۰۱
۲۰	PSO-Std	۰٫۶۹	۹٫۲۱
	PSO-I.W	۰٫۰۰۸	۷٫۰۷
	PSOLA-L_{RP}	۰٫۰۰۶	۰٫۰۱
	PSOLA-L_{RI}	۰٫۰۰۱	۰٫۰۰۲
۳۰	PSO-Std	۲٫۱۱	۱۵٫۰۸
	PSO-I.W	۰٫۰۲	۵٫۸۲
	PSOLA-L_{RP}	۰٫۰۰۸	۰٫۰۱
	PSOLA-L_{RI}	۰٫۰۰۵	۰٫۰۰۹

نتایج جداول (۱) تا (۴) نشان میدهد که الگوریتم PSOLA عملکرد بهتری در مقایسه با الگوریتم حرکت دسته جمعی ذرات حتی زمانیکه از وزن میانی استفاده میکند داشته است. این برتری مخصوصا در مورد توابع روزنبراک و اسفیر کاملا محسوس است. شکل (۵)، نمودار تغییرات احتمال انتخاب عمل «ادامه مسیر فعلی» را برای بهترین اجراهای PSOLA-L_{RP} و PSOLA-L_{RI} بر روی تابع اسفیر با

بعد ۳۰ نشان میدهد. با مشاهده این شکل و اطلاعات جدول (۴) ملاحظه میشود، نوع اتوماتای یادگیر تاثیر بسزایی در تغییرات احتمال اعمال و در نتیجه در حاصل کار دارد. این که کدام اتوماتای یادگیر جواب بهتری تولید میکند به مساله و شکل فضای جستجو بستگی دارد. شکل (۶) نموداری مقایسه ای از تغییرات بهترین مقدار تناسب ذرات جمعیت هنگام بکار گیری دو الگوریتم PSO استاندارد و PSO-LA-L_{RI} بر روی تابع اسفیر با بعد ۳۰ را نشان میدهد..



شکل (۵) : نمودار تغییرات احتمال انتخاب عمل "ادامه مسیر فعلی" در بهترین اجراهای PSOLA-L_{RP} و PSOLA-L_{RI} با تابع اسفیر بعد ۳۰



شکل (۶) : نمودار تغییرات بهترین مقدار تناسب ذرات جمعیت برای الگوریتمهای PSO استاندارد و PSOLA-L_{RI} با تابع اسفیر بعد ۳۰

۶- نتیجه گیری

در این مقاله یک مدل جدید برای بهینه سازی که بر پایه دو مدل حرکت دسته جمعی ذرات و اتوماتاهای یادگیر میباشد پیشنهاد گردید. در مدل پیشنهادی جمعیت به یک اتوماتای یادگیر مجهز میباشد که به مثابه مغز متفکر و کنترل کننده حرکات ذرات عمل میکند. اتوماتای یادگیر دارای دو عمل دنبال کردن بهترینها و یا ادامه مسیر فعلی میباشد. با انتخاب یک عمل توسط اتوماتای یادگیر در هر گام نحوه بروز رسانی سرعت ذرات در ان گام تعیین میشود. نتایج شبیه سازیها نشان داد که مدل ارائه شده جوابهایی بهتری در مقایسه با مدل PSO استاندارد و همچنین مدل PSO که از پارامتر وزن میانی استفاده میکند تولید مینماید.

مراجع

[1] Kennedy, J. and Eberhart, R.C., "*Particle Swarm Optimization*", Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, pp. 1942-1948, 1995.

[2] Kennedy, J. and Eberhart, R. C., "*A Discrete Binary Version of the Particle Swarm Algorithm*", Proceedings of the International Conference on Systems, Man, and Cybernetics, IEEE Service Center, Piscataway, NJ, pp. 4104-4108, 1997.

[3] Narendra K. S. and Thathachar M. A. L., Learning Automata: An Introduction, Prentice Hall, 1989.

[4] Thathachar, M.A.L. and Sastry, P.S., "*Varieties of Learning Automata: An Overview*", IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics, Vol. 32, No. 6, pp. 711-722, 2002.

[5] Shi, Y. and Eberhart, R. C., "*A Modified Particle Swarm Optimizer*", IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA, 1998.

[6] Hagashi, N. and Iba, H., "*Particle Swarm Optimization with Gaussian Mutation*" Proceedings of the IEE swarm intelligence symposium, Indianapolis, Indiana, USA, pp. 72-79, 2003.

[7] Secrest, B. R. and Lamont, G.B., "*Visualizing Particle Swarm Optimization – Gausssian Particle Swarm Optimization*", Proceedings of the IEEE Swam intelligence Symposium, Indianapolis, Indiana, USA, pp. 198-204, 2003.

[8] Stacey, A., Jancic, M., and Grundy, I., "*Particle Swarm Optimization with Mutation*", Proceedings of the Congress on Evolutionary Computation, p.1425-1430, 2003.

[9] Riget, J. and Vesterstroem, J.S., "*A Diversity Guided particle Swarm Optimizer - the ARPSO*", Depatment of Computer Science, University of Aarhus, Tech. Rep.No. 2002-02, 2002.

[10] Kiink, T., Vesterstroem, J. S. and Riget, J., "*Particle Swam Optimization with Spatial Particle Extension*", Proceedings of the lEEE Congress on Evolutionary Computation, pp. 1474-1479, 2002.

[11] Krink, T. and Lovhjerg, M., "*The Life Cycle Model: Combining Particle Swarm Optimization, Genetic Algorithms and Hill Climbers*" Proceedings of Parallel Problem Solving from Nature VI1 (PPSN 2002), pp. 621-630, 2002.

[12] Rastegar, R., Meybodi, M. R. and Badie, K. "*A New Discrete Binary Particle Swarm Optimization based on Learning Automata*", Proceedings of the International Conference on Machine Learning and Applications, ICMLA '04, pp. 456-462, 2004.

[13] Shi, Y. and Eherhart, R. C., "*Empirical Study of Particle Swarm Optimization*", Proceedings of the IEEE Congress 011 Evolutionary Computation, pp. 1945-1950, 1999.

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

[15] Beigy, H. and Meybodi, M. R., "*A Learning Automata Based Algorithm for Determination of Minimum Number of Hidden Units for Three Layers Neural Networks*", Journal of Amirkabir, Vol. 12, No. 46, pp. 111-136, 2001.

[16] Munetomi, M., Takai, Y., and Sato, Y., "*StGA: An Application of Genetic Algorithm to Stochastic Learning Automata*", Syst. Comput. Jpn., Vol. 27, PP. 68-78, 1996.

[17] Howell, M. N., Gordon, T. J., and Brandao, F. V., "*Genetic Learning Automata for Function Optimization*", IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics, Vol. 32, No. 6, 2002.

[18] Wang, X. H. and Li, J. J., "*Hybrid Particle Swarm Optimization with Simulated Annealing*", Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, China, 2004.

---

1 Particle Swarm Optimization (PSO)
2 Mutation
3 Genetic Algorithms
4 Hill Climbing
5 Simulated Annealing
6 Learning Automata