یک رویکرد خودتطبیقی نوین بر اساس بهینه‌سازی دسته ذرات برای بهینه‌سازی در محیط‌های پویا

بابک نصیری[۱]، دانیال یزدانی[۲]، محمدرضا میبدی[۳]

[۱] دانشکده برق، رایانه و فناوری اطلاعات، دانشگاه آزاد اسلامی، قزوین، ایران
nasiri.babak@qiau.ac.ir

[۲] دانشگاه آزاد اسلامی، واحد شیروان، ایران
d.yazdani@IEEE.org

[۳]دانشکده مهندسی کامپیوتر، فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران، ایران
mmeybodi@aut.ac.ir

چکیده

در بسیاری از مسائل بهینه‌سازی در دنیای واقعی، تابع هدف یا محدودیت‌ها می‌توانند در طول زمان تغییر یابند که در نتیجه بهینه این مسائل نیز می‌تواند تغییر یابد. اگر هر یک از این رویدادهای نامعین در فرآیند بهینه‌سازی مورد توجه قرار گیرند، این مسأله دینامیک یا پویا نامیده می‌شود. بسیاری از مسائل در دنیای واقعی به صورت پویا، غیرقطعی و پیچیده می‌باشند و حل آنها بصورت ایستا چندان به حل مسئله در دنیای واقعی کمک نمی‌کند. با توجه به الزامات محیط های پویا، الگوریتم هایی که برای بهینه سازی در این محیط ها طراحی شده اند دارای اصولی هستند که آنها را از الگوریتم های طراحی شده برای محیط های ایستا متمایز می کند. در این مقاله یک الگوریتم نوین برای بهینه‌سازی در محیط پویا مبتنی بر الگوریتم بهینه‌سازی دسته ذرات پیشنهاد شده است. نتایج حاصل از رهیافت پیشنهادی بر روی معیار قله‌های متحرک که در حال حاضر شناخته‌شده‌ترین معیار برای ارزیابی در محیط-های پویا می‌باشد ارزیابی شده و با نتایج حاصل از چندین الگوریتم معتبر مورد مقایسه قرار گرفته است. نتایج بدست آمده نشان-دهنده کارایی بالای الگوریتم پیشنهادی در مقایسه با سایر الگوریتم‌ها می‌باشد.

کلمات کلیدی

بهینه‌سازی دسته ذرات، محیط‌های پویا، چنددستگی، ذرات کوانتوم، معیار قله‌های متحرک.

۱- مقدمه

عدم قطعیت در بسیاری از مسائل در دنیای واقعی کاملا واضح و مشهود است. یکی از روشهای برخورد با عدم قطعیت استفاده از روشهای تکاملی و هوش جمعی می‌باشد. مسائل غیرقطعی که تاکنون توسط روش‌های تکاملی مورد بررسی قرار گرفته‌اند را بطور کلی می-توان به چهار دسته تقسیم نمود. وجود نویز در تابع ارزیاب، آشفتگی در متغیرهای طراحی، تقریبی بودن تابع ارزیاب و پویا بودن راه‌حل بهینه. در این مقاله عدم قطعیت از نوع پویا بودن راه‌حل بهینه که جزو عمومی‌ترین انواع عدم قطعیت می‌باشد مورد توجه قرار گرفته است.

تاکنون روشهای مختلفی برای حل مسائل پویا با استفاده از روش‌های پردازش تکاملی [۱] و هوش جمعی[۲][۳] ارائه شده است. دو مشکل عمده روش‌های پردازش تکاملی که باعث عدم توانایی استفاده مستقیم از این روش‌ها برای بهینه‌سازی در محیط‌های پویا شده است، حافظه غیرمعتبر و از دست رفتن تنوع می‌باشد. هنگامی که محیط تغییر می-کند، راه‌حل‌های بدست آمده موجود در حافظه دیگر معتبر نمی‌باشد و یا باید آنها بطور کامل فراموش کرد یا دوباره آنها را ارزیابی نمود.

همچنین از آنجا که اکثر روش‌های پردازش تکاملی و هوش جمعی بدلیل ماهیتشان به یک نقطه همگرا می‌شوند لذا تنوع دسته در محیط از بین می‌رود و در صورت تغییر در محیط همگرا شدن به نقطه بهینه جدید در صورت امکان، بسیار زمانگیر خواهد بود. روشهای مختلفی برای ایجاد یا نگهداری تنوع دسته در محیط وجود دارد که در ادامه به آنها پرداخته می‌شود.

ساده‌ترین راه برای واکنش در برابر تغییر محیط، در نظر گرفتن هر تغییر به عنوان ورود یک مسأله بهینه‌سازی جدید می‌باشد که باید از ابتدا حل شود. در صورت داشتن زمان کافی، این یک گزینه مناسب می‌باشد اما در بیشتر مواقع زمان موجود برای بهینه‌سازی مجدد نسبتاً کوتاه می‌باشد. یک تلاش طبیعی برای تسریع فرآیند بهینه‌سازی پس از یک تغییر، استفاده از اطلاعات مربوط به فضای جستجوی قبلی برای پیشروی در جستجو پس از تغییر می‌باشد. به عنوان مثال اگر فرض شود که بهینه جدید نزدیک به بهینه قبلی قرار دارد، می‌توان جستجو را محدود به نقطه مجاور بهینه قبلی کرد. این که آیا استفاده مجدد از اطلاعات گذشته، مطلوب است یا نه، وابسته به ماهیت تغییر می‌باشد.

اگر تغییر به صورت گسترده باشد و فضای مسئله پس از تغییر دارای شباهت کمی با قبل باشد، شروع مجدد می‌تواند تنها گزینه مناسب باشد و هرگونه استفاده مجدد از اطلاعات جمع‌آوری شده بر مبنای مسأله قبلی گمراه‌کننده خواهد بود. در اکثر مسائل دنیای واقعی، امید می‌رود که تغییرات نسبتاً هموار باشند یعنی بین محیط مسئله پیش از تغییر و پس از تغییر ارتباطی وجود داشته باشد تا بتوان از اطلاعات قبلی برای تسریع فرآیند بهینه سازی استفاده کرد.

سؤال اساسی که در این جا مطرح می‌شود این است که چه اطلاعاتی باید نگهداری شود و این اطلاعات چگونه باید برای تسریع عمل جستجو پس از تغییر محیط استفاده شوند. مسئله دیگر این است که در بیشتر الگوریتم های بهینه سازی، پس از اینکه الگوریتم به یک نقطه همگرا شد، تنوع خود را از دست می دهد که این امر باعث کاهش قابلیت انطباق و سازگاری با تغییر محیط می شود، بنابراین در کنار انتقال اطلاعات بین عامل های الگوریتم های بهینه سازی از قبل به بعد از تغییر محیط، باید به دنبال راهکارهایی برای افزایش تنوع و سازگاری الگوریتم پس از تغییر محیط باشیم.

تاکنون برای بهینه‌سازی در محیط‌های پویا از روش‌های تکاملی و هوش جمعی مختلفی از جمله الگوریتم بهینه‌سازی دسته ذرات(PSO) و غیره استفاده شده است. الگوریتم بهینه سازی دسته ذرات در سال ۱۹۹۵ توسط kennedy و Eberhart معرفی شد[۴]. از آن سال تاکنون نسخه‌های مختلفی از این الگوریتم پیشنهاد شده و بهبودهای مناسبی نیز بر روی آن داده شده است. از این الگوریتم برای بهینه-سازی در محیط‌های پویا بسیار استفاده شده است [۵][۶][۷].

در این مقاله یک الگوریتم بهینه‌سازی مبتنی بر الگوریتم PSO پیشنهاد می‌گردد که برای عمل در محیط‌های پویا طراحی شده است و تمام الزامات محیط‌های پویا را برآورده می‌کند. الگوریتم پیشنهادی از راهکارهای خودتطبیقی نوینی برای رسیدن به نتایج بهتر بهره‌مند شده است. الگوریتم پیشنهادی بر روی سناریوهای مختلف بنچمارک قله-های متحرک (MPB[1])[۸] که از معروفترین بنچمارک‌های محیط‌های پویا است به کار رفته است و کارایی آن با شش الگوریتم دیگر به نام-های mQSO [۵]، mCPSO [۵]، CellularPSO [۹]، SPSO [۱۰]، RPSO[۱۱] و rSPSO[۱۲] مقایسه شده است. معیار مقایسه، خطای برون‌خطی[2] است که یکی از معیارهای اصلی مقایسه الگوریتم‌های طراحی شده برای محیط‌های پویا می‌باشد[۳]. نتایج آزمایشات نشان می‌دهد که الگوریتم پیشنهادی از کارایی قابل قبولی برخوردار است.

ادامه این مقاله بدین ترتیب سازماندهی شده است. در بخش دوم به تشریح الگوریتم پیشنهادی می‌پردازیم. در بخش سوم نتایج آزمایشات مورد بررسی قرار می‌گیرند و بخش نهایی به بیان نتیجه‌گیری می-پردازد.

۲- الگوریتم پیشنهادی

در الگوریتم پیشنهادی تعداد از پیش تعیین شده‌ای دسته وجود دارد که هر دسته وظیفه یافتن و تعقیب کردن یک قله را بر عهده دارد زیرا در محیط‌های پویا بهینه‌های محلی ممکن است پس از تغییر محیط به بهینه سراسری تبدیل شوند بنابراین لازم است تا جای امکان تمام آنها تحت پوشش دسته‌ها قرار گیرند. ارتباط بین دسته‌ها به دو صورت محلی بر اساس r_{excl} و سراسری بر اساس r_{conv} می‌باشد[۵]. در واقع هنگامی که تمام دسته‌ها همگرا شده باشند و تعداد قله‌ها بیش از تعداد دسته‌ها باشد، دسته‌ای که کمترین مقدار شایستگی را دارد در فضای مسئله مقداردهی اولیه می‌شود. همچنین اگر دو دسته فاصله کمتر از r_{excl} از هم داشته باشند، دسته‌ای که شایستگی کمتری دارد مقداردهی اولیه می‌شود. زیرا برای هر قله، تحت پوشش بودن توسط یک دسته کفایت می‌کند بنابراین هنگامی که دو دسته به یک قله همگرا شدند باید تنها یکی از آنها باقی بماند و دیگری در فضای مسئله مقداردهی اولیه شود تا شانس این را داشته باشد که به قله‌ای همگرا شود که هنوز تحت پوشش هیچ دسته‌ای قرار نگرفته باشد. همچنین هنگامی که همه دسته‌ها همگرا شده باشند، هیچ دسته‌ای نمی‌تواند از قله‌ای که تحت پوشش آن است خارج شود زیرا تنوع، گستردگی دسته و سرعت ذرات کمتر از آن است که دسته‌ها بتوانند از بهینه‌هایی که در آنها قرار گرفته‌اند به بیرون حرکت کنند. در این حالت مشکل از آنجا ناشی می‌شود که وقتی تعداد قله‌ها بیش از دسته‌ها باشد و همه دسته-ها همگرا شده باشند، برخی از قله‌ها هیچ وقت تحت پوشش قرار نمی-گیرند بنابراین در [۵] پیشنهاد شد در چنین شرایطی دسته‌ای که

بدترین مقدار شایستگی را دارد باید در محیط مسئله مقدار دهی اولیه شود تا شانس پیدا کردن قله‌های دیگر که احتمالاً بلندتر هستند نیز به وجود آید. در الگوریتم پیشنهادی ما از دو تکنیک بالا استفاده می‌کنیم یعنی الگوریتم پیشنهادی از چنددستگی[۳]، انحصار[۴] و ضدهمگرایی[۵] استفاده می‌کند.

در الگوریتم پیشنهادی همانند [۵] هر دسته از دو نوع ذره یعنی ذرات معمولی و ذرات کوانتوم تشکیل شده است اما نحوه کارکرد ذرات کوانتوم دارای تفاوت‌های اساسی با [۵] است. ذرات معمولی در هر تکرار از اجرای الگوریتم بر اساس سرعت قبلی، بهترین تجربه شخصی و بهترین تجربه گروهی موقعیت خود را بهنگام می‌کنند[۱۳]. در الگوریتم پیشنهادی ابتدا ذرات معمولی موجود در تمام دسته‌ها موقعیت خود را بهنگام می‌کنند سپس موقعیت Gbest هر دسته با استفاده از تنها یک ذره کوانتوم در چند مرحله بهبود می‌یابد. در الگوریتم پیشنهادی پارامتر جدیدی به نام Try-number اضافه شده است که یک ذره کوانتوم تا Try-number بار می‌تواند موقعیت Gbest را بهبود دهد.

پس از اجرای الگوریتم PSO و بهنگام شدن موقعیت Gbest، با استفاده از رابطه (۱) یک موقعیت را در شعاع r_{cloud} در اطراف Gbest در نظر می‌گیریم که در واقع همان موقعیت جدید ذره کوانتوم است.

$$X_{i,j} = Gbest_{i,j} + (R_j \times r_{Cloud}) \qquad (۱)$$

که در رابطه ۱، $X_{i,j}$ مؤلفه jام از موقعیت ذره کوانتوم دسته iام است و R_j یک عدد تصادفی با توزیع یکنواخت در بازه [۱، ۱-] می باشد. بنابراین موقعیت ذره کوانتوم در هر یک از ابعاد فضای جستجو می‌تواند درون شعاع r_{cloud} از اطراف موقعیت Gbest باشد. پس از تعیین موقعیت ذره کوانتوم، مقدار شایستگی آن سنجیده می‌شود و با مقدار شایستگی Gbest مقایسه می‌شود. در صورتی که مقدار شایستگی موقعیت ذره کوانتوم بهتر از Gbest باشد، موقعیت Gbest به موقعیت ذره کوانتوم تغییر می‌کند و در غیر این صورت موقعیت آن بدون تغییر می‌ماند. رابطه ۱ برابر با Try-number بار اجرا می‌شود و مطابق با آن موقعیت Gbest می‌تواند تا Try-number بار بهبود یابد.

استفاده از ایده ذره کوانتوم به صورت ذکر شده موجب افزایش سرعت همگرایی بر اساس تعداد ارزیابی شایستگی می‌شود. همچنین می‌تواند مشکل کاهش تنوع در دسته‌ها پس از تغییر محیط را به خوبی حل کند. همان‌طور که در رابطه ۱ مشاهده می‌شود مقدار شعاع r_{cloud} در تعیین موقعیت ذره کوانتوم نقش اساسی دارد. در واقع کارایی ذره کوانتوم تا حد زیادی وابسته به اندازه این شعاع است. در صورتی که مقدار r_{cloud} کوچک باشد توانایی جستجوی محلی الگوریتم بسیار بالا می‌رود اما باعث کاهش طول گام حرکت در هر تکرار و در نتیجه کاهش سرعت همگرایی الگوریتم می‌شود و از توانایی بازیابی تنوع دسته نیز کاسته می‌شود. در صورتی که مقدار این شعاع زیاد باشد، احتمال یافتن موقعیت‌های بهتر در آن کاهش می‌یابد و توانایی جستجوی محلی نیز کاهش می‌یابد.

در الگوریتم پیشنهادی مقدار شعاع r_{cloud} بر اساس طول گام حرکت قله‌ها تعیین می‌شود. در واقع مقدار این پارامتر باید به گونه‌ای تعیین شود که بتواند مشکل کاهش تنوع را در دسته پس از تغییر محیط به سرعت از بین ببرد و پس از آن کوچکتر شود تا دسته بتواند با انجام یک جستجوی محلی قوی‌تر به نتایج بهتری دست یابد. در الگوریتم پیشنهادی برای دستیابی به این هدف مقدار r_{cloud} برابر با نصف طول گام حرکت قله‌ها در نظر گرفته می‌شود [۶] سپس با استفاده از یک رهیافت خودتطبیقی مقدار این پارامتر کاهش می یابد.

```
For each swarm i
 For each Particle j
       initialize x_{i,j} , v_{i,j}
       Pbesti,j = x_{i,j}
 endfor
Gbest = arg min_{Pbest_{i,j}} f(Pbest_{i,j})
endfor
Initialize Test_point
repeat:
for each swarm i
  for each particle j
    Update X_{i,j}(t+1) and Pbest_{i,j}(t+1)
  endfor
Gbest = arg min_{Pbest_{i,j}} f(Pbest_{i,j})
Endfor
for each swarm i
  For cnt=1 to Try_number
    Update Q_{i,j} based on equation 1
    If f(Q_{i,j}) > f(Gbest_i) THEN
       Gbesti = Q_{i,j}
  Endfor
Endfor
Update r_cloud based on equations 2 & 3
Evaluate Test_point
IF new value is different from last
iteration THEN
   Reinitialize r_cloud
   Reevaluate each particle attractor
   Update swarm attractor
Execute exclusion and anti-convergence[5]
until stopping criterion is met
```

شکل(۱): شبه‌کد رهیافت پیشنهادی.

روال این رهیافت خودتطبیقی بدین صورت است که در هر دسته پس از اجرای جستجوی ذره کوانتوم، درصد موفقیت‌های این ذره سنجیده می‌شود. درصد موفقیت یعنی نسبت تعداد دفعاتی که در Try-number بار تلاش ذره برای بهبود موقعیت Gbest، ذره توانسته موفق به بهبود موقعیت Gbest شود به کل دفعات تلاش یعنی Try-number که این نسبت را نرخ موفقیت می‌نامیم. سپس بر اساس نرخ

موفقیت ضریبی به نام ضریب فشردگی که با استفاده از رابطه ۲ بدست می‌آید را محاسبه می‌کنیم:

$$CF_i = CF_{\min} + ((1 - CF_{\min}) \times (S_i)) \qquad (۲)$$

که در آن S_i برابر با نرخ موفقیت ذره کوانتوم دسته iام در تکرار جاری است. CF_i برابر با ضریب فشردگی دسته iام در پس از اجرای جستجوی ذره کوانتوم است. CF_{min} برابر با حد پایین مقدار ضریب فشردگی است که عددی مثبت و کوچکتر از یک است. بنابراین با استفاده از رابطه ۲ ضریب فشردگی عددی در بازه $[CF_{min}, ۱]$ خواهد بود تا بتواند با استفاده از رابطه ۳ مقدار r_{cloud} را در هر تکرار کاهش دهد. بنابراین با استفاده از رابطه ۳ مقدار r_{cloud} در هر تکرار کاهش می‌یابد تا الگوریتم با افزایش توانایی جستجوی محلی‌اش به نتایج بهتری دست یابد.

$$r_{Cloud,i}(t+1) = r_{Cloud,i}(t) \times CF_i \qquad (۳)$$

دلیل استفاده از یک حدپایین در رابطه ۲ این است که کوچکتر شدن بیش از حد مقدار r_{cloud} موجب کاهش شدید کارایی ذره کوانتوم شود. برای مثال در صورتی که نرخ موفقیت ذره کوانتوم صفر باشد باعث صفر شدن r_{cloud} و در نتیجه از کار افتادن ذره کوانتوم شود. در رابطه (۳) مقدار r_{cloud} در هر تکرار به صورت خودتطبیقی بر اساس مقدار این پارامتر در تکرار قبل بدست می‌آید. شایان ذکر است که مقدار r_{cloud} پس از هر تغییر محیط مقداردهی اولیه می‌شود. بدین ترتیب در الگوریتم پیشنهادی بین توانایی افزایش تنوع و توانایی جستجوی محلی تعادل برقرار شده است و الگوریتم می‌تواند هر دوی آنها را به خوبی انجام دهد.

در محیط‌های پویا معمولاً فرکانس تغییر محیط بر اساس تعداد ارزیابی شایستگی الگوریتم ها تعریف می‌شود. بدون در نظر گرفتن این موضوع، هر چه مقدار Try_number بیشتر باشد، توانایی جستجوی الگوریتم بالاتر می‌رود و الگوریتم می‌تواند در تعداد تکرارهای کمتری به نتایج بهتر دست یابد اما با بالا بردن مقدار این پارامتر، تعداد ارزیابی شایستگی انجام شده توسط الگوریتم در هر تکرار بیشتر می‌شود و پیش از آنکه دسته‌ها بتوانند به نتایج بهتر دست یابند محیط تغییر خواهد کرد. برای رفع این مشکل، در الگوریتم پیشنهادی مقدار Try_number برای دسته‌ای که بهترین مقدار شایستگی را در میان دسته‌ها دارد بیشتر از سایر دسته‌ها در نظر گرفته می‌شود. این رهیافت باعث می‌شود در اطراف بهترین موقعیت یافت شده توسط دسته‌ها، جستجوی محلی قوی‌تری انجام شود و از طرف دیگر به دلیل متعادل بودن مقدار Try_number برای دیگر دسته‌ها، مشکل تعداد ارزیابی شایستگی زیاد در هر تکرار را رفع شود.

در الگوریتم پیشنهادی، برای کشف تغییر در محیط، یک نقطه در ابتدای اجرای الگوریتم در فضای مسئله در نظر گرفته می‌شود و در پایان هر تکرار مقدار شایستگی آن سنجیده می‌شود. در صورتی که تغییری در محیط رخ داده باشد مقدار شایستگی این نقطه تغییر کرده است. پس از کشف تغییر محیط ابتدا مقدار r_{cloud} مقداردهی اولیه می‌شود و موقعیت بهترین تجربه شخصی و گروهی ذرات مجدداً مورد ارزیابی قرار می‌گیرد تا مقادیر حافظه معتبر شوند.

شبه‌کد رهیافت پیشنهادی در شکل (۱) نشان داده شده است.

۳- نتایج آزمایشات

برای ارزیابی صحت و کارایی الگوریتم پیشنهادی، این الگوریتم همراه با شش الگوریتم شناخته‌شده به نام‌های mQSO [۵]، mCPSO [۵]، Cellular PSO [۹]، SPSO [۱۰]، RPSO [۱۱] و rSPSO [۱۲] بر روی MPB مورد مقایسه قرار گرفته اند. همچنین دو الگوریتم mQSO و mCPSO همراه با ضدهمگرایی نیز در مقایسات آورده شده‌اند. نتایج با توجه به پارامترهای جدول (۱) که با نام سناریو ۲ در مسئله MPB معروف است آورده شده است[۸]. تنها پارامتر متفاوت، تعداد قله‌ها می‌باشد که برای ارزیابی بهتر بین روش‌ها از ۱ قله تا ۲۰۰ قله در فرکانس تغییر ۵۰۰۰ ارزیابی شایستگی در نظر گرفته شده است.

جدول (۱) : پارامترهای MPB

پارامتر	مقدار
Mتعداد قله ها	بین ۱ تا ۲۰۰
فرکانس تغییر	۵۰۰۰
میزان تغییر ارتفاع	۷.۰
میزان تغییر عرض	۱.۰
شکل قله	cone
تابع اولیه	ندارد
s طول جابجایی	۱.۰
N تعداد ابعاد	۵
محدوده مکانی قله ها	[۰.۰,۱۰۰.۰]
محدوده پارامتر ارتفاع	[۳۰.۰,۷۰.۰]
محدوده پارامتر عرض	[۱,۱۲]
مقدار ارتفاع اولیه قله ها	۵۰

در آزمایشات در الگوریتم پیشنهادی ۱۰ دسته وجود دارد و تعداد ذرات در هر دسته برابر ۵ و مقدار Try_number نیز برابر ۵ در نظر گرفته شده است و مقدار این پارامتر برای بهترین دسته برابر ۵۰ در نظر گرفته شده است. مقدار اولیه r_{cloud} برابر با نصف مقدار طول جابه‌جایی قله‌ها در نظر گرفته می‌شود. تنظیمات PSO ، انحصار و ضدهمگرایی مطابق با [۵] انجام شده است. مقدار CF_{min} برابر ۰،۶ در

نظر گرفته شده است. مقدار این پارامترها بر اساس منابع ذکرشده و آزمایشات بسیار زیادی که در این زمینه انجام داده‌ایم تنظیم شده‌اند. آزمایشات ۳۰ بار انجام شده‌اند و نتایج بدست آمده از آزمایشات (خطای برون‌خطی و خطای استاندارد[۴]) بر روی MPB با ۱۰۰ بار تغییر محیط در هر آزمایش بر روی تعداد قله‌های مختلف با فرکانس تغییرات محیط ۵۰۰۰ ارزیابی شایستگی در جدول (۲) نشان داده شده است. خطای استاندارد در کنار خطای برون‌خطی درون پرانتز نشان داده شده است. دو نتیجه بهتر در هر سطر پررنگ شده است و بهترین نتیجه به صورت کج می‌باشد. در این جدول ها P برابر تعداد قله‌ها می‌باشد. دو الگوریتم mQSO و mCPSO همراه با ضدهمگرایی با ستاره مشخص شده‌اند. همانطور که در جدول (۲) مشاهده می‌شود، الگوریتم پیشنهادی در مجموع به نتایج بهتری نسبت به دیگر الگوریتم‌ها دست یافته است. در واقع الگوریتم پیشنهادی در تمام موارد جزو دو الگوریتم برتر است و تنها در سه مورد در رتبه دوم قرار گرفته است. نتیجه به دست آمده در مسئله با ۱۰ قله بسیار قابل توجه است و اختلاف زیادی با دیگر الگوریتم‌ها دارد. دلیل این امر این است که در این حالت تعداد دسته‌ها و قله‌ها با هم برابر است و در نتیجه هر دسته می‌تواند یک قله را پوشش دهد و در نهایت تمام قله‌ها تحت پوشش دسته‌ها قرار می‌گیرند.

هنگامی که تعداد دسته‌ها بیشتر از تعداد قله‌ها باشد دسته‌ها بر سر قله‌ها با یکدیگر به رقابت می‌پردازند و در نتیجه با استفاده از انحصار یکدیگر را از قله‌ها بیرون می‌کنند، که در نتیجه آن نتایج بدست آمده، کمتر از حد انتظار برای تعداد قله‌های پایین می‌باشد. همچنین هنگامی که تعداد قله‌ها بیش از تعداد دسته‌ها باشد، تعدادی از قله‌ها بدون پوشش می‌مانند و در نتیجه نتایج بدتر می‌شوند. البته مکانیزم ضدهمگرایی باعث بهبود این مشکل شده است اما نمی‌تواند باز هم جلوی کاهش کیفیت نتایج را بگیرد.

جدول(۲) : خطای برون خطی و خطای استاندارد الگوریتم ها در فرکانس تغییرات ۵۰۰۰ با تعداد قله های مختلف.

P	Cellular PSO	RPSO	mCPSO*	mQSO*	mCPSO	mQSO	rSPSO	SPSO	الگوریتم پیشنهادی
۱	۲.۵۵(۰.۱۲)	***۰,۵۶(۰,۰۴)***	۴,۹۳(۰,۱۷)	۵,۰۷(۰,۱۷)	۴,۹۳(۰,۱۷)	۵,۰۷(۰,۱۷)	۱,۴۲(۰,۰۶)	۲,۶۴(۰,۱۰)	**۱.۲۶(۰.۰۹)**
۵	۱.۶۸(۰.۱۱)	۱۲,۲۲(۰,۷۶)	۲,۰۷(۰,۰۸)	۱,۸۱(۰,۰۷)	۲,۰۷(۰,۰۸)	۱,۸۱(۰,۰۷)	***۱,۰۴(۰,۰۳)***	۲,۱۵(۰,۰۷)	**۱.۰۸(۰.۱۰)**
۱۰	۱.۷۸(۰.۰۵)	۱۲,۹۸(۰,۴۸)	۲,۰۸(۰,۰۷)	۱,۸۰(۰,۰۶)	۲,۰۵(۰,۰۷)	۱,۷۵(۰,۰۶)	**۱,۵۰(۰,۰۸)**	۲,۵۱(۰,۰۹)	***۰,۹۵(۰.۰۵)***
۲۰	۲.۶۱(۰.۰۷)	۱۲,۷۹(۰,۵۴)	۲,۶۴(۰,۰۷)	۲,۴۲(۰,۰۷)	۲,۹۵(۰,۰۸)	۲,۷۴(۰,۰۷)	***۲,۲۰(۰,۰۷)***	۳,۲۱(۰,۰۷)	**۲.۲۷(۰.۱۲)**
۳۰	۲.۹۳(۰.۰۸)	۱۲,۳۵(۰,۶۲)	۲,۶۳(۰,۰۸)	**۲,۴۸(۰,۰۷)**	۳,۳۸(۰,۱۱)	۳,۲۷(۰,۱۱)	۲,۶۲(۰,۰۷)	۳,۶۴(۰,۰۷)	***۲.۴۱(۰.۱۱)***
۴۰	۳.۱۴(۰.۰۸)	۱۱,۳۷(۰,۴۱)	۲,۶۷(۰,۰۷)	**۲,۵۵(۰,۰۷)**	۳,۶۹(۰,۱۱)	۳,۶۰(۰,۰۸)	۲,۷۶(۰,۰۸)	۳,۸۵(۰,۰۸)	***۲.۴۹(۰.۱۳)***
۵۰	۳.۲۶(۰.۰۸)	۱۱,۳۴(۰,۲۹)	۲,۶۵(۰,۰۶)	**۲,۵۰(۰,۰۶)**	۳,۶۸(۰,۱۱)	۳,۶۵(۰,۱۱)	۲,۷۲(۰,۰۸)	۳,۸۶(۰,۰۸)	***۲.۴۲(۰.۰۹)***
۱۰۰	۳.۴۱(۰.۰۷)	۹,۷۳(۰,۲۸)	۲,۴۹(۰,۰۴)	**۲,۳۶(۰,۰۴)**	۴,۰۷(۰,۰۹)	۳,۹۳(۰,۰۸)	۲,۹۳(۰,۰۶)	۴,۰۱(۰,۰۷)	***۲.۳۰(۰.۰۷)***
۲۰۰	۳.۴۰(۰.۰۶)	۸,۹۰(۰,۱۹)	۲,۴۴(۰,۰۴)	**۲,۲۶(۰,۰۳)**	۳,۹۷(۰,۰۸)	۳,۸۶(۰,۰۷)	۲,۷۹(۰,۰۵)	۳,۸۲(۰,۰۵))	***۲.۲۴(۰.۰۵)***

۴- نتیجه‌گیری

در این مقاله یک روش نوین برای بهینه‌سازی در محیط‌های پویا بر اساس بهینه‌سازی دسته ذرات پیشنهاد شد و نتایج بر روی تابع بنچمارک حرکت قله‌ها با چندین روش شناخته شده دیگر مورد مقایسه قرار گرفت. نتایج آزمایشات نشان داد که الگوریتم پیشنهادی از کارایی قابل قبولی مخصوصاً هنگامی که تعداد دسته‌ها با تعداد قله‌ها برابر باشد برخوردار است. بنابراین می‌توان به این نتیجه رسید که برای بهبود نتایج الگوریتم پیشنهادی، تعداد دسته‌ها را نیز باید به صورت تطبیقی بر اساس تعداد قله‌های یافت شده در فضای مسئله تغییر داد تا هر قله توسط یک دسته پوشش داده شود. این راهکار باید جایگزین راهکار ضدهمگرایی شود.

مراجع

[1] Y. Jin and J. Branke, "Evolutionary Optimization in uncertain environments –A Survey", in IEEE Transaction on Evolutionary Computation, vol. 9 , No. 3 , pp. 303-317, 2005.

[2] C. Li and S. Yang, "Fast Multi-Swarm Optimization for Dynamic Optimization Problems", In 4th International Conference on Natural Computation, Jinan, Shandong, China, vol. 7, pp.624–628, 2008.

[3] S. Yang and C. Li, "A Clustering Particle Swarm Optimizer for Locating and Tracking Multiple Optima in Dynamic Environments", in IEEE Transaction on Evolutionary Computation, pp. 1-16, 2010.

[4] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", in IEEE International Conference on Neural Networks, Vol. 4, pp. 1942-1948, Perth, November 1995.

[5] T. Blackwell and J. Branke, "Multiswarm, Exclusion, and Anti-Convergence in Dynamic Environment", in IEEE

Transaction on Evolutionary Computation, Vol. 10, No. 4, pp. 459-472, 2006.
[6] T. Blackwell and J. Branke, "Particle Swarms for Dynamic Optimization Problems", in Swarm Intelligence, pp.193–217, 2008.
[7] W. Du and B. Li, "Multi-Strategy Ensemble Particle Swarm Optimization for Dynamic Optimization", in Information Sciences: an International Journal Vol.178, pp.3096–3109, 2008.
[8] http://www.aifb.unikarlsruhe.de/~jbr/MovPeaks/
[9] B. Hashemi and M. R. Meybodi, "Cellular PSO: A PSO for Dynamic Environments", in Advances in Computation and Intelligence, Lecture Notes in Computer Science, vol. 5821, pp. 422-433, 2009.
[10] D. Parrott and X. Li, "Locating and Tracking Multiple Dynamic Optima by A Particle Swarm Model Using Speciation," in IEEE Transaction on Evolutionary Computation, vol. 10, No. 4, pp. 440–458, 2006.
[11] X. Hu, and R. C. Eberhart, "Adaptive particle Swarm Optimization: Detection and Response to Dynamic Systems," in IEEE Congress on Evolutionary Computation, Honolulu, HI, USA, 2002, pp. 1666-1670.
[12] S. Bird and X. Li, "Using regression to improve local convergence," in Proc. IEEE Congr. Evol. Comput., 2007, pp. 592–599.
[13] Y. Shi and R. Eberhart, "A Modified Particle Swarm Optimization", In IEEE International Conference on Evolutionary Computation Proceedings, pp. 69-73, Anchorage, 1998.

زیرنویس‌ها

[1] Moving Peak Benchmark
[2] Offline Error
[3] Multi swarm
[4] Exclusion
[5] Anti Convergence
[6] Standard Error