هجدهمین کنفرانس ملی سالانه انجمن کامپیوتر ایران
دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران، 22-24 اسفندماه 1391

یک الگوریتم ترکیبی از بهینه‌ساز گروه ذرات و جستجوگرهای محلی برای بهینه‌سازی در محیط‌های پویا

علی‌شریفی[1]، مهشید مهدویانی[2]، وحید نوروزی[3]،محمدرضا میبدی[4]

[1] کارشناس ارشد، دانشکده مهندسی کامپیوتر و فن‌آوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران،
alish@aut.ac.ir

[2] کارشناس، دانشکده مهندسی کامپیوتر و فن‌آوری اطلاعات، دانشگاه صنعتی امیرکبیر ، تهران
IA89231740@aut.ac.ir

[3] کارشناس ارشد، دانشکده مهندسی کامپیوتر و فن‌آوری اطلاعات، دانشگاه صنعتی امیرکبیر ، تهران
vnoroozi@aut.ac.ir

[4] دکتر، دانشکده مهندسی کامپیوتر و فن‌آوری اطلاعات، دانشگاه صنعتی امیرکبیر ، تهران
mmeybodi@aut.ac.ir

چکیده

اکثر قریب به اتفاق مسائل واقعی ذاتا با جنبه‌های گوناگونی از عدم اطمینان مواجه هستند. یکی از رایج‌ترین جنبه‌های عدم اطمینان، پویایی یا غیر ایستا بودن مسائل واقعی است. الگوریتم‌های بهینه‌سازی در مواجهه با محیط‌های پویا علاوه بر یافتن بهینه یا بهینه‌های محیط مکلف به تعقیب تنگاتنگ بهینه و یا بهینه‌های محیط و همچنین کشف بهینه‌های جدید ایجاد شده در محیط به دلیل تغییرات محیط هستند. الگوریتم‌های مبتنی بر جمعیت دارای توانایی اجتناب از همگرایی زودرس و گذر از برخی بهینه‌های محلی هستند، اما در سوی دیگر این الگوریتم‌ها قالبا از توانایی استخراج مناسبی برخوردار نیستند. الگوریتم‌های جستجوی محلی در مقایسه با الگوریتم‌های مبتنی بر جمعیت از توانایی اکتشاف کمتری برخوردار هستند ولی قالبا دارای توانایی استخراج مناسبی هستند. در این مقاله یک الگوریتم ترکیبی همکارانه از الگوریتم بهینه‌ساز گروه ذرات و جستجوگرهای محلی تحت عنوان جستجوی محلی هدایت شده توسط بهینه‌ساز گروه ذرات (PSOledLS) ارائه می‌شود. به منظور انجام جستجو محلی دو الگوریتم متداول جستجوی محلی به علاوه یک الگوریتم جستجوگر محلی پیشنهادی مورد استفاده قرار می‌گیرند. نتایج حاصل از آزمایش‌های انجام شده در محیط‌های پویای ایجاد شده توسط تولید کننده تابع محک قله‌های روان نشان دهنده کارایی بسیار مناسب الگوریتم پیشنهادی در مقایسه با بهترین الگوریتم‌های پیشنهاد شده برای محیط‌های پویا می‌باشد.

کلمات کلیدی

بهینه‌ساز گروه ذرات، استراتژی تکامل، جستجوی الگو، جستجوی مستقیم ساده لوحانه، محیط پویا، تابع محک قله‌های روان

1- مقدمه

1-1- بهینه‌سازی در محیط‌های پویا

در محیط‌های پویا هدف بهینه‌سازی، ابعاد مسئله، و یا برخی از محدودیت‌ها می‌توانند در امتداد زمان تغییر یابند. در چنین مواردی، هدف یک الگوریتم بهینه‌سازی تنها به یافتن پاسخ‌های ارضا کننده برای یک مسئله ثابت محدود نمی‌شود، بلکه تعقیب تا حد ممکن تنگاتنگ بهینه و یا بهینه‌های محیط نیز توانایی دیگری است که الگوریتم باید به آن مجهز باشد. چالش پیش رو استفاده از پاسخ‌های به دست آمده‌ی قبل از وقوع تغییر برای دستیابی سریع به پاسخ‌های محیط پس از تغییر است، این ویژگی به خصوص در کاربردهای آنلاین

و زمان حقیقی بسیار حائز اهمیت است. اما الگوریتم‌های کلاسیک مبتنی بر جمعیت و تکاملی پس از همگرایی قادر به تطبیق مناسب با تغییرات محیط نیستند، به همین دلیل باید با ایجاد تغییرات و بهبودهایی آنها را برای کاربرد در مسائل پویا تطابق داد.

به طور کلی الگوریتم‌های مبتنی بر جمعیت در مواجه با محیط‌های پویا با مشکل از دست دادن تنوع مواجه هستند، علاو بر این الگوریتم‌های دارای حافظه مانند بهینه‌ساز گروه ذرات با مشکل منسوخ شدن حافظه به دلیل بروز تغییر در محیط نیز دست به گریبان هستند. از این رو لازم است که سازوکارهایی جهت حفظ یا تزریق تنوع و ارزیابی مجدد و یا پاک کردن حافظه‌های منسوخ در این گونه الگوریتم‌ها تعبیه شود.

۱-۲- اصول بهینه‌ساز گروه ذرات

بهینه‌ساز گروه ذرات [1]، یک الگوریتم مبتنی بر جمعیت و تکراری است. تفاوت اصلی این الگوریتم با الگوریتم‌های تکاملی در حرکت راه‌حل‌های بالقوه (ذرات) بهینه‌سازی گروه ذرات نهفته است. در واقع در الگوریتم بهینه‌سازی گروه ذرات حرکت در فضای جستجو جایگزین تکامل از طریق تولید نسل شده‌است. یک ذره توسط مکان و سرعتش بازنمایی می‌شود. در هر تکرار، هر ذره مبتنی بر سرعت کنونی خود، مکانش، بهترین مکان مشاهد شده توسط خودش و بهترین مکان یافت شده توسط همسایگانش، خود را بهنگام می‌کند. این بهنگام‌رسانی را می‌توان توسط روابط زیر بیان نمود:

$$v_i(t+1) = \omega v_i(t) + c_1\xi(p_i(t)-x_i(t)) + c_2\eta(p_{gi}(t)-x_i(t)) \quad (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1), \quad (2)$$

که در این روابط، $v_i(t)$ و $x_i(t)$ به ترتیب بردارهایی معرف سرعت و مکان فعلی ذره i در تکرار t هستند، بهترین موقعیت مشاهده شده توسط ذره i و بهترین موقعیت مشاهده شده در همسایگی ذره i (ما از نسخه‌ی عمومی بهینه‌ساز گروه ذرات استفاده می کنیم که در آن تمامی ذرات با یکدیگر همسایه محسوب می‌شوند)، به ترتیب توسط بردارهای p_i و p_{gi} نمایش داده می‌شوند. ω ضریب اینرسی است که مشخص کننده تاثیر سرعت قبلی ذره بر تعیین سرعت فعلی آن است، c_1 و c_2 به ترتیب ضرایب یادگیری شناختی و اجتماعی هستند. ξ و η بردارهای تصادفی یکنواختی در بازه‌ی [۰،1] هستند.

۱-۳- جستجوگرهای محلی

۱-۳-۱- استراتژی تکامل (۱+۱)

در اوایل دهه 70 میلادی شوفل و ریچنبرگ گونه‌ای از الگوریتم‌های تکاملی مبتنی بر جهش تطبیقی را ابداع و تحت عنوان استراتژی‌های تکامل معرفی نمودند [2]، [3]، [4]. مهمترین مزیت استراتژی‌های تکامل نسبت به سایر الگوریتم‌های جستجوی محلی تنظیم تطبیقی اندازه گام جهش در این الگوریتم‌ها است. نسخه‌های کلاسیک و اولیه استراتژی‌های تکامل در هر تکرار تنها شامل دو عضو بودند (والد و فرزند) به همین دلیل به آنها استراتژی‌های تکاملی دو عضوی اطلاق می‌شود. البته بعدها نسخه‌های مبتنی بر جمعیت این الگوریتم‌ها نیز توسعه داده شد ولی از آنجا که ما تنها به عنوان جستجوگر محلی از استراتژی تکامل استفاده می‌کنیم، از نسخه استاندارد دو عضوی استراتژی تکامل با تطبیق گام جهش توسط قانون یک پنجم موفقیت استفاده می‌کنیم. شبه کد مربوط به نحوه تطبیق جهش و یک تولید نسل استراتژی تکامل (1+1) با تطبیق گام جهش توسط قانون یک پنجم موفقیت در شکل‌های (1) و (2) نمایش داده شده است.

شبه‌کد: تطبیق گام جهش توسط قانون یک پنجم موفقیت

```
PROCEDURE adaptMutation_OneFifthSuccessRule
BEGIN
IF currentGenerationIsSuccessful THEN
    P_s^New = ((k-1)/k) P_s^Old + 1/k
ELSE
    P_s^New = ((k-1)/k) P_s^Old
END-IF
IF P_s^New < 1/5 THEN
    σ^New ← σ^Old . θ
ELSE IF P_s^New > 1/5
    σ^New ← σ^Old / θ
ELSE
    σ^New ← σ^Old
END IF
END PROCEDURE
```

شکل 1)

شبه‌کد: استراتژی تکامل (1+1) با قانون یک پنجم موفقیت

```
PROCEDURE (1+1)_ES_OneFifthSeccessRule
BEGIN
   mutationVector ~σN(0,1)
   child ← parent + mutationVector
   IF child.fitness > parent.fitness THEN
      parent ← child
      currentGenerationIsSuccessful ← True
   ELSE
     currentGenerationIsSuccessful ← false
   END IF
CALL adaptMutation_OneFifthSuccessRule
END PROCEDURE
```

شکل 2)

۱-۳-۲- جستجوی الگوی هوک و جیوز

با وجود اینکه جستجوی الگوی هوک و جیوز بیش از 40 سال پیش پیشنهاد شده است [5]، اما همچنان در زمره اولین انتخاب‌های محققین به عنوان یک جستجوگر محلی قطعی محسوب

می‌شود.جستجوی الگو متعلق به خانواده‌ای از روش‌های بهینه‌سازی عددی است که برای بهینه‌سازی به محاسبه‌ی شیب تابع هدف نیازی ندارند، به همین دلیل ابزارهایی مناسب برای بهینه‌سازی توابع نا پیوسته یا مشتق ناپذیر محسوب می‌شوند. این الگوریتم‌های جستجو تحت عنوان جستجوی مستقیم، جستجوی مستقل از مشتق و یا جعبه‌ی سیاه شناخته می‌شوند. هنگامی که از جستجوی الگوی HJ برای بهینه‌سازی در محیط‌های چند قله‌ای استفاده شود به شدت مستعد ابتلا به رکود در نزدیکترین بهینه‌ی محلی است. بنا بر این اغلب از آن به صورت ترکیبی با یک استراتژی جستجوی عمومی استفاده می‌شود.

جستجو الگوی HJ از یک مکان تحت عنوان نقطه‌ی پایه، X، با حرکات اکتشافی که در آن همه‌ی ابعاد به نوبت با یک گام خاص مورد بررسی قرار می‌گیرند شروع به کار می‌کند. در ابتدا در هر بعد در یک جهت تصادفی یک حرکت با اندازه‌ی گام مشخص انجام می‌شود و یک نامزد جدید، X'، برای نقطه‌ی پایه ایجاد می‌شود. اگر شایستگی X' بیش از شایستگی نقطه‌ی پایه فعلی یعنی X باشد آنگاه، X توسط X' جایگزین شده و حرکت اکتشافی در ابعاد بعدی ادامه می‌یابد، در غیر این صورت X' با حرکتی در همان بعد از نقطه‌ی پایه X و با همان اندازه‌ی گام جهش ولی در جهت عکس جایگزین می‌شود، اگر بهبودی حاصل شود X توسط X' جایگزین می‌شود و حرکت اکتشافی در ابعاد بعدی ادامه می‌یابد در غیر این صورت با شکست در هر دو جهت در بعد فعلی، بدون تغییری در نقطه‌ی پایه X تنها حرکت اکتشافی در ابعاد بعدی ادامه می‌یابد. اعمال یک دور حرکات اکتشافی حد اقل به اندازه‌ی ابعاد مسئله (d) و حد اکثر به اندازه‌ی دو برابر ابعاد مسئله (2d) محاسبه شایستگی نیاز دارد.

هنگامی که حرکت اکتشافی در همه ابعاد به پایان می‌رسد در صورتی که بهبودی صورت پذیرفته باشد، آنگاه تمامی حرکات منجر به بهبود مجددا تکرار می‌شود که این تکرار حرکات موفقیت آمیز حرکت الگو نام دارد، و در نتیجه‌ی این تکرار حرکات یک نقطه‌ی جدید $\overline{X}$ به دست می‌آید. صرف نظر از اینکه $\overline{X}$ از شایستگی بالاتری نسبت به X برخوردار است یا خیر، این حرکت به صورت موقت پذیرفته می‌شود و $\overline{X}$ موقتا به عنوان نقطه‌ی پایه در نظر گرفته می‌شود و یک دور حرکات اکتشافی روی این نقطه‌ی پایه‌ی موقت انجام می‌پذیرد. در طول اعمال این حرکات اکتشافی هر گاه بهبودی نسبت به $\overline{X}$ حاصل شود $\overline{X}$ بهنگام می‌شود و در پایان اگر نسبت به نقطه‌ی پایه اصلی X نیز بهبودی حاصل شده باشد حرکت الگو و حرکات اکتشافی بعد از آن پذیرفته شده و نقطه‌ی پایه‌ی اصلی یعنی X توسط $\overline{X}$ بهنگام می‌شود. اگر در یک تکرار در حرکات اکتشافی ابتدایی هیچ بهبودی حاصل نشود، بدیهی است که حرکت الگوی بعدی نیز لغو می‌شود در چنین حالتی مشخص است در تمامی ابعاد فاصله نقطه‌ی پایه از مکان بهینه در حال استخراج کمتر از اندازه‌ی گام حرکت کنونی است. در این حالت گام جهش کاهش می‌یابد و مجددا فرایند فوق ادامه می‌یابد. این روال تا جایی که اندازه‌ی گام جهش به اندازه‌ی دلخواه کوچک شود و یا به بیان دیگر نقطه‌ی پایه به اندازه‌ی کافی به بهینه‌ی عمومی نزدیک شود ادامه می‌یابد.

۱-۳-۳- جستجوی مستقیم ساده لوحانه

در جستجوی الگوی HJ هنگامی که در یک تکرار الگوریتم حرکت حتی در یک بعد با بهبودی مواجه شود آنگاه در تکرار بعد جستجو در ابعاد دیگر با همان اندازه‌ی گام قبلی ادامه می‌یابد، حتی اگر در تکرارهای قبلی با آن اندازه‌ی گام حرکت در هر دو جهت آن ابعاد با شکست مواجه شده باشد. در واقع اگر بین ابعاد مسئله وابستگی وجود داشته باشد ممکن است حرکت در ابعاد دیگر موجب شود که حرکت در بعدی که با شکست مواجه شده است نیز منجر به بهبود شود. اما در مسائلی که بین ابعاد آنها وابستگی وجود ندارد (کانتور گستره‌ی قله‌های آن محیط بیضوی باشد به گونه‌ای که محورهای بیضی‌های کانتور موازی با محورهای فضای مسئله باشد) آنگاه هنگامی که با یک اندازه‌ی گام مشخص در یک بعد با شکست مواجه می‌شویم حتی موفقیت و حرکت در راستای ابعاد دیگر تاثیری بر ایجاد فرصت بهبود در بعد شکست خورده نخواهد داشت.

این ایده ما را به سمت پیشنهاد یک الگوریتم جستجوی مناسب برای اینگونه محیط‌ها سوق داد. الگوریتم جستجوی پیشنهادی ما دارای دو تفاوت با الگوریتم جستجوی الگوی HJ است. نخست هنگامی که با یک اندازه‌ی گام مشخص در یک بعد با شکست مواجه می‌شویم، حرکات اکتشافی در آن بعد تا زمانی که در تمامی ابعاد دیگر نیز با آن اندازه‌ی گام مشخص با شکست مواجه شویم و اندازه گام حرکت را کاهش دهیم متوقف می‌شود. دوم، عمده‌ی مزیت حرکات الگوی موجود در الگوریتم جستجوی الگوی HJ افزایش قدرت اکتشاف الگوریتم و افزایش سرعت حرکت به سمت بهینه‌ی محلی است در حالیکه در الگوریتم ترکیبی پیشنهادی ما وظیفه‌ی برداشتن گام‌های بلند به سمت بهینه‌ها و اکتشاف به عهده الگوریتم بهینه‌ساز گروه ذرات است، از این رو ما حرکات الگو را از جستجوی محلی خود حذف کرده‌ایم. شکل (۴) شبه کد مربوط به الگوریتم جستجوی محلی پیشنهاد شده که آن را جستجوی مستقیم ساده‌لوحانه NDS می‌نامیم نمایش می‌دهد.

۲- کارهای مرتبط

در طول دهه پیش، رویکردهای بسیاری جهت توسعه و بهبود الگوریتم‌های مبتنی بر جمعیت و به ویژه الگوریتم‌های تکاملی برای محیط‌های پویا ارائه شده‌است. برای دسترسی به یک مرور کلی بر

رویکردها، روش‌ها و الگوریتم‌های ارائه شده به [6]، [7]، [8]، [9] و [10] مراجعه شود.

بهینه‌سازی گروه ذرات، به عنوان یکی از موفق‌ترین الگوریتم‌های مبتنی بر جمعیت و هوش گروهی که از شبیه‌سازی رفتار اجتماعی گروهی برخی حیوانات، برای حل مسئله استفاده می‌کند، یکی از فعال‌ترین زمینه‌های تحقیقاتی در طول دهه گذشته بوده‌است. بهینه سازی گروه ذرات در بسیاری از مسائل بهینه‌سازی به کار گرفته شده و نتایج خوبی را به نمایش گذاشته است. در سال‌های اخیر به صورت گسترده از بهینه‌ساز گروهی ذرات برای حل مسائل پویا نیز استفاده شده است. برای مراجعه به یک مرور و دسته‌بندی مناسب بر کارهای گذشته‌ی مبتنی بر بهینه‌ساز گروه ذرات برای در محیط‌های پویا به [8], [11] مراجعه شود. در ادامه این بخش به معرفی چند الگوریتم که در این مقاله الگوریتم پیشنهادی خود را با آنها مقایسه می‌کنیم می‌پردازیم.

روش‌های چند جمعیتی از رایج‌ترین راهبردهای مورد استفاده در محیط‌های پویا محسوب می‌شوند. بلکول و برنکی در [12] از یک مجموعه از گروه‌های متعامل برای تعقیب چندین بهینه به صورت همزمان استفاده کردند. در آن الگوریتم که بهینه‌ساز گروه ذرات کوانتومی چند جمعیتی (mQSO) نامیده شده است، دو عملگر ویژه، دفع و ضد همگرایی به ترتیب وظیفه ممانعت از همگرایی چند گروه به یک بهینه یکسان و افزایش ظرفیت اکتشاف برای یافتن قله‌های جدید در فضای جستجو را به عهد دارند. علاوه بر این، ذرات باردار و یا ذرات کوانتومی به حفظ تنوع در سطح میان ذرات درون هر زیر گروه کمک می‌کنند.

در [13] الگوریتمی ترکیبی از الگوریتم تکاملی تفاضلی و بهینه‌ساز گروه ذرات، تحت عنوان بهینه‌سازی همکارانه تکاملی- گروهی (CESO) پیشنهاد شده است. در این الگوریتم یافتن بهینه‌ی عمومی محیط بر عهده‌ی گروه ذرات است در حالی که یک نسخه ازدحامی الگوریتم تکاملی تفاضی موظف به حفظ تنوع و استقرار بر بهینه‌های محلی محیط است. در هر تکرار اگر بهترین خاطره‌ی گروهی ذرات از بهترین عضو جمعیت تکاملی شایسته‌تر باشد، بهترین جمعیت تکاملی توسط بهترین خاطره‌ی گروه ذرات جایگزین می‌شود. همچنین هر گاه فاصله‌ی بهترین خاطره‌ی گروه ذرات و بهترین موجود جمعیت تکاملی از حد معینی کمتر شود یا تغییری در محیط شناسایی شود، گروه ذرات در موقعیت کنونی جمعیت تکاملی تفاضلی ازدحامی جایگذاری مجدد می‌شوند.

الگوریتم‌های ممتیک نیز به دلیل ادغام جستجوی محلی و جستجوی عمومی از ظرفیت مناسبی در بهینه‌سازی در محیط‌های پویا برخوردار هستند. در [14] یک الگوریتم ممتکیک مبتنی بر بهینه‌ساز گروه ذرات تحت عنوان LPSO-SOFCMA ارائه شده‌است. در آن الگوریتم یک نسخه محلی از الگوریتم بهینه‌سازی گروه ذرات با یک ساختار توپولوژی حلقوی به عنوان عملگر جستجوی عمومی و یک روش جستجوی محلی شناختی فازی به عنوان جستجوگر محلی مورد استفاده قرار گرفته است. علاوه بر این، به منظور افزایش بیشتر توانایی اکتشاف برای یافتن بهینه‌های جدید در فضای جستجو، یک شیوه‌ی مهاجرت تصادفی خودسازمانده توسعه داده شده و در الگوریتم تعبیه شده است.

افراز فضای جستجو نیز یکی از رویکردهای موفق و جدید در زمینه بهینه‌سازی در محیط‌های پویا است. در [15] یک الگوریتم مبتنی بر افراز فضای جستجو تحت عنوان cellular PSO ارائه شده که در آن یک اتوماتای سلولی در فضای افراز شده تعبیه می‌شود که تعامل میان قسمت‌های مختلف و کنترل تراکم و توزیع ذرات در فضای جستجو را عهده‌دار می‌شود. در واقع ذرات موجود در یک سلول یک زیر گروه غیر صریح را تشکیل می‌دهند که وظیفه‌ی جستجوی بهینه‌های احتمالی موجود در همسایگی همان سلول را به عهده خواهند داشت. هنگامی که تراکم ذرات موجود در یک سلول از یک حد آستانه معین بیشتر شود، تعدادی از ذرات موجود در سلول به سلول‌های دیگری که به صورت تصادفی انتخاب می‌شوند ارسال می‌شوند.

در [11] یک نسخه بهبود یافته این الگوریتم ارائه شده است، که در آن پارامتر حد آستانه تراکم ذرات درون سلول حذف شده است. در این الگوریتم که بهینه‌ساز گروه ذرات سلولی دو فازه (TP-CPSO) نامیده می‌شود برای سلول‌ها دو وضعیت عملیاتی متفاوت در نظر گرفته می‌شود، در یک وضعیت که فاز اکتشاف نامیده می‌شود، الگوریتم بهینه‌ساز گروه ذرات وظیفه اکتشاف در همسایگی سلول را به عهده خواهد داشت. در فاز اکتشاف کنترل تراکم در سلول صورت نمی‌پذیرد. پس از پایان فاز اکتشاف در یک سلول، فاز استخراج اجرا می‌شود که در آن از یک جستجوگر محلی به منظور استخراج دقیق تر پاسخ به دست آمده در فاز اکتشاف استفاده می‌شود. در فاز استخراج تراکم مجاز سلول 1 می‌باشد و سایر ذراتی که در سلول موجود هستند و یا وارد سلول می‌شوند به صورت تصادفی یکنواخت به سلول‌های دیگر ارسال می‌شوند.

۳- الگوریتم پیشنهادی (PSOledLS)

در این مقاله یک الگوریتم ترکیبی از بهینه‌ساز گروه ذرات و چند جستجوگر محلی برای بهینه‌سازی در محیط‌های پویا پیشنهاد شده است. در این الگوریتم از بهینه‌ساز گروه ذرات به منظور اکتشاف و مکان یابی نسبی بهینه‌های نوید بخش‌تر محیط استفاده می‌شود. برای استقرار همزمان بر چندین قله‌ی محیط پس از اتمام اکتشاف یک قله توسط بهینه‌ساز گروه ذرات از یک عامل جستجوگر محلی که از استراتژی تکامل (۱+۱) با تطبیق گام جهش توسط قانون یک پنجم موفقیت استفاده می‌کند بهره می‌بریم. برای استخراج با سرعت بیشتر و دقیق‌تر بهترین بهینه‌یافت شده‌ی کنونی از جستجوی مستقیم ساده

لوحانه (NDS) که در همین مقاله پیشنهاد شده است استفاده می‌کنیم. از آنجایی معیار خطای برون خط تنها مبتنی بر کاهش فاصله از بهینه‌ی عمومی محیط است، هنگامی که اندازه‌ی گام جهش در عامل‌های استراتژی تکامل دو عضوی از حد معین $\sigma_{\min}$ کمتر شود، ادامه تولید نسل توسط آنها را متوقف می‌کنیم در این حالت تنها وظیفه‌ی این عامل‌ها پاسبانی از قله‌ای است که بر روی آن مستقر شده‌اند. استخراج بلندترین قله‌ی یافت شده توسط جستجوی الگوی HJ یا جستجوی مستقیم ساده لوحانه NDS هرگز متوقف نمی‌شود. در ادامه جزئیات بیشتری از چگونگی انجام مراحل مختلف الگوریتم ارائه می‌شود.

۳-۱- جستجوی عمومی و مکان‌یابی برای استخراج

در PSOledLS از یک گروه ذرات برای اکتشاف و مکان یابی نسبی قله‌ها استفاده می‌شود. پس از هر بار همگرایی نسبی گروه ذرات، انجام عملیات بهینه‌سازی توسط گروه ذرات متوقف می‌شود، در صورتی که در شعاع r از بهینه‌ی عمومی گروه ذرات p_g هیچ عامل جستجوگر محلی وجود نداشته باشد یک عامل جستجو گر محلی در مکان p_g مستقر می‌شود. وظیفه این عامل‌های جستجوگر محلی نگهبانی و استخراج دقیق‌تر قله‌ی یافت شده می‌باشد. سپس گروه ذرات مجددا به صورت تصادفی در فضای مسئله مقداردهی اولیه می‌شوند و فرایند اکتشاف قله‌های دیگر محیط را انجام می‌دهند.

سوال این است که همگرایی نسبی گروهی ذرات چگونه شناسایی می‌شود؟ در واقع معیارهای متفاوتی می‌توان برای شناسایی همگرایی نسبی بیان نمود. برای مثال هنگامی که میانگین سرعت ذرات از حد معینی کمتر شود، یا برای چندین تکرار متوالی بهبود در بهینه‌ی عمومی گروه از حد معینی کمتر باشد و یا بیشترین فاصله دو به دوی میان ذرات از حد معینی کمتر شود. در الگوریتم پشنهادی ما هنگامی که بیشترین فاصله‌ی همه‌ی ذرات گروه از بهترین خاطره‌ی کل گروه (شعاع گروه) کمتر از حد معین δ باشد به منزله‌ی همگرایی نسبی گروه قلمداد می‌شود.

۳-۲- جستجوی محلی، استخراج و دنبال کردن بهینه‌های محیط

پس از اتمام اکتشاف نسبی یک قله، نگهداری و استخراج دقیق‌تر بهینه یافت شده توسط استراتژی تکامل (1+1). برای استخراج دقیق‌تر بهترین بهینه‌ی یافت شده پس از آخرین تغییر در محیط از جستجوی مستقیم ساده لوحانه NDS استفاده می‌شود. در جستجوی NDS تنظیم خودکار گام حرکت تنها در جهت کاهش آن صورت می‌پذیرد در حالی که در استراتژی تکامل (1+1) این تطبیق هم در جهت افزایش گام جهش و هم در جهت کاهش آن صورت می‌پذیرد. در واقع استراتژی تکاملی دارای این مزیت عمده است که پس از بروز تغییر در محیط به صورت خودکار گام جهش خورد را افزایش می‌دهد و به تعقیب بهینه‌ی مستقر بر آن ادامه می‌دهد در حالی که در NDS حتما باید پس از بروز تغییر در محیط و شناسایی تغییر مقدار گام جهش مقدار دهی اولیه شود.

شبه‌کد: جستجوی مستقیم ساده لوحانه (NDS)

```
PROCEDURE HJ_PatternSerach
BEGIN
    stepsCounter ← 0
    Randomly initialize position of Base Point X
    FOR_EACH Dimension j select Dir_j Randomly from {-1, 1}
    failedDims ← NULL
    Repeat
        stepSize ← initalStepSize . discountFactor^stepsCounter
        FOR_EACH Dimension j that j ∉ failedDims DO
            X_j' = X_j + Dir_j . stepSize
            IF X'.fitness > X.fitness THEN
                X ← X'
            ELSE
                Dir_j ← -1 . Dir_j ;      X_j' = X_j + Dir_j . stepSize
                IF X'.fitness > X.fitness THEN X ← X'
                ELSE
                    ADD j to failedDims
                END IF
            END IF
        END FOR
        IF |failedDims| equals numberOfDims THEN
            increase stepCounter by 1
            failedDims ← NULL
        END IF
    UNTIL stepSize > ε
END PROCEDUR
```

شکل 3)

۳-۳- شناسایی و پاسخ به تغییر

گروه ذرات مورد استفاده در این الگوریتم پس از هر بار همگرایی نسبی مجددا مقدار دهی اولیه می‌شود، پس به شناسایی تغییر به منظور اعمال سازوکاری برای افزایش تنوع میان ذرات گروه نیاز نیست. علاو بر این به دلیل اینکه در هر بار مقدار دهی اولیه به گروه ذرات همه حافظه‌های ذرات و گروه نیز به فراموشی سپرده می‌شود، برای مقابله با مشکل حافظه‌ی منسوخ گروه ذرات نیز به شناسایی تغییر نیاز نداریم. تنها به دلیل اینکه جستجوی مستقیم ساده لوحانه تنها در جهت کاهش اندازه‌ی گام حرکت قادر به تنظیم خودکار هستند، هنگامی که از این جستجوگر استفاده می‌کنیم، شناسایی تغییر به منظور مقدار دهی اولیه به گام حرکت این جستجوگرها الزامی است.

در مقالات روش‌های متعددی برای شناسایی تغییر پیشنهاد شده است، در این کار ما با ارزیابی مجدد بهترین خاطره‌ی گروهی ذرات p_g ، تغییر را شناسایی می‌کنیم. در واقع در ابتدای هر تکرار شایستگی موقعیت p_g توسط تابع ی شایستگی ارزیابی می‌شود، وجود تغییر در

مقدار ارزیابی شده به منزله بروز تغییر در محیط در نظر گرفته می‌شود.

۴- آزمایش‌ها و نتایج

۴-۱- تولید کننده تابع محک و معیار کارایی

کارایی الگوریتم پیشنهادی توسط تولید کننده تابع محک قله‌های روان MPB مورد ارزیابی قرار می‌گیرد تابع محک قله‌های روان (MPB) که توسط برنکی در [16] معرفی شد، در واقع یک تولید کننده‌ی محیط‌های پویای مصنوعی است که با پذیرش عمومی محافل تحقیقاتی پیرامون بهینه‌سازی در محیط‌های پویا مواجه شده است و به عنوان معتبرترین و مرسوم‌ترین تولید کننده محیط‌های پویا با اقبال قابل ملاحظه‌ای مواجه بوده است. محیط‌های تولید شده توسط آن، گستره‌هایی چند بعدی و چند قله‌ای با قابلیت تنظیم تعداد بهینه‌های محلی است که ارتفاع، پهنا و مکان قله‌ها متغییر با زمان است. همچنین سرعت و شدت تغییرات محیط توسط پارامترهایی قابل تنظیم است. تنظیمات پیش فرض برای پارامترهای MPB مطابق با پیکربندی شناخته شده‌ای تحت عنوان سناریو II در جدول (1) آورده شده است.

جدول 1) پیکربندی پارامترهای MPB مبتنی بر سناریو II

نام پارامتر	مقدار
Number of dimensions (d)	5
Number of peaks (m)	10
Change frequency (f)	5000
Height severity (h_s)	0.7
Width severity (w_s)	0.1
Peak shape	Cone
Shift length (s)	1.0
A	[0, 100]
H	[30, 70]
W	[1, 12]
I	50

که در آن s مشخص کننده‌ی حد اکثر میزان جابجایی مکانی قله‌ها در هر تغییر است، m تعداد قله‌های محیط و T تناوب تغییرات محیط بر مبنای تعداد محاسبات شایستگی میان دو تغییر متوالی است. H و W به ترتیب مشخص کننده‌ی دامنه‌ی ارتفاع و پهنای قله‌ها است که با شدت h_s و w_s تغییر می‌کنند. I مشخص کننده ارتفاع اولیه قله‌ها است و A دامنه‌ی فضای جستجو در تمامی ابعاد را مشخص می‌کند.

یکی از روش‌های متداول و قدیمی برای مقایسه کارایی الگوریتم‌های ارائه شده برای یک محیط پویا بررسی بصری نمودارهای مربوط به بهترین و میانگین شایستگی یا خطای حاصل شده در طول تکرارها یا محاسبات شایستگی صورت پذیرفته است، اما معروفت‌ترین معیار کارایی کمی برای محیط‌های پویا معیار خطای آفلاین است. به صورت رسمی خطای آفلاین به صورت زیر تعریف می‌شود.

$$offline-error = \frac{1}{T}\sum_{t=1}^{T} minimumErrorAfterLastChang \quad (۳)$$

که در آن T تعداد کل محاسبات شایستگی است.

۴-۲- تنظیمات

برای تمامی آزمایش‌ها مقادیر پیش‌فرض برای پارامترهای الگوریتم پیشنهادی مطابق با مقادیر گزارش شده در جدول (2) انجام شده است. الگوریتم پیشنهادی ما با الگوریتم‌های mQSO [12]، cellular PSO [15]، TP_CPSO [11]، LSPO_SOFCMA [14] و CESO [13] مورد مقایسه قرار می‌گیرد. تنظیمات مورد استفاده برای الگوریتم‌های فوق مطابق با تنظیمات پیشنهاد شده در مقاله‌های ارجاع داده شده است.

جدول 2) تنظیمات پیش‌فرض الگوریتم PSOledLS

نام پارامتر	مقدار
Swarm size	5
Inertial weight (w)	0.7298
Cognitive AND Social Factors (c_1=c_2)	1.4961
initialStepSize	0.5
discountFactor	0.2
Minimum Radius of Swarm (δ)	7
Exclusive Radius (r)	35

۴-۳- مقایسه‌ی کارایی الگوریتم پیشنهادی با الگوریتم‌های دیگر در محیط‌های متفاوت

الگوریتم‌های TP_CPSO و LPSO-SOFCMA نیز مانند الگوریتم پیشنهاد شده در این مقاله از بهینه‌ساز گروه ذرات به عنوان جستجوگر عمومی و از جستجوگرهای محلی برای بهبود نتایج حاصل بهره می‌برند، به همین برای مقایسه با الگوریتم پیشنهادی ما انتخاب‌های مناسبی محسوب می‌شوند.

الگوریتم CESO یک الگوریتم همکارانه از بهینه‌ساز گروه ذرات و الگوریتم تکاملی تفاضلی ازدحامی است که در مقالات بسیاری مورد ارجاع قرار گرفته است و نسخه‌های گسترش یافته آن نیز در مقالات دیگری نظیر [17] ارائه شده است. با وجود این پیاده‌سازی‌های ما نتایج گزارش شده در مقاله را تایید نمی‌کنند. در واقع این الگوریتم دارای یک مشکل کاملاً آشکار است. آنگونه که در مقاله مذکور شرح داده شده است، هر گاه فاصله بهترین خاطره‌ی گروه ذرات و بهترین موجود حاضر در جمعیت الگورتیم تکاملی تفاضلی ازدحامی از 1 کمتر شود، اعضای جمعیت گروه ذرات توسط اعضای جمعیت تکاملی تفاضلی مقدار دهی می‌شوند، در واقع در این وضعیت گروه ذرات و جمعیت تکاملی کاملا بر یکدیگر منطبق می‌شوند. حال با توجه به اینکه در مقاله مذکور حد اکثر سرعت مجاز برای حرکت ذرات 0.1 در نظر گرفته شده است، مطمعنا در تکرارهای بعدی هرگز فاصله بهترین

حافظه گروه از بهترین موجود جمعیت تکاملی بیش از 1 نخواهد بود، در واقع پس از اینکه اولین بار این اتفاق رخ می‌دهد در تکرارهای بعدی لزوما این شرط برقرار خواهد بود، و به همین دلیل عملا ذرات گروه در ابتدای هر تکرار توسط جمعیت تکاملی مقدار دهی می‌شوند.

همه‌ی آزمایش‌ها در خلال 100 رخداد تغییر در محیط گزارش شده است، میانگین 100 اجرای مستقل هر یک از الگوریتم‌ها بعلاوه خطای استاندارد در محیط‌های پویای متفاوت در جداول (6-3) گزارش شده است. برای هر محیط، آزمون t با سطح معناداری 0.05 اعمال شده است و نتیجه بهترین الگوریتم به صورت برجسته متمایز شده است. هنگامی که خطای برون خط میان چند الگوریتم برتر دارای تفاوت معنی داری نیست همه‌ی آنها به صورت متمایز از سایرین نمایش داده شده اند.

در جداول 3 الی 6 نتایج حاصل الگوریتم‌های مورد مقایسه بر حسب خطای برون خط آورده شده است. نتایج حاصل نشان دهنده‌ی برتری مطلق الگوریتم پیشنهادی در محیط‌هایی با سرعت بالای تغییرات (تناوب تغییرات برابر با 500 و 1000 محاسبه‌ی شایستگی) است. در محیط‌هایی با تغییراتی آهسته‌تر نیز تنها الگوریتمی که توانایی قابل رقابتی را به نمایش می‌گذارد TP_CPSO است. که این الگوریتم نیز در محیط‌هایی با تعداد زیاد قله (تعداد قله‌ها برابر با 50) و محیط‌های تک قله‌ای حتی در محیط‌هایی با تغییرات آهسته‌تر نیز مغلوب الگوریتم PSOledLS می‌باشد. نمودار شکل (5) نیز نشان دهنده روند کاهش خطای برون خط و کمترین خطای حاصل پس از بروز تغییر در محیط برای سه الگوریتم PSOlesLS، TP_CPSO، CESO و LPSO_SOFCMA می‌باشد. همانطور که مشاهده می‌شود چه از منظر قدرت اکتشاف (کاهش سریع خطا در تکرارهای ابتدایی پس از بروز تغییر) و چه از نظر توانایی استخراج (ادامه روند کاهش خطا) الگوریتم پیشنهادی ما برتری محسوسی نسبت به دو الگوریتم ترکیبی رغیبش یعنی CESO و LPSO_SOFCMA داراست و تنها الگوریتم TP_CPSO کارایی قابل رقابتی را به نمایش می‌گذارد.

۵- نتیجه‌گیری

در این مقاله یک الگوریتم ترکیبی و همکارانه از بهینه ساز گروه ذرات و جستجوگرهای محلی استراتژی تکامل دو عضوی با قانون یک پنجم موفقیت، جستجوی الگوی HJ و یک جستجوگر محلی پیشنهادی تحت عنوان جستجوی مستقیم ساده لوحانه ارائه شده است. هدف از ترکیب این الگوریتم‌ها، ارائه‌ی الگوریتمی با توانایی اکتشاف سریع و جستجوی عمومی بهینه‌ساز گروه ذرات در کنار توانایی جستجوی محلی و استخراج دقیق جستجوگرهای محلی بیان شده بوده است. نتایج نشان دهنده‌ی کارایی بسیار قابل قبول الگوریتم پیشنهادی در مقایسه با چندین الگوریتم موفق در زمینه بهینه سازی در محیط‌های پویا است. در اکثر محیط‌های پویای مورد بررسی کارایی الگوریتم پیشنهادی بر مبنای معیار کارایی خطای برون خط برتری مطلقی را به نمایش می‌گذارد. به عنوان روال آتی تحقیقات می‌توان ارائه ساختاری مبتنی بر الگوریتم ممتیک برای الگوریتم بیان شده را پیشنهاد نمود. در یک الگوریتم ممتیک همزمان می‌توان از توانایی اکتشاف و استخراج جستجوگرها استفاده نمود و در حالی که در الگوریتم پیشنهاد شده تقریبا هیچ تبادل اطلاعاتی میان جستجوگرهای محلی استقرار یافته در فضای مسئله صورت نمی پذیرد.

جدول 3) خطای برون خط بعلاوه‌ی خطای استاندارد به ازای تعداد قله‌های متفاوت (T=500)

CESO (Our Implementation)	LPSO-SOFCMA	mQSO 10(5+5^q)	Cellular PSO	TP-CPSO	PSOledLS	M
6.62±0.39	9.71±0.30	36.52±3.2	11.62±0.77	5.96±0.22	**1.81±0.17**	1
16.93±0.32	14.02±0.23	11.18±0.4	8.78±0.28	5.56±0.11	**4.16±0.26**	10
17.25±0.31	12.77±0.15	10.37±0.2	8.24±0.22	5.59±0.08	**4.14±0.13**	30
17.45±0.34	12.05±0.14	10.33±0.2	8.37±0.20	5.57±0.08	**4.07±0.12**	50

جدول 4) خطای برون خط بعلاوه‌ی خطای استاندارد به ازای تعداد قله‌های متفاوت (T=1000)

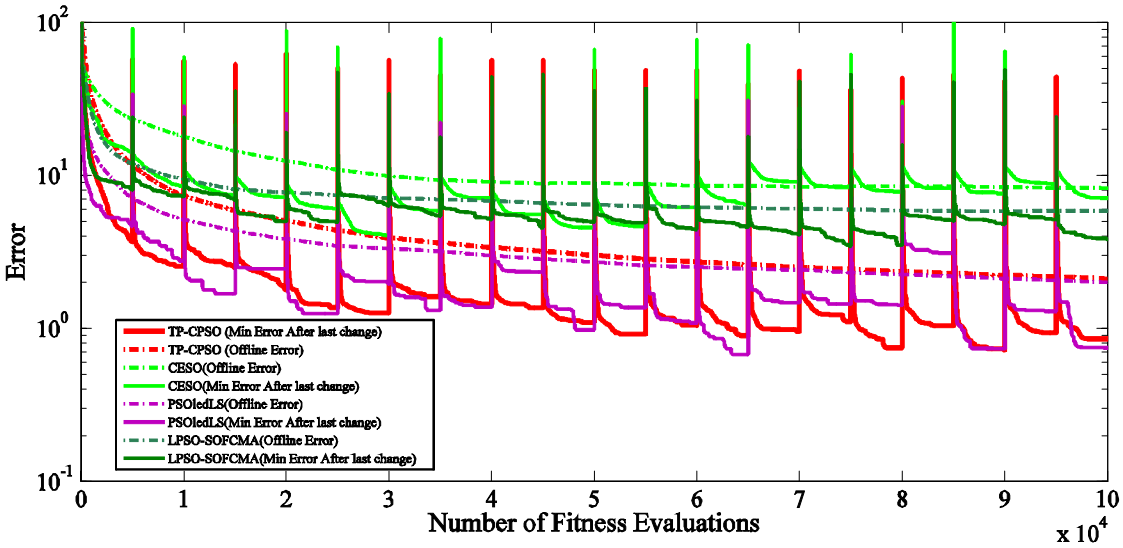
CESO (Our Implementation)	LPSO-SOFCMA	mQSO 10(5+5^q)	Cellular PSO	TP-CPSO	PSOledLS	M
3.78±0.15	5.75±0.16	19.1±1.5	5.86±0.42	2.66±0.08	**0.79±0.13**	1
14.16±0.40	9.98±0.20	6.33±0.23	5.75±0.23	3.17±0.06	**2.55±0.04**	10
13.54±0.30	9.05±0.11	6.51±0.16	5.84±0.16	3.45±0.06	**2.90±0.05**	30
13.68±0.28	8.56±0.10	6.67±0.16	5.84±0.14	3.57±0.05	**2.82±0.04**	50

جدول 5) خطای برون خط بعلاوه‌ی خطای استاندارد به ازای تعداد قله‌های متفاوت (T=2500)

CESO (Our Implementation)	LPSO-SOFCMA	mQSO 10(5+5^q)	Cellular PSO	TP-CPSO	PSOledLS	M
1.70±0.05	3.80±0.12	7.79±0.72	3.78±0.25	0.92±0.03	**0.39±0.04**	1
11.78±0.41	8.15±0.21	3.20±0.14	3.18±0.16	**1.59±0.06**	**1.65±0.06**	10
11.40±0.35	6.72±0.11	4.03±0.12	3.90±0.11	**1.99±0.04**	**1.91±0.07**	30
11.62±0.38	6.14±0.07	3.95±0.10	4.08±0.11	2.01±0.03	**1.75±0.08**	50

جدول 6) خطای برون خط بعلاوه‌ی خطای استاندارد به ازای تعداد قله‌های متفاوت (T=5000)

CESO (Our Implementation)	CESO ([13])	LPSO-SOFCMA	mQSO 10(5+5[q])	Cellular PSO	TP-CPSO	PSOledLS	M
0.99±0.06	1.04±0.00	3.01±0.09	4.06±0.40	2.79±0.18	0.40±0.01	**0.18±0.02**	1
10.66±0.42	1.38±0.02	7.09±0.14	1.93±0.09	2.06±0.12	**0.96±0.05**	1.12±0.03	10
10.18±0.31	1.24±0.02	5.81±0.09	2.84±0.08	3.21±0.11	**1.32±0.03**	**1.33±0.06**	30
10.17±0.35	1.45±0.01	5.23±0.07	2.74±0.07	3.37±0.12	1.40±0.03	**1.30±0.03**	50



شکل 4) نمودارهای خطای برون خط و کمترین خطای حاصل پس از بروز آخرین تغییر در سناریو II (با 50 قله)

مراجع

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in , *IEEE International Conference on Neural Networks, 1995. Proceedings*, 1995, vol. 4, pp. 1942 –1948 vol.4.

[2] I. Rechenberg, "Evolutionsstrategie–Optimierung technisher Systeme nach Prinzipien der biologischen Evolution," 1973.

[3] I. Rechenberg, "Evolution Strategy: Optimization of Technical systems by means of biological evolution," *Fromman-Holzboog, Stuttgart*, vol. 104, 1973.

[4] H. P. Schwefel, "Evolutionsstrategie und numerische Optimierung," Technische Universität Berlin, 1975.

[5] R. Hook and T. Jeeves, "Direct search solutions of numerical and statistical problems," *Journal of the Association for Computing Machinery*, vol. 8, pp. 212–229, 1961.

[6] J. Branke, *Evolutionary optimization in dynamic environments*, vol. 142. kluwer academic publishers Norwell, MA, 2002.

[7] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *Evolutionary Computation, IEEE Transactions on*, vol. 9, no. 3, pp. 303–317, 2005.

[8] T. Blackwell, "Particle Swarm Optimization in Dynamic Environments," in *Evolutionary Computation in Dynamic and Uncertain Environments*, vol. 51, S. Yang, Y.-S. Ong, and Y. Jin, Eds. Springer Berlin / Heidelberg, 2007, pp. 29–49.

[9] C. Cruz, J. González, and D. Pelta, "Optimization in dynamic environments: a survey on problems, methods and measures," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 15, no. 7, pp. 1427–1448, 2011.

[10] T. T. Nguyena, S. Yangb, and J. Brankec, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, 2012.

[11] A. Sharifi, V. Noroozi, M. Bashiri, A. B. Hashemi, and M. R. Meybodi, "Two phased cellular PSO: A new collaborative cellular algorithm for optimization in dynamic environments," in *2012 IEEE Congress on Evolutionary Computation (CEC)*, 2012, pp. 1 –8.

[12] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 4, pp. 459–472, 2006.

[13] R. I. Lung and D. Dumitrescu, "A collaborative model for tracking optima in dynamic environments," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, 2007, pp. 564–567.

[14] H. Wang, S. Yang, W. Ip, and D. Wang, "A particle swarm optimization based memetic algorithm for dynamic optimization problems," *Natural Computing*, vol. 9, no. 3, pp. 703–725, 2010.

[15] A. Hashemi and M. Meybodi, "Cellular PSO: A PSO for dynamic environments," in *Advances in Computation and Intelligence*, 2009, pp. 422–433.

[16] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99*, 1999, vol. 3.

[17] R. Lung and D. Dumitrescu, "Evolutionary swarm cooperative optimization in dynamic environments," *Natural Computing*, vol. 9, no. 1, pp. 83–94, 2010.