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 در مسائل واقعی تغییر پذیر با زمان بودن این گونه مسائل یا همان پویایی است. الگوریتم‌های بهینه‌سازی در مواجهه با محیط‌های پویا علاوه بر یافتن بهینه یا بهینه‌های محیط مکلف به تعقیب تنگاتنگ جابجایی بهینه و یا بهینه‌های محیط و همچنین یافتن بهینه‌های جدید ایجاد شده در محیط به دلیل تغییرات محیط هستند. الگوریتم بهینه‌ساز گروه ذرات یکی از روش‌های پذیرفته شده در هوش گروهی است که کارایی مناسب آن در محیط‌های ایستا و پویا پذیرفته شده است. یک چالش پیش روی دیگر در الگوریتم‌هایی دارای حافظه مانند الگوریتم بهینه‌ساز گروه ذرات در محیط‌های پویا، منسوخ شدن حافظه به دلیل بروز تغییر در محیط است، به همین دلیل باید مکانیسمی برای ارزیابی مجدد و یا پاک کردن حافظه‌های منسوخ الگوریتم نیز در نظر گرفته شود. الگوریتم‌های مبتنی بر جمعیت مانند الگوریتم‌های تکاملی و یا الگوریتم بهینه‌ساز گروه ذرات دارای مزیت اجتناب از همگرایی زودرس و توانایی گذر از بهینه‌های محلی هستند، اما در سوی دیگر این الگوریتم‌ها قالبا از توانایی استخراج مناسبی برخوردار نیستند. الگوریتم‌های جستجوی محلی اما در مقایسه با الگوریتم‌های مبتنی بر جمعیت از توانایی اکتشاف کمتری برخوردار هستند ولی قالبا دارای توانایی استخراج مناسبی هستند. در این مقاله ما یک الگوریتم ترکیبی همکارانه از الگوریتم بهینه‌ساز گروه ذرات و جستجوگرهای محلی ارائه می‌شود. جستجوگرهای محلی مورد استفاده الگوریتم پیشنهادی ما شامل استراتژی تکامل (1+1)، جستجوی الگوی هوک و جیوز و یک جستجوگر محلی پیشنهادی می باشد. الگوریتم همکارانه پیشنهاد شده از توانایی اکتشاف بهینه ساز گروه ذرات برای کشف بهینه‌های محیط و از جستجوگرهای محلی برای حفظ و استخراج بهینه‌های کشف شده استفاده می‌کند. نتایج گزارش شده 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، محیط‌های پویا، تولید کننده تابع محک قله‌های روان

1- مقدمه

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 تجهیز 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.

تابع محک قله‌های روان (MPB) که توسط برنکی در [1] 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. محیط‌های تولید شده توسط MPB شامل یک گستره با قابلیت تنظیم تعداد بهینه‌های محلی یا همان قله‌های محیط است که ارتفاع، پهنا و موقعیت هر قله متغیر با زمان است. همچنین سرعت و شدت تغییرات محیط توسط پارامترهایی قابل تنظیم است. 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 [2]، [3]، [4]، [5]، [6] و [7] مراجعه شود.

بهینه‌سازی گروه ذرات [8]، به عنوان یکی از موفق‌ترین الگوریتم‌های مبتنی بر جمعیت و هوش گروهی که از شبیه‌سازی رفتار اجتماعی گروهی ماهی‌ها و پرندگان 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 [5]، [9] مراجعه شود.

در الگوریتم پیشنهادی ما در این مقاله جستجوی عمومی و اکتشاف بهینه‌های نوید بخش محیط به عهده‌ی الگوریتم بهینه‌ساز گروه ذرات است، در حالی که حفظ و استخراج قله‌ها به الگوریتم‌های جستجوی محلی واگذار می‌شود. الگوریتم جستجوی محلی استراتژی تکامل (1+1) به منظور حفظ، تعقیب و استخراج بهینه‌های محلی کشف شده و از جستجوی الگوی هوک و جیوز به منظور حفظ و استخراج موثرتر نویدبخش‌ترین بهینه‌ی یافت شده‌ی کنونی محیط استفاده می‌شود.

الگوریتم جستجوی الگوی هوک و جیوز یک الگوریتم جستجوی محلی بسیار ساده و کارآمد است که بیش از 40 سال پیش پیشنهاد شده است و همچنان یکی از قابل اطمینان‌ترین الگوریتم‌های جستجوی محلی محسوب می‌شود [10].

در واقع در الگوریتم پیشنهادی ما سه نوع عامل بهینه‌ساز به صورت همکارانه فرایند بهینه‌سازی را به پیش می‌برند. عامل‌هایی تحت عنوان ذرات گروه در الگوریتم بهینه‌ساز گروه ذرات که یک گروه اکتشافی را برای کشف بهینه‌های محلی نوید بخش محیط و همچنین قله‌های بهبود یافته به دلیل بروز تغییرات در محیط را به عهده خواهند داشت. دومین گروه عامل‌ها، عامل‌های استراتژی تکاملی (1+1) است که از قانون یک پنجم موفقیت برای تطبیق گام جهش استفاده می‌کنند، وظیفه‌ی عامل‌های این گروه حفظ، استخراج و تعقیب بهینه‌های کشف شده توسط الگوریتم بهینه‌ساز گروه ذرات است. نوع سوم عامل‌های موجود در محیط، عامل جستجوی الگوی هوک و جیوز است که موظف به استخراج بهترین قله‌ی کشف شده در محیط پس از بروز آخرین تغییر در محیط است.

۲- کارهای مرتبط

۲-۱- اصول بهینه‌ساز گروه ذرات

مشابه با الگوریتم‌های تکاملی، بهینه‌سازی گروه ذرات نیز یک الگوریتم مبتنی بر جمعیت و تکراری است. تفاوت اصلی 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. قوانینی که این حرکت را هدایت می‌کنند، از روابط اجتماعی و گروهی میان ماهی‌ها و پرندگان در طبیعت ملهم شده‌است. در یک مدل بهینه‌سازی گروه ذرات، یک ذره توسط مکان و سرعتش بازنمایی می‌شود. در هر تکرار، هر ذره در گروه می‌تواند مبتنی بر سرعت کنونی خود، مکانش، بهترین مکان مشاهد شده توسط خودش و بهترین مکان یافت شده توسط همسایگانش، خود را بهنگام کند. 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.

۲-۲- جستجوگرهای محلی

۲-۲-۱- 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 [11]، [12]، [13]. 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 که از قانون یک پنجم موفقیت برای تنظیم گام جهش استفاده می‌کند استفاده می‌کنیم.

برای تنظیم گام جهش توسط قانون یک پنجم موفقیت از یک ایده‌ی بسیار روشن و ساده الهام گرفته شده است: موفقیت‌های پی در پی نشان دهنده‌ی فاصله‌ی زیاد از بهینه‌ی محلی است و بنا براین باید با اندازه‌ی گام جهش و به تبع آن سرعت حرکت را افزایش داد و در سوی دیگر شکست‌های متوالی در چندین تولید نسل نشان دهنده‌ی نزدیکی بیش از حد به بهینه‌ی محلی است پس باید گام جهش را کاهش داد تا از نوسان در اطراف بهینه اجتناب شود. ریچنبرگ در رساله‌ی دکتری خود به صورت نظری و ریاضی مرزی دقیق را برای میزان موفقیت زیاد و کم مشخص نمود. او اثبات می‌کند هنگامی که احتمال موفقیت در k تکرار گذشته برابر با یک پنجم است بیشترین سرعت همگرایی به سمت بهینه‌ی محلی حاصل می‌شود و هنگامی که این احتمال بیش از یک پنجم باشد نشان دهنده‌ی موفقیت‌های زیاد و در نتیجه فاصله بیش از حد از بهینه‌ی محلی است پس باید اندازه‌ی گام جهش افزایش یابد و هنگامی که احتمال موفقیت در تکرارهای گذشته کمتر از یک پنجم است مشخص کننده‌ی کم بودن فاصله از بهینه‌ی محلی است در نتیجه باید گام جهش کاهش یابد.

شبه‌کد: تطبیق گام جهش توسط قانون یک پنجم موفقیت

```
PROCEDURE adaptMutation_OneFifthRuleSuccess
BEGIN
IF CurrentGenerationIsSuccessful = True THEN
   P_s^New = ((k-1)/k) P_s^Old
ELSE
   P_s^New = ((k-1)/k) P_s^Old + 1/k
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)

شبه کد مربوط به یک تولید نسل استراتژی تکامل (1+1) با تطبیق گام جهش توسط قانون یک پنجم موفقیت در شکل (1) نمایش داده شده است.

نکته قابل توجه اینکه حتی اگر در تکرارهای انتهایی پیش از بروز تغییر در محیط گام جهش به شدت کاهش یافته باشد، سازوکار تطبیقی جهش باعث خواهد شد که پس از تغییر در محیط و به دلیل افزایش نرخ تولید نسل‌های موفق، مجددا گام جهش افزایش یابد و پس از تعقیب بهینه‌ی جابجا شده و نزدیک شدن به مکان جدید آن گام جهش به صورت خودکار کاهش یابد. این ویژگی به ویژه در مواقعی که شناسایی لحظه بروز تغییر به دلایلی ممکن نباشد برای مثال در محیط‌های به شدت نویزی که شناسایی بروز تغییر در محیط را دشوار می‌سازد، بسیار حائز اهمیت خواهد بود، چرا که دیگر نیاز نیست پس از بروز تغییر اندازه‌ی گام جهش مقدار دهی اولیه شود در نتیجه می‌توان از شناسایی لحظه‌ی بروز تغییر در محیط صرف نظر کرد.

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 > Parrent.fitness THEN
      Parrent ← Child
      CurrentGenerationIsSuccessful ← True
   ELSE
      CurrentGenerationIsSuccessful ← False
   END IF
CALL adaptMutation_OneFifthRuleSuccess
END PROCEDURE
```

شکل 2)

۲-۲-۲- جستجوی الگوی هوک و جیوز

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.

با وجود اینکه جستجوی الگوی هوک و جیوز بیش از 40 سال پیش پیشنهاد شده است [10]، اما همچنان در زمره اولین انتخاب‌های محققین به عنوان یک جستجوگر محلی قطعی محسوب می‌شود. 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. پس از اعمال یک دور حرکات اکتشافی از این نقطه‌ی پایه اگر بهبودی نسبت به نقطه‌ی پایه اصلی X 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. شبه‌کد مربوط به الگوریتم جستجوی الگوی HJ در شکل (3) قرار داده شده است.

۲-۲-۳- 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 تکرار قبل یا 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 یا به عبارت 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.

شبه‌کد: جستجوی الگوی هوک و جیوز (HJ_PatternSearch)

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

شکل 3)

۳- الگوریتم پیشنهادی

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 دو جستجوگر محلی بسیار قدرتمند جستجوی الگوی HJ و جستجوی مستقیم ساده لوحانه 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.

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 4)

۳-۱- جستجوی عمومی و مکان‌یابی برای استخراج

در الگوریتم پیشنهادی ما از یک گروه ذرات برای اکتشاف قله‌ها 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 تا وظیفه‌ی نگهبانی و استخراج بیشتر قله‌ی یافت شده را عهده دار شود. 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 جستجوهای دقیق‌تر در اطراف بهینه‌ی یافت شده و نگهداری از بهینه توسط استراتژی تکامل (1+1) با تطبیق گام جهش توسط قانون یک پنجم موفقیت صورت می‌پذیرد. برای استخراج دقیق‌تر بهترین بهینه‌ی یافت شده پس از آخرین تغییر در محیط از جستجوی الگوی HJ و یا جستجوی مستقیم ساده لوحانه NDS استفاده می‌شود. در جستجوی الگوی HJ و جستجوی مستقیم ساده لوحانه 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 دو جستجوگر محلی دیگر حتما باید پس از بروز تغییر در محیط و شناسایی تغییر مقدار گام جهش مقدار دهی اولیه شود.

۳-۳- شناسایی و پاسخ به تغییر

الگوریتم‌های مبتنی بر جمعیت و تکاملی در رویارویی با محیط‌های پویا با چالش از دست دادن تنوع جمعیت و در نتیجه ناتوانی در تعقیب مسیر حرکت بهینه‌های جابجا شده مواجه هستند. هر چند در الگوریتم پیشنهادی ما نیز از الگوریتم بهینه‌ساز گروه ذرات استفاده می‌شود که یک الگوریتم مبتنی بر جمعیت است اما این الگوریتم با مشکل از دست دادن تنوع به صورت جدی مواجه نیست. 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.

الگوریتم‌های دارای حافظه نیز در مواجهه با محیط‌های پویا با یک مشکل اساسی که همانا حافظه‌های منسوخ شده پس از بروز تغییر در محیط است رو به رو هستند. اگر چه در الگوریتم بهینه‌ساز گروه ذرات مورد استفاده‌ی ما نیز همه ذرات دارای حافظه‌هایی هستند که پس از بروز تغییر در محیط فاقد اعتبار خواهد بود اما باز هم به دلیل اینکه پس از هر بار همگرایی نسبی کل گروه مجددا مقدار دهی اولیه می‌شود (که شامل فراموشی کامل حافظه‌ی قبلی نیز می‌شود)، به شناسایی تغییر با هدف ارزیابی مجدد یا فراموشی حافظه‌های منسوخ شده نیز نیازی نمی‌باشد.

جستجوگرهای محلی HJ و جستجوی مستقیم ساده لوحانه تنها در جهت کاهش اندازه‌ی گام حرکت قادر به تنظیم خودکار هستند به همین دلیل هنگامی که از این جستجوگرها 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 قله‌های روان (MPB)

کارایی الگوریتم پیشنهادی توسط تولید کننده تابع محک قله‌های روان MPB مورد ارزیابی قرار می‌گیرد. در مقالات مرتبط با بهینه‌سازی در محیط‌های پویا MPB 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 هستند که در آنها ارتفاع، پهنا و مکان قله‌ها در زمان بروز تغییر در محیط تغییر می‌کند.

مگر در مواقعی که خلاف آن ابراز شود، مقادیر پارامترهای MPB مطابق با پیکربندی شناخته شده‌ای تحت عنوان سناریو II صورت پذیرفته است (جدول 1).

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.

۴-۲- معیار کارایی

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 معروف‌ترین 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.

اکثر کاربردهای واقعی بهینه‌سازی در محیط‌های پویا مسائلی هستند که نیاز به عملکرد آنلاین و زمان حقیقی دارند. برای مثال در مسئله قیمت گذاری پویا، هر محاسبه‌ی شایستگی تنظیم یک قیمت و محاسبه‌ی سود حاصل از فروش با قیمت تنظیم شده است، در چنین حالتی هر محاسبه‌ی شایستگی مهم است نه تنها بهترین قیمت تنظیم شده پس از بروز آخرین تغییر در بازار. معیارکارایی که برای این ارزیابی عملکرد الگوریتم‌ها در چنین شرایطی مناسب‌تر خواهد بود معیار خطای برخط است که به صورت زیر تعریف می‌شود.

$$online-error=\frac{1}{T}\sum_{t=1}^{T} currentError \quad (۴)$$

از آنجایی که در اکثریت قریب به اتفاق مقالات مرتبط با بهینه‌سازی در محیط‌های پویا نتایج تنها بر مبنی معیار کارایی برون خط گزارش شده است ما نیز به ناچار و به هدف مقایسه‌ی کارایی الگوریتم پیشنهادی خود با سایر الگوریتم‌های ارائه شده، نتایج خود را بر مبنی همین معیار کارایی گزارش می‌کنیم.

۴-۲-۱- تنظیمات

برای تمامی آزمایش‌ها مقادیر پیش‌فرض برای پارامترهای الگوریتم پیشنهادی مطابق با مقادیر گزارش شده در جدول (2) انجام شده است.

۴-۲-۲- مقایسه‌ی کارایی الگوریتم پیشنهادی با الگوریتم‌های دیگر در محیط‌های متفاوت

نتایج الگوریتم پیشنهادی با الگوریتم‌های cellular PSO [14]، cellularDE [15]، adaptive mQSO و TP_CPSO که همگی دارای بهترین نتایج گزارش شده در محیط‌های تولید شده توسط تابع محک MPB هستند مورد مقایسه قرار می‌گیرد. بر اساس دانسته‌ها و جستجوهای ما نتایج حاصل از TP_CPSO بهترین نتایج حاصل از الگوریتم‌های مبتنی بر بهینه‌ساز گروه ذرات برای محیط‌های پویا است. تنظیمات مربوط به همه‌ی الگوریتم‌های مزبور مطابق با تنظیمات گزارش شده در مقاله‌ی ارجاع داده شده برای هر یک می‌باشد. همه‌ی آزمایش‌ها در خلال 100 رخداد تغییر در محیط گزارش شده است، دقت شود از آنجا که هر تغییر پس از T محاسبه شایستگی رخ می‌دهد، شرایط مقایسه از نظر زمانی کاملا عادلانه می‌باشد. نتایج حاصل از 100 اجرای هر یک از الگوریتم‌ها با فاصله اطمینان 95% در محیط‌های پویای متفاوت در جداول (3-6) 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.

جدول 2) تنظیمات پیش‌فرض الگوریتم پیشنهادی

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

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 پیشنهادی ما است.

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.

برای کارهای آینده می‌توان ارائه ساختاری مبتنی بر الگوریتم ممتیک برای الگوریتم بیان شده را پیشنهاد نمود. در یک ساختار مبتنی بر 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 به ازای تعداد قله‌های متفاوت (T=500)

Adaptive mQSO	Proposed Algorithm	CellularDE	Cellular PSO	Cellular MuPSO	TP-CPSO	m
5.08±0.27	**1.81**	8.20±0.19	11.62±0.77	8.42±0.49	5.96±0.22	1
6.20±0.11	**4.16**	5.93±0.04	8.78±0.28	6.39±0.22	5.56±0.11	10
7.23±0.16	**4.14**	5.56±0.03	8.24±0.22	6.58±0.17	5.59±0.08	30
7.49±0.09	**4.07**	5.47±0.02	8.37±0.20	6.60±0.17	5.57±0.08	50

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

Adaptive mQSO	Proposed Algorithm	CellularDE	Cellular PSO	Cellular MuPSO	TP-CPSO	m
2.68±0.14	**0.79±0.13**	4.98±0.35	5.86±0.42	5.10±0.27	2.66±0.08	c1
4.11±0.08	**2.55±0.04**	3.98±0.03	5.75±0.23	4.34±0.17	3.17±0.06	10
4.98±0.10	**2.90±0.05**	4.77±0.02	5.84±0.16	4.81±0.13	3.45±0.06	30
5.12±0.05	**2.82±0.04**	4.87±0.02	5.84±0.14	4.92±0.11	3.57±0.05	50

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

Adaptive mQSO	Proposed Algorithm	CellularDE	Cellular PSO	Cellular MuPSO	TP-CPSO	m
1.09±0.06	**0.39±0.04**	2.38±0.78	3.78±0.25	2.32±0.13	0.92±0.03	1
2.33±0.11	**1.65±0.06**	2.42±0.02	3.18±0.16	2.41±0.12	**1.59±0.06**	10
3.13±0.09	**1.91±0.07**	3.29±0.03	3.90±0.11	3.44±0.11	**1.99±0.04**	30
3.24±0.07	**1.75±0.08**	3.44±0.02	4.08±0.11	3.59±0.10	2.01±0.03	50

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

Adaptive mQSO	Proposed Algorithm	CellularDE	Cellular PSO	Cellular MuPSO	TP-CPSO	m
0.55±0.02	**0.18±0.02**	1.53±0.07	2.79±0.18	1.27±0.07	0.40±0.01	1
1.43±0.04	1.12±0.03	1.64±0.03	2.06±0.12	1.67±0.10	**0.96±0.05**	10
2.15±0.05	**1.33±0.06**	2.62±0.05	3.21±0.11	2.72±0.10	**1.32±0.03**	30
2.28±0.02	**1.30±0.03**	2.75±0.05	3.37±0.12	3.15±0.11	1.40±0.03	50

مراجع

[1] 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.

[2] Krzysztof Trojanowski and Z. Michalewicz, "Evolutionary Algorithms for Non-Stationary Environments," CiteSeerX, 1999.

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

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

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

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

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

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

[15] V. Noroozi, A. Hashemi, and M. Meybodi, "CellularDE: A Cellular Based Differential Evolution for Dynamic Optimization Problems," in *Adaptive and Natural Computing Algorithms*, 2011, vol. 6593, pp. 340–349.