الگوریتم ژنتیک آشوب‌گونه مبتنی بر حافظه و خوشه‌بندی برای حل مسائل بهینه‌سازی پویا

مجید محمدپور [۱]، دانشجوی کارشناسی ارشد، حمید پروین [۲]، استادیار
۱- دانشکده فنی و مهندسی- دانشگاه آزاد اسلامی واحد یاسوج -یاسوج- ایران – mohammadpour@iauyasooj.ac.ir
۲- دانشکده مهندسی کامپیوتر، دانشگاه آزاد اسلامی، واحد ممسنی، ممسنی، ایران-parvin@iust.ac.ir
۲- باشگاه پژوهشگران جوان و نخبگان، واحد نورآباد ممسنی، دانشگاه آزاد اسلامی، نورآباد ممسنی، ایران

چکیده: اکثر مسائل موجود در دنیای واقعی یک مسئله بهینه‌سازی با ماهیتی پویا هستند، به‌طوری‌که مقدار بهینه سراسری آن‌ها در طول زمان ممکن است تغییر کند، بنابراین برای حل این مسائل الگوریتم‌هایی نیاز داریم که بتوانند خود را با شرایط این مسائل به‌خوبی سازگار نموده و بهینه جدید را برای این مسائل ردیابی نمایند. در این مقاله، یک الگوریتم ژنتیک آشوب‌گونه مبتنی بر خوشه‌بندی و حافظه برای حل مسائل پویا ارائه شده است. یک سیستم آشوب‌گونه پیش‌بینی دقیق‌تری از آینده نسبت به یک سیستم تصادفی دارد و میزان همگرایی را در الگوریتم افزایش می‌دهد. به‌طور معمول استفاده از اطلاعات گذشته اجازه می‌دهد الگوریتم به‌سرعت بعد از تغییر محیط به سازگاری در شرایط محیطی جدید برسد، بنابراین ایده موردنظر در این زمینه، استفاده از یک حافظه است که با استراتژی مناسبی اطلاعات مفید گذشته را ذخیره نموده و برای استفاده مجدد آن‌ها را بازیابی می‌نماید. خوشه‌بندی در حافظه و جمعیت اصلی، تنوع را در حین اجرای الگوریتم با تبادل اطلاعات میان خوشه‌های متناظر (خوشه‌ها با برچسب شبیه به‌هم) در حافظه و جمعیت اصلی حفظ می‌نماید. به‌طورکلی در این روش پیشنهادی دو جنبه نوآوری اساسی پیشنهاد شده است. یکی روش خوشه‌بندی استفاده‌شده که هم جمعیت اصلی و هم جمعیت حافظه را خوشه‌بندی (خوشه‌بندی مبتنی بر میانگین) می‌کند و دیگری راهکار مناسبی است که برای به‌روزرسانی حافظه استفاده شده است. برای آزمایش کارایی روش پیشنهادی از مسئله محک قله‌های متحرک استفاده شده که رفتاری شبیه به مسائل پویا در دنیای واقعی را شبیه‌سازی می‌کند. نتایج آزمایش‌ها کارایی مناسب روش پیشنهادی را در حل مسائل بهینه‌سازی پویا در مقایسه با دیگر روش‌ها نشان می‌دهد.

واژه‌های کلیدی: بهینه‌سازی پویا، الگوریتم ژنتیک، حافظه صریح، آشوب، خوشه‌بندی.

Chaotic genetic algorithm based on clustering and memory for solving dynamic optimization problems

M. Mohammadpour[1], student, H. Parvin[2], Assistant professor
1- Faculty of technical and Engineering, Azad University of Yasouj, Yasouj, Iran,
2- Faculty of computer Engineering, Azad University of Mamasani, Mamasani, Iran
Email: 1-mohammadpour@iauyasooj.ac.ir, 2- parvin@iust.ac.ir

Abstract: Most of the problems in the real world are of dyanamic optimization ones. It means that their optima may change over time, thus algorithms to solve these problems must be well adapt to their conditions in such a way that they should be able to track the optima during evolution. This article proposes a chaotic genetic algorithm based on clustering and memory for solving dynamic optimization problems. A chaotic system has more precise prediction of the future in comparison with random system and increases the speed of convergounce in the algorithm. The utilization of some information from the past allows quickly adapting right after a change. Thus the underlining idea of the paper is the use of memory in this field, which is a good strategy to store the useful information and retrieves them for reuse in the future. The clustering method maintains diversity in the memory and the population during running of the algorithm by exchanging information between corresponding clusters (clusters with similar tag) of the memory and the population. This algorithm uses a k-means clustering method to maintain diversity and improve local search. In this paper used tow the main innovation: clustering method and memory update. To test the effectiveness of the proposed method we choose the Moving Peaks Benchmark that has similar behaviors to real world dynamic proplems. The experimental results show the efficiency of the proposed algorithm for solving optimization problems in comparison with other methods.

Keywords: Dynamic optimization, genetic algorithm, explicit memory, chaos, clustering.

تاریخ ارسال مقاله: ۱۳۹۴/۴/۲۰
تاریخ اصلاح مقاله: ۱۳۹۴/۰۵/۲۶
تاریخ پذیرش مقاله: ۱۳۹۴/۷/۲۰
نام نویسنده مسئول: حمید پروین
نشانی نویسنده مسئول: ایران - یاسوج – کیلومتر ۴ جاده اصفهان - دانشگاه آزاد اسلامی واحد علوم تحقیقات یاسوج- دانشکده فنی و مهندسی

۱- مقدمه

امروزه در مسائل مهندسی، مسائلی وجود دارند که باید بهینه شوند، در بعضی از این مسائل هدف بهینه‌سازی کم کردن هزینه است (کمینه‌سازی) و در بعضی دیگر هدف حداکثر نمودن سود (بیشینه‌سازی) است. بهینه‌سازی بیان می‌کند، که باید در بین پارامترهای یک تابع به دنبال مقادیری بود که بتوانند تابع هدف را کمینه یا بیشینه نمایند. کلیه مقادیر مناسب جهت این امر، راه‌حل‌های ممکن و بهترین راه‌حل از این راه‌حل‌های ممکن ، راه‌حل بهینه نامیده می‌شوند. در بهینه‌سازی مسائل پیچیده، استفاده از روش‌های بهینه‌سازی دقیق غیرممکن است، بنابراین از روش‌های جستجوی تصادفی برای رسیدن به یک پاسخ نزدیک به بهینه استفاده می‌شود. درواقع در این نوع از الگوریتم‌ها پاسخ‌های مناسب در یک بازه زمانی قابل‌قبول به دست خواهند آمد، اما هیچ تضمینی جهت به دست آوردن بهترین پاسخ وجود ندارد. در میان روش‌های جستجوی تصادفی الگوریتم‌های تکاملی که برگرفته از طبیعت هستند از جایگاه ویژه‌ای برخوردار هستند. با توجه به پیچیدگی بسیاری از مسائل بهینه‌سازی توسعه‌هایی برای بهبود کارایی این الگوریتم‌ها در حل مسائل مختلف انجام‌گرفته است. به‌طورکلی مسائل بهینه‌سازی به دو دسته ایستا و پویا تقسیم می‌شوند. اگر مسئله بهینه‌سازی تابعی از زمان نبوده و بهینه برای این مسائل ثابت باشد در این صورت مسئله ماهیتی ایستا دارد. هرگاه تغییری در مقادیر تابع هدف اعم از محدودیت‌های تابع هدف و یا متغیرهای تابع هدف رخ دهد ممکن است، بهینه آن تابع نیز تغییر یابد و به‌عبارتی‌دیگر محیط تغییر نموده و مسئله دارای ماهیتی پویا است. چنین شرایطی در دنیای واقعی برای یک مسئله بهینه‌سازی نیز وجود دارد.

ردیابی بهینه سراسری مسائلی که در طول زمان تغییر نمی‌کنند و ایستا می‌باشند تا حدودی کار آسانی است، اما اگر تابع هدف در طول زمان دچار تغییر شود مسئله بهینه‌سازی ماهیتی پویا داشته و یافتن بهینه موردنظر کار مشکلی است. در این مسائل با ماهیت پویا با تغییر در تابع هدف ممکن است بهینه موردنظر نیز تغییر یابد، در این صورت یک الگوریتم باید بتواند به نحو مطلوبی این بهینه در حال جابجایی را ردیابی نموده و جمعیت را به سمت آن سوق دهد. الگوریتم‌های تکاملی ممکن است در محیط‌های ایستا کارایی مناسبی را از خود نشان دهند ولی این الگوریتم‌ها به‌تنهایی برای حل مسائل بهینه‌سازی پویا نمی‌توانند کارایی مناسبی داشته باشند، بنابراین برای حل مسائل بهینه‌سازی پویا نیاز به الگوریتم‌های بهینه‌سازی کارا و مناسب است. در یک مسئله بهینه‌سازی پویا هدف بهینه‌سازی صرفاً یافتن پاسخ بهینه نیست بلکه هدف ردیابی تمامی پاسخ‌های مسئله است.

در مسائل پویا، چالش‌های اساسی و مهمی پیش روی یک الگوریتم بهینه‌سازی است. برخی از این چالش‌ها می‌توانند شامل مشکل به‌روز کردن حافظه افراد جمعیت بعد از هر تغییر در محیط، حفظ تنوع[1] در محیط بعد از همگرایی افراد به بهینه موردنظر، ظرفیت محدود حافظه، شناسایی زمان تغییرات در محیط باشند. در این مقاله برای غلبه بر این چالش‌ها یک الگوریتم کارا و مناسب پیشنهاد شده است. الگوریتم پیشنهادی ترکیب مناسب الگوریتم ژنتیک استاندارد آشوب‌گونه با حافظه و خوشه‌بندی است. اکثر رفتارهای موجود در طبیعت رفتارهای آشوب‌گونه‌ای هستند. ازجمله این رفتارها عبارت‌اند از: رفتار حرکتی بال پروانه، امواج دریا، گردباد و غیره. یک سیستم آشوب‌گونه نسبت به یک سیستم تصادفی می‌تواند تخمین دقیق‌تری از آینده داشته باشد. بنابراین برای افزایش سرعت همگرایی در این روش به‌جای تصادفی‌سازی از تئوری آشوب برای تولید جمعیت اولیه استفاده شده است. برای ذخیره راه‌حل‌های مناسب گذشته و استفاده مجدد از این راه‌حل‌ها بعد از جابه‌جایی بهینه موردنظر از یک حافظه صریح استفاده شده است. حفظ راه‌حل‌های مناسب گذشته برای ردیابی بهینه جدید کمک شایانی به الگوریتم می‌نماید. در این روش برای به‌روزرسانی حافظه یک راهکار جدید پیشنهادشده که باعث می‌شود در هر زمان بهترین راه‌حل‌ها در حافظه ذخیره شوند. همچنین در این روش، خوشه‌بندی مناسبی برای جمعیت اصلی و حافظه انجام می‌گیرد. خوشه‌بندی استفاده‌شده تنوع لازم را در الگوریتم ایجاد نموده و در صورت تغییر در محیط الگوریتم به‌سرعت به همگرایی خواهد رسید. تبادل اطلاعات میان خوشه‌های متناظر (خوشه‌های با برچسب شبیه به هم) برای جمعیت اصلی و حافظه صورت می‌گیرد. این تناظر میان خوشه‌ها به افزایش تنوع در الگوریتم کمک شایانی می‌نماید. در مقایسه با روش‌های قدیمی این روش به‌طور مناسبی، کارایی را افزایش می‌دهد. استفاده از حافظه و خوشه‌بندی در این روش به افزایش سرعت همگرایی در مقایسه با دیگر روش‌های قبلی منجر می‌شود.

مزایای روش پیشنهادی در مقایسه با سایر روش‌های مشابه قبلی عبارت‌اند از:

۱- استفاده از به‌روزرسانی مناسبی برای حافظه که به افزایش سرعت همگرایی در الگوریتم کمک بسیار شایانی نموده است.

۲- خوشه‌بندی استفاده‌شده در این روش نسبت به روش‌های مشابهی که از خوشه‌بندی برای حفظ تنوع استفاده نموده‌اند مناسب‌تر عمل کرده و کارایی را برای الگوریتم بسیار بهبود بخشیده است.

۳- در این روش در مقایسه با سایر روش‌های مشابه قبلی به‌جای تصادفی‌سازی جمعیت اولیه از تئوری آشوب استفاده شده است که تئوری آشوب به بهبود کارایی الگوریتم کمک شایانی نموده است.

معایب روش پیشنهادی در مقایسه با سایر روش‌های مشابه قبلی عبارت‌اند از:

۱- کاهش کارایی الگوریتم در ابعاد خیلی بالا (ابعاد بالاتر از ۵۰)

۲-کاهش کارایی الگوریتم در صورت افزایش تعداد خوشه‌ها

۳- کاهش کارایی الگوریتم در صورت افزایش اندازه حافظه

این مقاله شامل ۹ بخش است. بخش ۱ شامل مقدمه و کلیات است. بخش ۲ مروری بر کارهای گذشته دارد. بخش ۳ به معرفی معیار سنجش خطای برون‌خطی می‌پردازد. بخش ۴ مسئله محک قله‌های متحرک که

معروف‌ترین شبیه‌ساز برای محیط‌های پویا در دنیای واقعی است را معرفی می‌نماید. بخش ۵ تئوری آشوب و معروف‌ترین تابع آشوب (تابع نگاشت لجستیک) را معرفی می‌کند. بخش ۶ به حافظه و انواع آن می‌پردازد. بخش ۷ روش پیشنهادی را تشریح می‌کند. بخش ۸ به آزمایش‌های انجام‌گرفته بر روی روش پیشنهادی می‌پردازد. بخش ۹ نتیجه‌گیری کلی از این مقاله را تشریح می‌کند.

۲- مروری بر کارهای گذشته

در بین روش‌های ارائه‌شده برای حل مسائل بهینه‌سازی پویا روش‌هایی که از استراتژی ترکیبی استفاده می‌کنند از جایگاه ویژه‌ای برخوردارند. استراتژی ترکیبی از به‌کارگیری تکنیک‌های مختلف در فرآیند حل مسئله حاصل می‌گردد. ازجمله روش‌های ترکیبی روش‌هایی هستند که ترکیب الگوریتم ژنتیک با تکنیک‌های مختلف است که چند نمونه از این روش‌ها در مرجع [۱، ۲] آمده است. در میان روش‌های ارائه‌شده در مسائل بهینه‌سازی ترکیبی در محیط‌های پویا می‌توان به تلاش‌های فراوان یانگ و همکاران اشاره کرد. بعضی از روش‌های ترکیبی که توسط یانگ و همکاران ارائه شده‌اند در ادامه آورده شده است. روش EIGA[۲] [۳] که توسط یانگ ارائه شده است یک راهکار ترکیبی است که از یک روش برای حفظ یک نسخه از بهترین کروموزوم‌ها استفاده می‌شود. راهکاری که در این روش استفاده شده است راهکاری بنام نخبه‌گرایی[۳] [۳] است. همچنین در این روش یک نسخه از بهترین راه‌حل‌ها که راه‌حل‌های نخبه‌ای هستند و خیلی قدیمی نیز نمی‌باشند ذخیره می‌شود. حفظ راه‌حل‌های خوب گذشته می‌تواند باعث افزایش سرعت همگرایی در الگوریتم شود. برای حفظ تنوع از روش مهاجرت نیز استفاده شده است. در این الگوریتم بهترین افراد از نسل‌های قبلی جایگزین بدترین افراد از جمعیت اصلی می‌شوند و قبل از انجام عملیات انتخاب[۴] و بازترکیب[۵]، $E(t-1)$ فرد نخبه از نسل‌های قبلی به‌عنوان نسل پایه انتخاب شده و جایگزین $P(t-1)$ فرد از جمعیت اصلی می‌شوند (t تعداد نسل‌ها را مشخص می‌کند) و سپس عملگرهای ژنتیکی برای بهبود نسل جدید اعمال می‌شود. در این روش برای به‌روزرسانی حافظه از روش شبیه‌ترین فرد استفاده می‌کند. بدین‌صورت که بهترین فرد جمعیت اصلی را جایگزین شبیه‌ترین فرد از حافظه می‌نماید (البته باید فرد جایگزین‌شده کارایی بالاتری نسبت به فرد خارج‌شده از حافظه داشته باشد) از مزایای این روش می‌توان به حفظ سطحی از تنوع با استفاده از حافظه و روش مهاجران تصادفی اشاره نمود [۳]. روش MIGA [۶] که توسط یانگ و همکاران ارائه شده است، از حافظه مهاجرتی استفاده نموده است. در این روش از حافظه مهاجرتی جهت ذخیره راه‌حل‌های خوب گذشته برای تولید مهاجرت‌ها استفاده می‌شود. روش مهاجران تصادفی به‌منظور افزایش تنوع در جمعیت به وجود آمده است که از همگرا شدن بیش‌ازحد الگوریتم به یک بهینه محلی جلوگیری کند. این روش پاسخ‌هایی را به‌صورت تصادفی تولید می‌کند و این پاسخ‌ها درون جمعیت درج می‌شوند. در روش مهاجران تصادفی، مهاجران را جایگزین $\frac{1}{4}$ از بدترین افراد درون جمعیت می‌کند. در روش MIGA در هر نسل بهترین فرد از حافظه انتخاب شده و مهاجرت‌ها را برای عملیات جهش تولید می‌نماید و اندازه حافظه $0/1 \times N$ در نظر گرفته شده است و $r_i \times N$ فرد از حافظه انتخاب می‌شوند و برای عملگر جهش با احتمال P_m^i استفاده می‌شوند (N اندازه جمعیت اصلی است و r_i نرخ مهاجرت و همچنین P_m^i احتمال جهش می‌باشند). در این روش زمانی که تغییر در محیط تشخیص داده شود بازیابی از حافظه صورت می‌گیرد. در این الگوریتم اطلاعات بازیابی‌شده از حافظه با مهاجران تصادفی از طریق عملگر جهش ترکیب می‌شود. برای هر نسل، حافظه مجدد ارزیابی شده و بهترین افراد حافظه بازیابی شده و به‌عنوان پایه برای تولید مهاجرت استفاده می‌شوند. نحوه به‌روزرسانی در این روش همانند روش EIGA است. از مزایای این روش می‌توان به تنوع ایجادشده در الگوریتم به کمک حافظه مهاجرتی استفاده شده اشاره نمود. یکی از عیوب این روش این است که در فرکانس تغییرات پایین و با افزایش شدت تغییرات در محیط کارایی الگوریتم افت محسوسی می‌نماید [۴]. در سال (۲۰۱۲) یانگ و همکاران روش CPSOR [۵] را ارائه نموده‌اند، که این روش ذرات را به افرازهایی (خوشه‌هایی) تقسیم نموده که ذرات موجود در هر افراز به جستجوی محلی در آن افراز می‌پردازند. خوشه‌بندی استفاده‌شده در این روش از نوع خوشه‌بندی سلسله مراتبی است. پیچیدگی محاسباتی برای روش خوشه‌بندی در این روش $O(M^3)$ است. M اندازه جمعیت را مشخص می‌کند. هر ذره به سمت میانگین بهترین موقعیت ملاقات‌شده توسط ذرات در افرازی که در آن قرار دارد حرکت می‌نماید. دست‌یابی به جواب بهینه برای توابع چندقله‌ای در محیط‌های پویا با استفاده از ایجاد تغییرات در الگوریتم تکاملی بهینه‌سازی گروه ذرات است. برای دست‌یابی به این امر، از یک مدل مبتنی بر اجزا که اجازه توسعه به زیرجمعیت‌های موازی را می دهد، استفاده می‌شود. این مدل راهکاری جهت ترغیب ردیابی شبیه‌سازی‌شده برای توابع چند قله‌ای است که از تجمع ذرات در قله‌ها جلوگیری می‌کند. در این الگوریتم به‌جای استفاده از یک گروه که دارای s ذره است و هدف آن‌ها یافتن جواب بهینه بردار n بعدی است، به ازای هر بعد یک گروه ذرات انتخاب می‌شود، بنابراین n گروه ذره استفاده می‌شود، که هر گروه خود دارای s ذره است. هر زیرجمعیت تلاش می‌کند تا قله محلی را ردیابی و"استخراج" کند. این زیر جمعیت‌ها یا خوشه‌ها به مرکزیت ذره‌ای با بهترین موقعیت در منطقه محلی که به‌صورت کره‌ای با شعاع r و به مرکزیت ذره با بهترین امتیاز تعریف شده است، قرار می‌گیرند. همه ذراتی که در یک خوشه قرار دارند بهترین تجربه شخصی مرکز خوشه خود را به‌عنوان بهینه عمومی می‌پذیرند. در این روش از معیاری بنام درجه همپوشانی ذرات[۷] استفاده می‌شود، بدین‌صورت که اگر تجمع ذرات در یک ناحیه به‌گونه‌ای باشد که ذرات روی‌هم قرار گیرند در آن صورت بعضی از ذرات به‌صورت تصادفی به ناحیه خلوت‌تری مهاجرت می‌کنند. از معیار

دیگری بنام درجه تنوع نیز برای حفظ تنوع بعد از هر تغییر در محیط استفاده می‌کند. یکی از عیوب این روش این است که در قله‌های با تعداد کم کارایی برای این روش کاهش می‌یابد [۵]. روش SOS با تعدادی زیرجمعیت جستجوی پایه برای راه‌حل‌های خوب شروع می‌شود. زمانی که یک قله (یک راه‌حل خوب) پیدا می‌شود جمعیت به دو زیربخش با نام‌های جمعیت پیش‌آهنگ و جمعیت جستجو تقسیم می‌شود. جمعیت پیش‌آهنگ باید قادر باشند که دیگر قله‌ها (راه‌حل‌های بهینه) را ردیابی نمایند، درحالی‌که جمعیت پایه قله‌های جدید (راه‌حل‌های بهینه جدید) را جستجو می‌نماید. این روش از جمعیت پایه برای بهره‌برداری و از جمعیت پیش‌آهنگ برای اکتشاف راه‌حل‌های جدید استفاده می‌کند. از مزایای این روش می‌توان به ایجاد تعادل میان اکتشاف و بهره‌وری، همچنین ایجاد تنوع با راهکار چندجمعیتی اشاره نمود [۶]. الگوریتم mQSO که به الگوریتم ازدحام ذرات مبتنی بر ذرات کوانتوم[۸] معروف است و در این الگوریتم کل جمعیت به چند گروه تقسیم می‌شوند و شامل سه عملگر تنوع با نام‌های ذرات کوانتوم، دفع و ضد همگرایی است. ذرات کوانتوم در موقعیت‌های تصادفی قرار می‌گیرند تا تنوع گروه‌ها را حفظ کنند. عملگر دفع هرگاه دو گروه همپوشانی پیدا می‌کنند، گروه بدتر را مقداردهی اولیه مجدد می‌کند. عملگر ضدهمگرایی زمانی که تمام گروه‌ها همگرا می‌شوند، گروه بدتر را مقداردهی اولیه مجدد می‌کند. از مزایای این روش به حفظ تنوع با راهکارهای ذرات کوانتوم و ضدهمگرایی می‌توان اشاره نمود. ازجمله معایب این روش می‌توان کاهش شدید کارایی با افزایش شدت تغییرات در محیط و افزایش ابعاد مسئله را نام برد [۷]. در الگوریتم Adaptive mQSO تعداد گروه‌ها از ابتدا معین نیست و با تغییر در محیط و پیدا کردن قله‌های جدید تعداد گروه‌ها افزایش می‌یابد، بدین‌صورت که در آن عملگر ضدهمگرایی هرگاه که همه گروه‌ها همگرا شدند یک گروه آزاد جدید ایجاد کرده که به پیدا کردن بهینه محلی جدید کمک می‌کند. ازجمله معایب این روش می‌توان به کاهش کارایی الگوریتم در ابعاد بالا و همچنین با کاهش تعداد قله‌های بهینه در فضای مسئله کارایی برای این روش کاهش می‌یابد [۸]. در روش Memory/Search،کل اندازه جمعیت به دو جمعیت تقسیم می‌شوند. که این دو جمعیت شامل جمعیت "حافظه" و جمعیت "جستجو" می‌باشند. اولین جمعیت مبتنی بر حافظه است و برای به خاطرسپاری راه‌حل‌های قدیمی و خوب استفاده می‌شود. جمعیت حافظه می‌تواند علاوه بر ذخیره راه‌حل‌های خوب در حافظه، بازیابی را نیز از حافظه انجام دهد. جمعیت دوم مبتنی بر جستجو است که برای اکتشاف قله‌های جدید و معرفی آن‌ها به حافظه استفاده می‌شود. جمعیت جستجو فقط عملیات ذخیره راه‌حل‌های جدید را در حافظه انجام می‌دهد و بازیابی از حافظه برای این جمعیت صورت نمی‌گیرد. جمعیت دوم به‌صورت تصادفی بعد از هر تغییری مقداردهی اولیه می‌شود. جمعیت جستجو فقط در این روش برای حفظ تنوع و افزایش سرعت همگرایی در الگوریتم از راهکار چندجمعیتی مبتنی بر حافظه و جستجو استفاده شده است[۹]. روش مهاجران تصادفی[۹] به‌منظور افزایش تنوع در جمعیت به وجود آمده است که از همگرا شدن بیش‌ازحد الگوریتم به یک بهینه محلی جلوگیری کند. این روش پاسخ‌هایی را به‌صورت تصادفی تولید می‌کند و این پاسخ‌ها درون جمعیت درج می‌شوند. در روش مهاجران تصادفی، مهاجران را جایگزین $\frac{1}{4}$ از بدترین افراد درون جمعیت می‌کند. روش مهاجران تصادفی به‌عنوان یکی از روش تک‌جمعیتی در روش‌های فراکتشافی محسوب می‌شود [۱۰]. در الگوریتم CPSO ذرات را به خوشه‌هایی تقسیم نموده که ذرات موجود در هر خوشه به جستجوی محلی در آن خوشه می‌پردازند. هر ذره به‌سمت میانگین بهترین موقعیت ملاقات‌شده توسط ذرات در خوشه‌ای که در آن قرار دارد حرکت می‌نماید. در این الگوریتم از معیاری بنام درجه روی‌هم افتادگی ذرات استفاده می‌شود که اگر در ناحیه‌ای از فضای جستجو شلوغی ذرات به وجود آمد به‌صورت تصادفی بعضی از ذرات که در این ناحیه ایجاد ازدحام نموده‌اند و به عبارتی روی‌هم افتاده‌اند به ناحیه خلوت‌تری از فضای جستجو مهاجرت تصادفی نمایند. دست‌یابی به جواب بهینه برای توابع چندقله‌ای در محیط‌های پویا با استفاده از ایجاد تغییرات در الگوریتم تکاملی بهینه‌سازی گروه ذرات است. برای دست‌یابی به این امر، از یک مدل مبتنی بر اجزا که اجازه توسعه به زیرجمعیت‌های موازی را می دهد، استفاده شده و جهت خوشه‌بندی این جمعیت‌ها روش مبتنی بر میانگین به کار گرفته شده است [۱۱]. روش MEGA [۱۰] از ترکیب حافظه با الگوریتم ژنتیک استفاده نموده است. حافظه و جمعیت اصلی مقداردهی اولیه تصادفی می‌شوند و از حافظه برای ذخیره راه‌حل‌های گذشته استفاده می‌نماید در این روش حافظه در یک بازه زمانی تصادفی به‌روزرسانی می‌شود و اندازه حافظه $N \times ۰/۱$ در نظر گرفته شده است. همچنین در این روش برای تشخیص تغییر در محیط بدین‌صورت عمل می‌شود که ارزیابی مجدد برای افراد جمعیت صورت می‌گیرد و در صورت تغییر در مقدار کارایی برای افراد جمعیت الگوریتم تغییر در محیط را تشخیص می‌دهد [۱۲]. در روش FMSO از یک گروه والد به‌عنوان یک گروه پایه برای شناسایی نواحی امیدبخش استفاده شده است و از گروه‌های فرزند برای جستجوی محلی استفاده می‌شود. هر گروه فرزند ناحیه مربوط به خود را جستجو می‌کند. در این روش به‌نوعی یک توازن میان جستجوی محلی و جستجوی سراسری به وجود آمده است. در روش FMSO ناحیه جستجو به شکل یک دایره به مرکزیت بهترین ذره گروه در نظر گرفته می‌شود. هر ذره که دارای فاصله کم‌تری از شعاع دایره باشد (فاصله نزدیک‌تری تا بهترین ذره که در مرکز قرار دارد) به آن گروه فرزند تعلق می‌یابد. در FMSO هر گروه فرزند دارای یک شمارنده خطا است که وظیفه آن کنترل مقدار برازندگی بهترین موقعیت یافت‌شده گروه فرزند است. اگر در هر مرحله بهبودی در مقدار برازندگی آن ایجاد نشده باشد مقدار شمارشگر خطا یک واحد افزایش پیدا می‌کند و درصورتی‌که مقدار شمارشگر خطا به بیشینه مقدار خود برسد در آن صورت از یک عملگر جهش بر روی بهترین ذره

برای پرش به موقعیت جدید استفاده می‌شود. تعداد گروه‌های فرزند با بهبود بهترین موقعیت یافت شده گروه والد افزایش پیدا می‌کند ولی مقدار آن از ماکزیمم مقدار تعیین‌شده برای آن تجاوز نمی‌نماید درصورتی‌که تعداد گروه‌های فرزند به بیشینه مقدار آن برسد گروه جدید جایگزین آن گروه فرزند که دارای شمارنده خطای بزرگ‌تری است می‌گردد. این روش در فرکانس‌های تغییرات پایین کارایی چندانی از خود نشان نمی‌دهد و این می‌تواند یکی از عیوب این روش محسوب شود. همچنین در این روش با افزایش شدت تغییرات در محیط کارایی افت محسوسی می‌نماید [۱۳]. روش جستجوی باکتریای جهت داده‌شده با ازدحام ذرات در مرجع [۱۴] پیشنهاد شده است. در این روش از ترکیب الگوریتم جستجوی باکتریای با ازدحام ذرات برای جایابی بهینه خازن‌ها و ژنراتورهای توزیع‌شده در شبکه‌های توزیع استفاده شده است. در این پژوهش با استفاده از الگوریتم ارائه شده مسئله چندهدفه جایابی خازن‌ها و ژنراتورها در شبکه توزیع شده بهینه می‌شود. تابع هدف در نظر گرفته شده شامل کاهش هزینه تلفات و بهبود پروفیل ولتاژ شبکه است. در این روش ابتدا مکان بهینه خازن‌ها با استفاده از تحلیل شاخص ولتاژ مشخص شده و سپس با استفاده از الگوریتم پیشنهادی مقدار توان تولیدی خازن‌ها محاسبه می‌گردد. روش ACPSO ترکیبی از دو الگوریتم APSO [۱۱] و CPSO [۱۲] است. در این روش از مزایای ساختار تطبیقی APSO و ساختار تعاونی CPSO بهره گرفته شده است. ساختار تطبیقی APSO باعث می‌شود در هر مرحله از اجرای الگوریتم پارامترها با مناسب‌ترین مقدار خود، معادله سرعت را به‌روزرسانی نمایند. ساختار تعاونی در این روش باعث می‌شود: ۱) برای حل مسئله‌های با ابعاد بالا مفید باشد، ۲) با افزایش تنوع جمعیت از به دام افتادن در بهینه محلی جلوگیری کرده و نرخ همگرایی را بهبود دهد، ۳) برخلاف دیگر روش‌ها که برخی از پارامترها را بهتر و برخی را بدتر می‌نمایند این روش در هر مرحله کلیه ابعاد مسئله را بهبود می‌دهد [۱۵].

هر یک از روش‌های معرفی‌شده به‌نوعی توانسته‌اند بعضی از چالش‌های موجود را برای مسائل بهینه‌سازی پویا حل نمایند. بعضی از این روش‌ها از راهکار چندجمعیتی، بعضی از راه‌کار حافظه و بعضی دیگر از راهکار خوشه‌بندی استفاده نموده‌اند.

روش پیشنهادشده در این مقاله تقریباً ترکیبی از تمام این روش‌ها محسوب می‌شود. در روش پیشنهادی که در ادامه تشریح شده است از چندین راه‌کار مناسب برای مواجهه با مسائل بهینه‌سازی پویا استفاده نموده است. در این روش از ترکیب خوشه‌بندی با حافظه و همچنین یک راه‌کار مناسب برای به‌روزرسانی اطلاعات از حافظه استفاده شده است. همچنین در این روش نشان داده شده که استفاده از رفتار آشوب‌گونه به‌جای رفتارهای تصادفی می‌تواند به بهبود کارایی الگوریتم در حل مسائل بهینه‌سازی پویا کمک نماید (اکثر تحقیقات اخیر از نظریه آشوب برای مسائل ایستا استفاده نموده‌اند). به‌طورکلی این روش پیشنهادشده دارای دو جنبه نوآوری اساسی است: ۱): به‌کارگیری یک روش خوشه‌بندی کا-میانگین که جمعیت حافظه و جمعیت اصلی را خوشه‌بندی می‌نماید و با ایجاد تناظر میان خوشه‌ها به حفظ تنوع در الگوریتم کمک می‌کند. ۲): به‌کارگیری یک روش به‌روزسانی مناسب برای ذخیره اطلاعات مفید گذشته در حافظه.

خوشه‌بندی مورداستفاده در این روش از راهکاری مناسب استفاده نموده تا حداکثر تنوع را در الگوریتم به وجود آورد. خوشه‌بندی استفاده‌شده از روشی استفاده نموده که با تبادل اطلاعات میان خوشه‌های متناظر هم تنوع را افزایش داده و هم جستجوی محلی را بهبود دهد. نحوه به‌روزرسانی حافظه به‌گونه‌ای است که بتواند در هر زمان مفیدترین اطلاعات را در حافظه ذخیره نماید. ذخیره مفیدترین اطلاعات در حافظه به بهبود کارایی (افزایش سرعت همگرایی) در الگوریتم کمک شایانی می‌کند.

۳- خطای برون‌خطی

برای نمایش دادن برتری یک الگوریتم نسبت به سایر الگوریتم‌ها بایستی ملاک‌های برای سنجش کارایی آن‌ها وجود داشته باشد تا از این طریق بتوان یک مقایسه کمی بین آن‌ها انجام داد. چندین معیار برای سنجش کارایی الگوریتم‌های تکاملی در محیط‌های پویا وجود دارند. ازآنجایی‌که برای مسائل بهینه‌سازی پویا یک راه‌حل بهینه تغییرناپذیر با زمان وجود ندارد یک هدف ثابت نمی‌تواند پیدا شود ولی می‌توان آن را در فضای جستجو تا حد امکان نزدیک به آن دنبال کرد. بسته به نوع مسئله‌ای که در دست است اندازه‌گیری‌های عددی مختلفی برای مقایسه وجود دارد. ازجمله این معیارها می‌توان به معیار خطای برون‌خطی، دقت، مجموع شایستگی‌ها، کارایی برون‌خطی اشاره نمود. معیار خطای برون‌خطی [۱۶] اولین بار توسط دی‌جونگ پیشنهاد شد و در محافل علمی نیز به تائید اکثر محققین رسیده است. اکثر محققین از معیار خطای برون‌خطی برای سنجش کارایی روش خود و مقایسه با دیگر روش‌ها استفاده نموده‌اند. در حقیقت میانگین خطای برون‌خطی به‌دست‌آمده از الگوریتم می‌تواند معیار مناسبی برای ارزیابی کارایی الگوریتم‌ها به‌صورت کمی محسوب شود. اگر بهینه صحیح در هر نقطه از زمان شناخته شده باشد میانگین خطا می‌تواند برای اندازه‌گیری کارایی مناسب باشد. میانگین خطا اجازه می‌دهد که مقدار کارایی را با پوشاندن تغییرات برازندگی بهینه با مانع کم‌تری بتوان مشاهده نمود. خطای برون‌خطی یک اندازه‌گیری تجمعی است بنابراین منحنی نتیجه رفته‌رفته صاف‌تر می‌شود.

خطای برون‌خطی طبق رابطه (۱) محاسبه می‌شود.

$$OfflineError = \frac{1}{FE_S}\sum_{t=1}^{FE_S}(h(t)-f(t)) \qquad (۱)$$

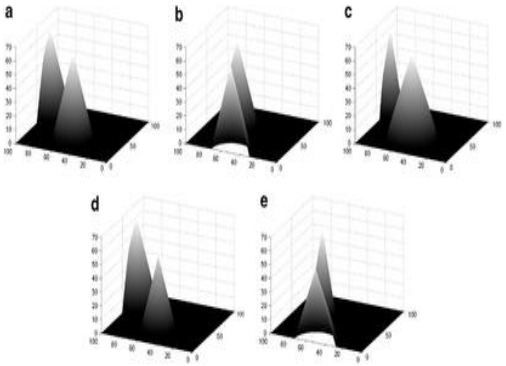
که در رابطه (۱)، FE_S برابر ماکزیمم تعداد ارزیابی کارایی برای افراد است و $h(t)$ مقدار بهینه سراسری قبل از تغییر است و $f(t)$ بهترین موقعیت پیداشده توسط افراد جمعیت در زمان tام است.

۴- مسئله محک قله‌های متحرک

مسئله محک قله‌های متحرک[۱۳] [۱۷] یک شبیه‌ساز مناسب برای شبیه‌سازی محیط‌های پویا است. این مسئله شامل n قله، در یک فضای m بعدی، با پارامترهایی با مقادیر واقعی است و ارتفاع، عرض و موقعیت قله‌ها در طول زمان می‌توانند تغییر یابند که این موضوع در حقیقت همان شبیه‌سازی یک مسئله پویا است. تابع محک قله‌های متحرک به‌صورت رابطه (۲) فرمول‌بندی می‌شود.

$$F(\vec{x},t)=\max(B(\vec{x}),\max P(\vec{x},h_i(t),w_i(t),\vec{p}_i(t))) \qquad (۲)$$

که $B(\vec{x})$ مقدار پایه محیط است که مستقل از زمان است و P تابعی است که شکل قله را تعریف می‌کند که هر یک از m قله، پارامترهای متغیر زمانی، ارتفاع $h_i(t)$ و عرض $w_i(t)$ و موقعیت $\vec{p}_i(t)$ خودشان را دارند در هر ΔE ارزیابی، ارتفاع ، عرض و محل هر یک از قله‌ها تغییر می‌کند. ارتفاع و عرض هر یک از قله‌ها با اضافه کردن یک متغیر تصادفی گوسی به آن‌ها تغییر می‌کند. پارامتر فرکانس تغییرات بیان می‌کند که در چه زمان‌هایی محیط تغییر پیدا می‌کند یا چه زمان الگوریتم بایستی به تغییرات محیط پاسخ دهد. تابع محک قله‌های متحرک دارای پارامترهای مختلفی است که با تغییر هر یک از این پارامترها در نتایج نیز ممکن است تغییر حاصل شود. شکل ۱ نحوه تغییرات قله‌ها را در این مسئله با دو قله نشان می‌دهد. پارامتر s شدت تغییر در قله‌ها را نشان می‌دهد که هرچه شدت تغییرات در قله‌ها بیش‌تر شود، ردیابی بهینه تغییریافته نیز مشکل‌تر خواهد بود.



شکل ۱: نحوه تغییرات در قله‌ها در تابع محک قله‌های متحرک

بنابراین پارامتر s میزان شدت تغییرات را کنترل می‌کند، ∇E فرکانس تغییرات را تعیین می‌کند. پارامتر λ تعیین می‌کند که چه مقدار تغییر موقعیت یک قله بستگی به حرکت قبلی خود دارد. درصورتی‌که مقدار λ=۰ باشد هر حرکت کاملاً تصادفی است، برای λ=۱ قله‌ها همیشه در یک مسیر مشخص حرکت می‌کنند. هر زمان یک تغییر در محیط رخ می‌دهد این تغییر بر روی مکان، ارتفاع و عرض یک قله به‌صورت روابط ذکرشده در (۳) و (۴) و (۵) و (۶) بیان می‌شود.

$$h_i(t)=h_i(t-1)+height_{severity}\cdot\sigma \qquad (۳)$$

$$w_i(t)=w_i(t-1)+width_{severity}\cdot\sigma \qquad (۴)$$

$$\vec{p}_i(t)=\vec{p}_i(t-1)+\vec{v}_i(t) \qquad (۵)$$

$$\sigma\in N(0,1) \qquad (۶)$$

بردار انتقال $\vec{v}_i(t)$ یک بردار تصادفی $\vec{r}$ را با بردار انتقال قبلی $\vec{v}_i(t-1)$ ترکیب می‌کند. بردار تصادفی $\vec{v}_i(t)$ به‌وسیله تولید کردن اعداد تصادفی در بازه (۱و۰) برای هر بعد و نرمال کردن آن به طول S ایجاد می‌شود. بردار $\vec{v}_i(t)$ را می‌توان وابسته به تغییر قبلی آن ایجاد نمود که در این صورت تغییر موقعیت قله‌ها همسو با تغییرات قبل آن می‌شود و یا به‌صورت تصادفی آن را ایجاد نمود که موجب می‌شود موقعیت قله‌ها به‌صورت تصادفی تغییر کند و هیچگونه وابستگی به تغییر قبلی نداشته باشد. بردار $\vec{v}_i(t)$ به‌صورت رابطه (۷) محاسبه می‌شود.

$$\vec{v}_i(t)=\frac{s}{\left|\vec{r}+\vec{v}_i(t-1)\right|}((1-\lambda)\vec{r}+\lambda\vec{v}_i(t-1)) \qquad (۷)$$

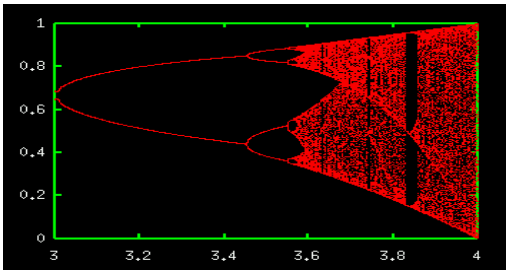
تابع قله برای ارتفاع، عرض و موقعیت هر قله به‌صورت فرمول (۸) محاسبه می‌شود.

$$P(\vec{x},h(t),w(t),\vec{p}(t))=h(t)-w(t).\sqrt{\sum_{j=1\ldots n}(x_j-p_j)^2} \qquad (۸)$$

بخش مربوط به رادیکال، فاصله بین نقطه موجود و موقعیت هر قله را بیان می‌نماید [۱۷].

۵- نظریه آشوب

نظریه آشوب[۱۴] [۱۹] یا نظریه بی‌نظمی‌ها به مطالعه سیستم‌های پویای آشوب‌گونه می‌پردازد. سیستم‌های آشوب‌گونه سیستم‌های پویای غیرخطی هستند که نسبت به شرایط اولیه خود بسیار حساس هستند. نمونه مشهور چنین سیستمی، مدل جمعیتی نگاشت لجستیک است. از ویژگی‌های تئوری آشوب می‌توان به خودسازمان‌دهی (وفق دادن خود با شرایط محیطی) در محیط‌های پویا و خودمانایی (هر جزئی از سیستم دارای ویژگی کل بوده و مشابه آن است) و حساسیت به شرایط اولیه اشاره کرد. شکل ۲ حساس بودن سیستم آشوب‌گونه به شرایط اولیه را نشان می‌دهد. در این شکل مشخص است که در سیستم‌های آشوب‌گونه یك واگرایی اولیه کوچك دلخواه نه‌تنها کوچك باقی نمی‌ماند بلکه به‌طور نمایی رشد می‌کند. همچنین با توجه شکل به‌وضوح می‌توان گفت که اگر از هرکدام از نقاط انتهایی شروع به حرکت نماییم درنهایت به نقطه شروع همگرا خواهیم شد که این از خواص یک سیستم آشوب‌گونه است.



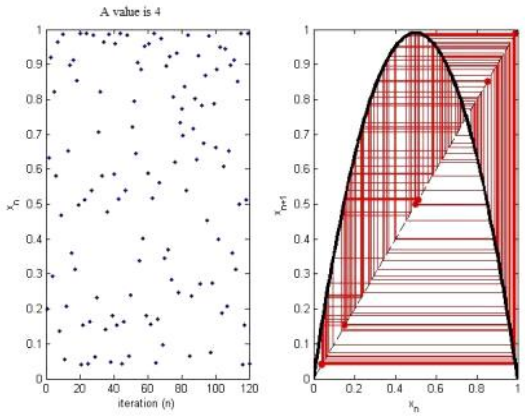
شکل ۲: حساسیت سیستم آشوب‌گونه به شرایط اولیه

در مدل‌های آشوب‌گونه خطای پیش‌بینی به‌سرعت رشد می‌کند و درنتیجه پیش‌بینی می‌تواند فقط برای مدت‌زمان محدود و با یك خطای پیش‌بینی مجاز صورت پذیرد زیرا وضعیت فعلی سیستم به وضعیت قبلی آن وابسته است. درحالی‌که در یک سیستم تصادفی، وضعیت فعلی سیستم مستقل از وضعیت پیشین آن است. در مدل تصادفی، حتی برای یك مدت‌زمان کوتاه هم نمی‌توان پیش‌بینی دقیق از خروجی مدل داشت. نمونه مشهور تابع نگاشت لجستیک به‌صورت رابطه (۹) محاسبه می‌شود.

$$r_{n+1} = A(1-r_n) \qquad (۹)$$

در رابطه (۹)، r_n یک عدد حقیقی در بازه [۱ و ۰] است و پارامتر A که به ضریب لجستیک معروف است، موجب ایجاد ویژگی‌های منحصربه‌فردی در این تابع می‌شود.

شکل ۳ نمودار مربوط به تابع نگاشت لجستیک با پارامتر A=۴ را نشان می‌دهد.



شکل۳: نمودار تابع لجستیک با مقدار A=۴

۶- حافظه

در مسائل پویا استفاده از حافظه برای ذخیره نمودن اطلاعات مفید گذشته و استفاده مجدد این اطلاعات در ردیابی مناسب پاسخ‌های مسئله می‌تواند مفید واقع شود. به‌طورکلی حافظه به دودسته کلی شامل حافظه صریح[۱۵] و دیگری حافظه ضمنی[۱۶] تقسیم می‌شود.

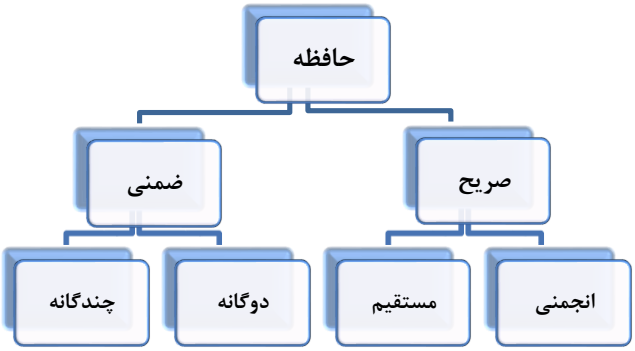
۶-۱- حافظه ضمنی

حافظه ضمنی برای ذخیره تمام اطلاعات (اطلاعات اضافی را نیز شامل می‌شود) استفاده می‌شود. در حقیقت از این حافظه برای ذخیره تمامی اطلاعات یک کروموزوم (هر کروموزوم دو یا بیش‌تر الل دارند) استفاده می‌شود. اطلاعات همگرایی یعنی توزیع الل در جمعیت می‌تواند به‌عنوان نمایش طبیعی محیط جاری به کار رود. حافظه ضمنی به دودسته تقسیم می‌شوند. یکی حافظه دوگانه و دیگری حافظه چندگانه. ازجمله کاربردهای حافظه ضمنی چندگانه در مرجع [۲۰] آمده است.

۶-۲- حافظه صریح

حافظه صریح برای ذخیره اطلاعات مفید از محیط استفاده می‌شود و برخلاف حافظه ضمنی که حتی اطلاعات اضافی را نیز ذخیره می‌کند، فقط اطلاعات مفید را ذخیره می‌نماید. حافظه صریح شامل دو نوع حافظه مستقیم و حافظه انجمنی است [۲۱]. در روش حافظه مستقیم راه‌حل‌های خوب به‌دست‌آمده توسط هر فرد (اطلاعات محلی) و یا راه‌حل‌های به‌دست‌آمده توسط همه افراد جمعیت (اطلاعات عمومی) به‌صورت مستقیم در حافظه ذخیره می‌شوند و در محیط‌های جدید مورداستفاده مجدد قرار می‌گیرند [۲۲]. در حافظه انجمنی اطلاعات محیطی همانند راه‌حل‌های خوب ذخیره می‌شوند ازجمله اطلاعات ذخیره‌شده در این نوع حافظه می‌توان، ذخیره لیستی از حالات فضای مسئله و یا احتمال رخداد یک جواب خوب در فضای مسئله را نام برد [۲۳] و در محیط جدید مورداستفاده مجدد قرار می‌گیرند.

شکل ۴ تقسیم‌بندی حافظه برای استفاده در محیط‌های پویا را نشان می‌دهد.



شکل ۴: تقسیم‌بندی حافظه برای محیط‌های پویا

ایده حفظ افراد نخبه (راه‌حل‌های خوب) در یک حافظه صریح مستقیم برای الگوریتم به نظر جذاب است. بدین ترتیب اگر بهینه دوباره در مکان قبلی خود ظاهر شود یک حافظه می‌تواند به‌راحتی آن را به خاطر آورد و خیلی سریع جمعیت را به سمت آن حرکت دهد. یکی از خواص محیط‌های پویا چرخه‌ای بودن محیط است. اگر یک محیط چرخه‌ای باشد به معنای آن است که نقاط بهینه ممکن است به نقاط قبلی خود بازگردند، بنابراین این حافظه کمک می‌کند که اگر بهینه‌ای به نقطه ماقبل خود برگشت بتوان به‌راحتی آن را ردیابی نمود و درنتیجه سرعت همگرایی الگوریتم را افزایش داد.

۶-۲-۱- راهکارهای جایگزینی در حافظه

برای جایگزینی افراد جمعیت در حافظه، راهکارهای مختلفی ارائه شده‌اند که چند نمونه از این راهکارهای جایگزینی در مرجع [۹، ۲۴] آمده است.

راهکار ۱: در این روش دو فرد در حافظه با کم‌ترین فاصله (بیش‌ترین شباهت) انتخاب شده و فردی که کارایی کم‌تری داشته باشد کاندید جایگزینی می‌شود. به‌طورمثال اگر *fit(i)* را به‌عنوان کارایی فرد *i*ام و

$fit(j)$ را به‌عنوان کارایی فرد jام در نظر بگیریم، اگر $fit(i)<fit(j)$ ، بهترین فرد جمعیت را جایگزین فرد i می‌نماییم و بالعکس.

راهکار ۲: اگر $fit(new) \leq fit(j) \times \frac{d_{ij}}{d_{max}}$ باشد درآن‌صورت فرد jام با بهترین فرد فعلی جایگزین می‌شود. که در این رابطه $fit(j)$ را کارایی فرد jام و d_{ij} فاصله بین دو فرد i و j ، d_{max} ماکزیمم فاصله ممکن بین دو فرد i و j است.

راهکار ۳: این روش به راهکار مشابهت معروف است، که در این روش بهترین فرد فعلی از جمعیت، جایگزین شبیه‌ترین فرد در حافظه می‌گردد، به شرطی که این فرد دارای کارایی بیش‌تری نسبت به فرد حافظه باشد. برای اندازه‌گیری شباهت می‌توان از فاصله اقلیدسی استفاده نمود.

۶-۲-۲- بازیابی از حافظه

اطلاعات ذخیره‌شده در حافظه را باید برای ردیابی بهینه جدید استفاده نمود. بنابراین بهترین زمان برای بازیابی اطلاعات از حافظه، لحظه‌ای است که محیط دچار تغییر می‌شود. برای بازیابی از حافظه می‌توان استراتژی‌های مختلفی اتخاذ نمود. یکی از استراتژی‌های بازیابی از حافظه بدین‌صورت است که بهترین فرد از حافظه را مشخص نموده و آن را جایگزین بدترین فرد از درون جمعیت اصلی نمود [۲۴].

۷- روش پیشنهادی

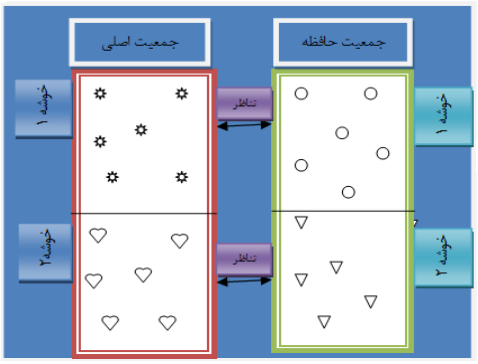
در این مقاله یک الگوریتم ژنتیک آشوب‌گونه با ترکیب حافظه برای حفظ راه‌حل‌های مناسب و خوشه‌بندی جمعیت در داخل حافظه و جمعیت اصلی ارائه شده است. گام‌های اصلی روش پیشنهادی در ادامه تشریح شده‌اند. در بخش‌های بعدی شباهت و تفاوت‌های میان روش پیشنهادی با دیگر روش‌های مشابه آورده شده است.

۷-۱- ایجاد جمعیت اولیه بر اساس تئوری آشوب

در این روش برخلاف الگوریتم ژنتیک استاندارد [۱۸] که جمعیت اولیه به‌صورت تصادفی ایجاد می‌شود، از تئوری آشوب برای ایجاد جمعیت اولیه استفاده شده است. همان‌طور که در بخش‌های قبل بیان شد، یک سیستم آشوب‌گونه پیش‌بینی دقیق‌تری نسبت به یک سیستم تصادفی از آینده دارد. در یک سیستم آشوب‌گونه (تابع نگاشت لجستیک) اگر از هرکدام از نقاط انتهایی شروع به حرکت کرد درنهایت به نقطه شروع همگرا می‌شود که این از خواص یک سیستم آشوب‌گونه است. بنابراین برای افزایش سرعت همگرایی در الگوریتم پیشنهادی جمعیت اولیه بر طبق تئوری آشوب (تابع نگاشت لجستیک) ایجاد می‌شود.

۷-۲- خوشه‌بندی

خوشه‌بندی یک از شاخه‌های یادگیر بی ناظر است که در طی آن نمونه‌ها به یک سری خوشه افراز می‌شوند به‌گونه‌ای که نمونه‌های موجود در هر خوشه ازنظر شباهت به هم نزدیک هستند و با نمونه‌های موجود در دیگر خوشه‌ها متفاوت هستند. معیارهای زیادی برای اندازه‌گیری شباهت وجود دارد که یکی از این معیارها می‌تواند فاصله اقلیدسی بین دو نمونه باشد. نمونه‌هایی که به هم شبیه هستند فاصله اقلیدسی کم‌تری نسبت به هم دارند و در یک خوشه قرار می‌گیرند. به این نوع خوشه‌بندی، خوشه‌بندی مبتنی بر فاصله می‌گویند. یکی از روش‌های خوشه‌بندی مبتنی بر فاصله، روش مبتنی بر مرکز است. مرکز هر خوشه را می‌توان از طریق میانگین داده‌های آن خوشه نمایش داد. در این روش مبتنی بر میانگین[۱۷] داده‌های شبیه به مرکز خوشه به آن خوشه تعلق پیدا می‌کنند. در این روش پیشنهادی از یک خوشه‌بندی مبتنی بر میانگین استفاده شده است و مرکز هر خوشه برابر میانگین افراد درون هر خوشه است، و ایده موردنظر بدین‌صورت است که افراد (کروموزوم‌ها) در جمعیت اصلی و حافظه خوشه‌بندی می‌شوند. تعداد خوشه‌ها در جمعیت اصلی و جمعیت حافظه باهم برابر هستند. تعداد خوشه‌ها می‌تواند بر روی کارایی روش پیشنهادی تأثیر به سزایی داشته باشد و این موضوع در بخش نتایج تجربی به‌وضوح تشریح شده است. خوشه‌بندی جمعیت می‌تواند به‌نوعی تنوع را که یکی از اصلی‌ترین چالش‌های موجود در محیط‌های پویا است را حفظ نماید. درواقع خوشه‌بندی جمعیت باعث می‌شود که فضای جستجوی محلی به‌خوبی مورد جستجو قرار گیرد و به بهبود جستجوی محلی توسط افراد هر خوشه کمک می‌نماید. هر فرد هم برای درج در جمعیت و هم برای بازیابی باید در خوشه متناظر به خود قرار گیرد. در این خوشه‌بندی افراد درون هر خوشه رتبه‌بندی می‌شوند. رتبه‌بندی افراد در هر خوشه بر اساس کارایی آن‌ها است. شکل ۵ نحوه خوشه‌بندی و تناظر میان خوشه‌ها را در دو جمعیت نشان می‌دهد.



شکل ۵: نحوه خوشه‌بندی و تناظر میان خوشه‌های دو جمعیت

۷-۳- حافظه

حافظه مورداستفاده در این روش از نوع حافظه صریح مستقیم بوده که در بخش‌های گذشته توضیح داده شد. در ادامه نحوه به‌روزرسانی و بازیابی از حافظه آورده شده است. حافظه مورداستفاده طبق نظریه آشوب و بر اساس تابع نگاشت لجستیکی مقداردهی اولیه می‌شود. اندازه حافظه موردنظر (mem_size) برابر $0/1 \times N$ است. که N اندازه جمعیت یا همان تعداد افراد جمعیت است. به‌روزرسانی حافظه در بازه‌های زمانی تصادفی صورت می‌گیرد، بدین‌صورت که اگر به‌روزرسانی

فعلی حافظه در بازه زمانی $rand(5,10)$ صورت گیرد، به‌روزرسانی بعدی در زمان $t + rand(5,10)$ است.

۷-۳-۱- نحوه به‌روزرسانی حافظه

اندازه حافظه برای ذخیره‌سازی اطلاعات محدود است بنابراین باید یک راه‌کار مناسب جهت به‌روزرسانی حافظه اتخاذ نمود. برای به‌روزرسانی حافظه از روش متفاوتی نسبت به راهکارهای ذکرشده در بخش‌های گذشته استفاده شده است. در این روش به هر یک از افراد موجود در حافظه یک مشخصه بنام سن اختصاص داده می‌شود. افراد در مرحله اول ورودشان به حافظه دارای سن صفر هستند و در هر نسل یک واحد به سن آن‌ها اضافه می‌گردد. در این روش زمانی که حافظه پر شود و نیاز به جایگزینی داشته باشد سن افراد طبق یک رابطه ترکیب خطی از سن واقعی افراد و کارایی آن‌ها محاسبه می‌شود. رابطه ترکیب خطی برای محاسبه سن افراد به‌صورت رابطه (۱۰) بیان می‌شود.

$$age_i = age_i + 1 + r \times fitness_i, \quad \forall\, i = 1 \dots m, \forall\, r \in (0,1) \qquad (۱۰)$$

در رابطه (۱۰) پارامتر r نرخ کارایی است و m اندازه حافظه را مشخص می‌کند. ایده موردنظر بدین‌صورت است که اگر حافظه پر شود و نیاز به به‌روزرسانی داشته باشد بهترین فرد جمعیت اصلی انتخاب شده و درصورتی‌که میزان کارایی این فرد بیش‌تر باشد جایگزین جوان‌ترین فرد حافظه می‌گردد. شبه‌کد مربوط به به‌روزرسانی حافظه در شکل ۶ آورده شده است. در این شبه‌کد B_p بهترین فرد از جمعیت اصلی را مشخص می‌کند. M_{sel} جوان‌ترین فرد از حافظه را مشخص می‌کند. $fitness$ بیانگر کارایی برای هر فرد است. age_i بیانگر سن فرد iام از جمعیت حافظه است.

Algorithm 1: memfunction(mem)

Input: *m is memory size;* $r \in (0,1)$.
Output: $\emptyset$

1. *initialize memory randomly*
2. $age_i = 0$
3. *every generation:*

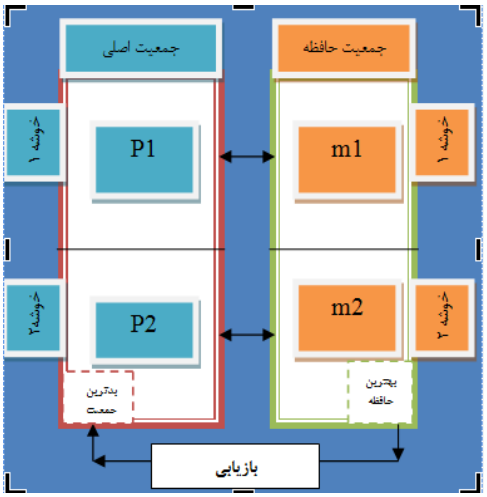
$$age_i = age_i + 1 + r \times fitness_i,$$

4. ***if*** *it is time to update memory*
   B_p *i s the best individual of the population*
   select memory individual M_{sel} *with the lowest age*
   if *fitness(*M_{sel}*) < fitness (*B_p*)*
   memory individual M_{sel} *is replaced by* B_p
   end of if
5. ***end of if***

شکل ۶: نحوه به‌روزرسانی حافظه

۷-۳-۲- نحوه بازیابی از حافظه

در این روش بازیابی از حافظه زمانی صورت می‌گیرد که محیط دچار تغییر شود. نحوه بازیابی از حافظه بدین‌صورت است که بهترین فرد از حافظه جایگزین بدترین فرد از جمعیت اصلی می‌شود. زمان تغییر در محیط بدین‌صورت شناسایی می‌شود که اگر شایستگی برای یکی از افراد جمعیت در ارزیابی مجدد تغییر نماید در این صورت الگوریتم تغییر در محیط را شناسایی نموده و باید بازیابی از حافظه انجام گیرد. ایده موردنظر بدین‌صورت است که بهترین فرد در هر خوشه از حافظه را مشخص نموده و بدترین فرد از هر خوشه جمعیت اصلی نیز مشخص شده و جایگزینی به‌صورت نظیر به نظیر در خوشه‌های حافظه و جمعیت اصلی صورت می‌گیرد. اگر p تعداد خوشه‌ها و q_x خوشه مربوط به جمعیت اصلی و q_m خوشه مربوط به حافظه را مشخص نماید، در این صورت شایسته‌ترین فرد از هر خوشه در حافظه انتخاب و همچنین بدترین فرد از هر خوشه از جمعیت اصلی نیز انتخاب شده و شایسته‌ترین فرد حافظه جایگزین بدترین فرد جمعیت اصلی می‌شود. بدین‌صورت که اگر j_r فرد انتخاب‌شده خوشه r-ام از جمعیت حافظه باشد این فرد دارای بالاترین شایستگی در خوشه مربوط به خود بوده و جایگزین بدترین فرد انتخاب‌شده از خوشه متناظر در جمعیت اصلی که همان فرد d_r است خواهد شد (هر فرد خارج‌شده از هر خوشه باید در خوشه متناظر با همان خوشه قبلی خود در جمعیت اصلی قرار گیرد) که در حقیقت $x_{j_r} = mem_{j_r}$ است که بیان‌گر همان جایگزینی بدترین فرد از هر خوشه جمعیت اصلی با بهترین فرد از خوشه متناظر در جمعیت حافظه است. نحوه بازیابی اطلاعات از حافظه به‌صورت شکل ۷ نشان داده شده است.



شکل ۷: نحوه بازیابی از حافظه

همان‌طور که شکل ۷ نشان می‌دهد بهترین فرد از هر خوشه حافظه با بدترین فرد از خوشه متناظر خود در جمعیت اصلی جایگزین می‌شود. در شکل ۵، p مراکز خوشه‌های جمعیت اصلی و m مراکز خوشه‌های حافظه است و همچنین best بهترین فرد از هر خوشه و worst بدترین فرد از هر خوشه است. در این روش پیشنهادی روش خوشه‌بندی به نحوی هم جستجوی محلی را بهبود دهد و هم با یک استراتژی جایگزینی مناسب در خوشه‌ها می‌تواند تنوع را در جمعیت افزایش دهد. همچنین با تبادل اطلاعات بین خوشه‌های متناظر به‌نوعی دامنه جستجوی سراسری در الگوریتم بهبود یافته است. این توازن میان جستجوی سراسری و محلی به نحوی سرعت همگرایی را در الگوریتم افزایش می‌دهد. شبه‌کد مربوط به الگوریتم ژنتیک و تولید نسل در این الگوریتم به‌صورت تابع *popfunction* بیان شده است. همان‌طور که بیان شد روش پیشنهادی ترکیبی از الگوریتم ژنتیک

استاندارد با خوشه‌بندی و حافظه است، که شبه‌کد الگوریتم ژنتیک برای استفاده در بدنه اصلی الگوریتم به‌صورت یک تابع در شکل ۸ بیان شده است. شبه‌کد مربوط به روش پیشنهادی به‌صورت شکل ۹ است.

Algorithm 2: popfunction(x)

function **{**

Repeat

1. *initialize population*
2. *evaluate the individual fitness of a certain proportion of the population*
3. *Apply the ranking evaluation individuals*
4. *select pair of best solution with roulette wheel strategy*
5. *apply crossover operator*
6. *apply mutation operator*
7. *until termination condition*
8. *end*

}

شکل ۸: شبه‌کد مربوط به الگوریتم ژنتیک استاندارد

Algorithm3: Proposed Algorithm

Input: *N, D, MCN,* x_j^{min}, x_j^{max}, *mem_size*

Output: *BEST Solution, BEST Fitness, offline error*

1. *Begin*
2. *initialize x and mem with* ***chaos theory***
   % x is population and mem is memory population
3. $f_i = fit(x_{:i})$ *% f_i is fitness for population*
4. $m_i = fit(mem_i)$ *% m_i is fitness for memory population*
5. *update_mem=1*
6. *update_time=rand(5,10)*
7. $cycle = 1$
8. *Repeat*
9. *update_mem:*
10. *update_mem=0*
11. *mem=update_memfunction(mem,x)*
12. *update_time=rand(5,10)+cycle*
13. *cx=update_popfunction(x) % cx is current x*
14. $cf_i = fit(x_{:i})$ *% fitness for current x*
15. *if* $\left(\exists i: \left((fit(mem_i) \neq m_i) \vee (fit(x_i) \neq f_i)\right)\right)$
    $ChangeFlag = 1$ *%Change detected and reuse memory*
    1- $\boldsymbol{m_i = fit(mem_{:i})}$, $\boldsymbol{f_i = fit(x_{:i})}$
    2- $\boldsymbol{mq_m = cluster(mem, p)}$ *%p is number of clusters*
    3- select $\boldsymbol{j_r \in mq_r \ \forall r \in \{1 \dots p\}}$
    subject to $\boldsymbol{m_{j_r} \leq m_l \ \forall l \in q_r}$
    4- $\boldsymbol{xq_x = cluster(x, p)}$
    5- select $\boldsymbol{d_r \in xq_r \ \forall r \in \{1 \dots p\}}$
    subject to $\boldsymbol{f_{dr} \geq f_l \ \forall l \in \{1 \dots p\}}$
    6- $\boldsymbol{x_{d_r} = mem_{j_r} \quad \forall r \in \{1 \dots p\}}$
16. *If*$(update\_time \geq cycle)$
17. $cycle = cycle + 1$
18. *Until cycle =* MCN
19. *End*

شکل ۹: شبه‌کد مربوط به روش پیشنهادی

در مرحله ۱۶ در اثر تغییر در محیط باید از اطلاعات ذخیره‌شده در حافظه برای محیط جدید استفاده نمود که این مرحله خود در ۶ مرحله مختلف صورت می‌گیرد که این شش مرحله به‌صورت زیر تشریح می‌شوند.

مرحله ۱: کارایی برای جمعیت اصلی و جمعیت حافظه محاسبه می‌گردد.

مرحله ۲: جمعیت حافظه خوشه‌بندی می‌شود.

مرحله ۳: بهترین فرد از هر خوشه از جمعیت حافظه انتخاب می‌شود که در حقیقت بهترین فرد از این جمعیت فردی است که بیش‌ترین کارایی را داراست.

مرحله ۴: جمعیت اصلی خوشه‌بندی می‌شود.

مرحله ۵: بدترین فرد از هر خوشه جمعیت اصلی انتخاب می‌شود (بدترین فرد از هر خوشه فردی است که کم‌ترین کارایی را دارا است).

مرحله ۶: بهترین فرد از هر خوشه در حافظه جایگزین بدترین فرد از خوشه متناظر در جمعیت اصلی می‌شود.

۷-۴- شباهت‌های روش پیشنهادی با روش‌های مشابه یانگ

روش پیشنهادی از جهاتی شبیه به روش‌های ارائه‌شده توسط یانگ است که برخی از این شباهت‌ها به‌صورت زیر بیان می‌شوند:

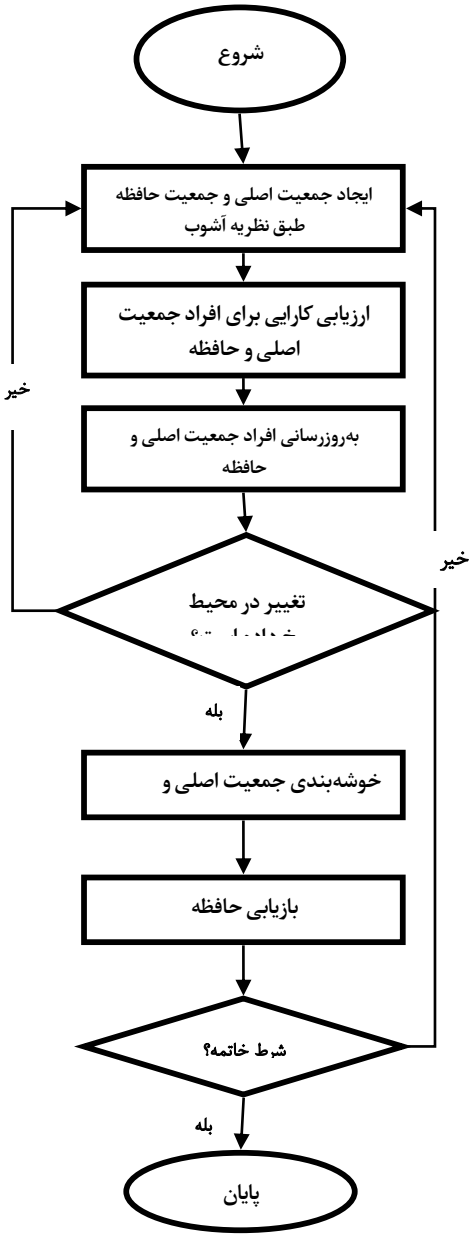
۱. هر دو روش از یک حافظه برای ذخیره‌سازی اطلاعات استفاده می‌نمایند.
۲. هر دو روش از حافظه برای حفظ سطحی از تنوع در جمعیت استفاده می‌نمایند.
۳. هر دو روش زمانی که محیط دچار تغییر می‌شود از حافظه اطلاعات را بازیابی می‌کنند.

۷-۵- تفاوت‌های روش پیشنهادی با روش‌های مشابه یانگ

۱. روش پیشنهادی از یک نوع استراتژی متفاوت برای به‌روزرسانی حافظه استفاده می‌نماید.
۲. روش پیشنهادی برای مقداردهی اولیه حافظه و جمعیت اصلی از نظریه آشوب استفاده می‌نماید (برخلاف دیگر روش‌ها که جمعیت مقداردهی اولیه تصادفی می‌شوند).
۳. روش پیشنهادی از یک نوع تکنیک خوشه‌بندی با استراتژی متفاوت استفاده می‌نماید، به‌طوری‌که در این تکنیک خوشه‌بندی برخلاف روش CPSO یانگ که فقط جمعیت اصلی را خوشه‌بندی می‌نماید، این روش حافظه را نیز خوشه‌بندی نموده و تبادل اطلاعات میان جمعیت اصلی و حافظه فقط توسط خوشه‌های متناظر صورت می‌گیرد و این ایده به‌نوعی جستجوی محلی را در الگوریتم بهبود داده است.
۴. روش‌های با حافظه ارائه‌شده توسط یانگ بر روی مسائل محک دیگری آزمایش شده‌اند درحالی‌که روش پیشنهادی از مسئله محک قله‌های متحرک برای شبیه‌سازی محیط‌های پویا استفاده نموده است.

۷-۶- پیچیدگی محاسباتی روش پیشنهادی

مؤلفه‌های اصلی روش پیشنهادی شامل خوشه‌بندی و به‌روزرسانی حافظه می‌باشند. عملیات خوشه‌بندی برای روش پیشنهادی فقط یکبار در هر بار تغییر انجام می‌گیرد. این عملیات بعد از تغییر در محیط انجام می‌شود، پس تعداد کل اجراهای این الگوریتم، $\frac{ite}{cf}$ بار است که ite نشان‌دهنده تعداد محاسبه کارایی و cf نشان‌دهنده فرکانس تغییرات است.



شکل ۱۰: نمودار گردش کار برای روش پیشنهادی

با توجه به این موضوع پیچیدگی محاسباتی روش پیشنهادی از مرتبه $O\left(\frac{ite}{cf}\times(M\times I\times k+M\times\log M)+ite\times B\right)$ است، که k. I, M و B به ترتیب اندازه جمعیت، تعداد چرخه‌های الگوریتم خوشه‌بندی، تعداد خوشه‌ها در الگوریتم خوشه‌بندی و زمان اجرای تابع کارایی است. با توجه به آنکه اصل زمان مصرفی در مرتب‌سازی کارایی افراد صرف می‌شود، مرتبه الگوریتم را می‌توانیم فرض کنیم که از جنس $O\left(\frac{ite}{cf}\times M\times\log M\right)$ خواهد بود که باحالت ساده آن یکسان است.

۷-۷- نمودار گردش کار برای روش پیشنهادی

برای درک بهتر موضوع ارائه‌شده نمودار گردش کار برای روش پیشنهادی در شکل ۱۰ نشان داده شده است. در این نمودار ترسیم‌شده برای روش پیشنهادی مشخص است که در صورت تغییر در محیط خوشه‌بندی برای جمعیت اصلی و جمعیت حافظه انجام می‌شود. بازیابی از حافظه نیز زمانی صورت می‌گیرد که تغییر در محیط رخ دهد. به‌روزرسانی برای جمعیت اصلی مطابق الگوریتم ژنتیک صورت می‌گیرد. به‌روزرسانی برای حافظه نیز مطابق راهکاری که در بخش‌های قبلی بیان شد صورت می‌گیرد. شرط خاتمه در این روش پیشنهادی تعداد تکرارهایی است که برای این روش در نظر گرفته شده است. در صورت رسیدن به پایان تکرارها الگوریتم خاتمه می‌یابد.

۸-۷- دلایل انتخاب الگوریتم پیشنهادی

- استفاده از حافظه صریح مستقیم برای ذخیره راه‌حل‌های مفید گذشته و استفاده مجدد از این اطلاعات در صورت تغییر در پارامترهای مسئله به افزایش کارایی کمک شایانی می‌کند.
- حفظ تنوع در حین اجرای الگوریتم و ایجاد تنوع بعد از تغییر در محیط با استفاده از حافظه و خوشه‌بندی باعث افزایش کارایی الگوریتم می‌شود.
- مقداردهی اولیه جمعیت بر اساس نظریه آشوب باعث افزایش سرعت همگرایی در الگوریتم می‌شود.
- ردیابی مناسب پاسخ‌های مسئله در صورت تغییر در محیط
- حل مشکل نامعتبر بودن حافظه افراد در صورت تغییر در محیط به‌وسیله یک حافظه مناسب
- بهبود جستجوی محلی در الگوریتم با استفاده از یک تکنیک خوشه‌بندی مناسب

۸- آزمایش‌ها

برای انجام آزمایش‌ها و تست کارایی روش پیشنهادی باید از تابع محکی استفاده شود که به‌خوبی بتواند یک محیط پویا را شبیه‌سازی نماید. جهت آزمایش بر روی روش پیشنهادی و مقایسه آن با سایر روش‌ها از تابع محک قله‌های متحرک برای شبیه‌سازی این الگوریتم‌ها در محیط‌های پویا استفاده می‌شود. در ادامه تنظیمات مربوط به پارامترهای تابع محک قله‌های متحرک برای ارزیابی الگوریتم پیشنهادی در محیط‌های پویا ذکر می‌شوند. روش پیشنهادی با روش‌های FMSO [۱۳]، CPSO [۱۱]، mQSO [۷]، rPSO [۲۵]، Cellular PSO [۲۶]، ESCA [۲۷]، CESO [۲۸] و CPSOR [۵] همچنین با روش‌های با حافظه ازجمله Memory/Search [۹]، SEAm [۱۷] و RIm [۲۹] مقایسه شده است. در این مقاله سعی شده است که مقایسه با روش‌هایی صورت گیرد که شرایط آن‌ها با شرایط روش پیشنهادی برابر باشد و بر روی یک مسئله محک (محک قله‌های متحرک) آزمایش شده باشند. در اینجا برای mQSO از پیکربندی (۱۰(۵+۵q استفاده شده است که در آن ۱۰ گروه ایجاد می‌شود و در آن هر یک از گروه‌ها دارای ۵ ذره خنثی و دارای ۵ ذره کوانتوم می‌باشند. همچنین برای این الگوریتم شعاع کوانتوم برابر ۰/۵ و شعاع دفع و شعاع همگرایی برابر ۳۱/۵ تعیین شده است. برای

الگوریتم FMSO ماکزیمم تعداد گروه‌های فرزند برابر ۱۰، شعاع دفع مابین گروه‌های فرزند برابر ۲۵، تعداد ذرات در گروه والد و گروه‌های فرزند به ترتیب برابر ۱۰۰ و ۱۰ در نظر گرفته شده است. برای cellular PSO یک آتاماتای سلولی ۵ بعدی با ۱۰۵ سلول و همسایگی مور ۱۸ با شعاع ۲ سلول در فضای جستجو به کار رفته است. بیشینه سرعت ذرات برابر شعاع همسایگی و ماکزیمم تعداد ذرات برای هر سلول برابر ۱۰ و شعاع جستجوی محلی برابر ۰/۵ تعیین شده است. همچنین جستجوی محلی را همه ذرات پس از مشاهده تغییر در محیط برای یک مرحله اجرا می‌کنند. برای روش CPSO اندازه جمعیت اصلی برابر ۱۰۰ و اندازه بزرگ‌ترین زیرگروه ۳۰ در نظر گرفته شده است. در روش CPSO ضرایب مؤلفه‌های اجتماعی و شناختی c_1 و c_2 در سرعت ذره به ترتیب برابر ۲/۸ و ۱/۳ و وزن اینرسی w میانگین c_1 و c_2 فرض شده است (۲/۰۵). پارامتر همپوشانی در این روش برابر ۰/۷ در نظر گرفته شده است. برای سه روش مبتنی بر حافظه memory/search و SEAm و RIm اندازه حافظه برابر ۱۰ و اندازه جمعیت اصلی برابر ۱۰۰ است. جدول ۱ تنظیمات پیش‌فرض مربوط به تابع قله‌های متحرک را نشان می‌دهد که مقادیر آن مبتنی بر سناریوی دوم برانک [۱۷] است. پارامترهای ذکرشده برای مسئله محک قله‌های متحرک در بخش‌های گذشته به‌صورت مختصر معرفی شده‌اند. جدول ۲ پارامترهای مربوط به الگوریتم پیشنهادی را نشان می‌دهد. نتایج به‌دست‌آمده از سایر الگوریتم‌ها از مرجع مربوط به آن‌ها استخراج شده است. تمامی آزمایش‌ها در محیط نرم‌افزار متلب صورت گرفته است.

جدول ۱: تنظیمات استاندارد پارامترها برای تابع قله‌های متحرک

پارامتر	مقدار
تعداد قله‌ها	۱۰
فرکانس تغییرات	۵۰۰۰
شدت تغییرات ارتفاع قله‌ها	۷
شدت تغییرات عرض قله‌ها	۱
شکل قله‌ها	مخروطی
میزان حرکت مکان قله‌ها	۱
ابعاد فضای جستجو (A)	۵
محدوده ارتفاع قله‌ها (H)	[۳۰ و ۷۰]
محدوده عرض قله‌ها (W)	[۱ و ۱۲]
ارتفاع استاندارد قله‌ها (I)	۵۰
محدوده‌ی فضای جستجو	[۰ و ۱۰۰]
پارامتر λ	۰
طول تغییرات (s)	۱

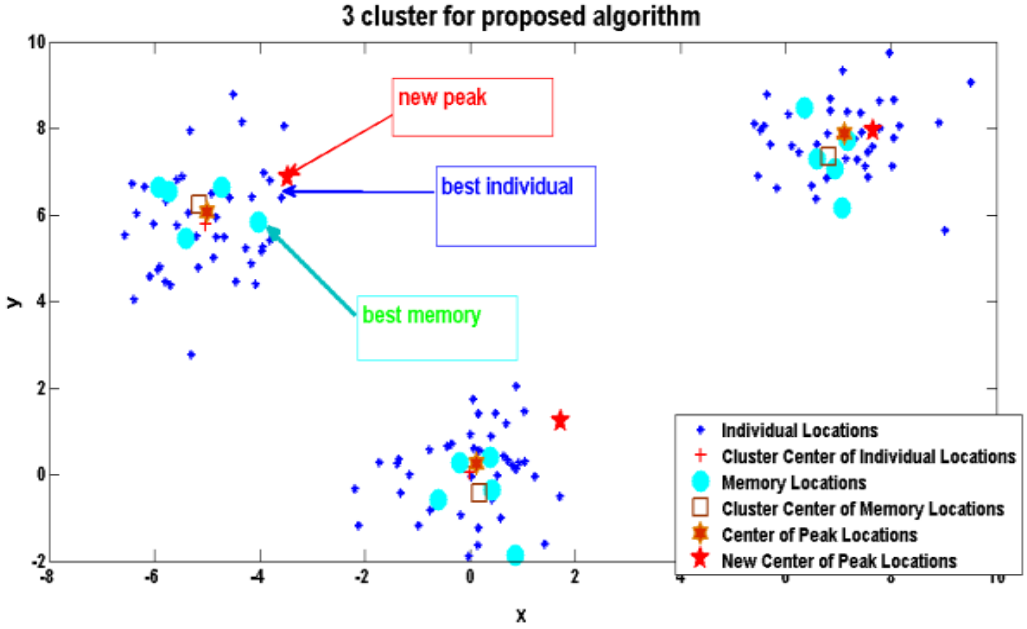
جدول2: پارامترهای الگوریتم پیشنهادی

پارامتر	مقدار
کران پایین	۰
کران بالا	۱۰۰
تعداد افراد جمعیت	۱۰۰
اندازه حافظه	۱۰
احتمال تقاطع	۰/۶
احتمال جهش	۰/۲
تعداد خوشه‌های جمعیت اصلی	۲
تعداد خوشه‌های جمعیت حافظه	۲

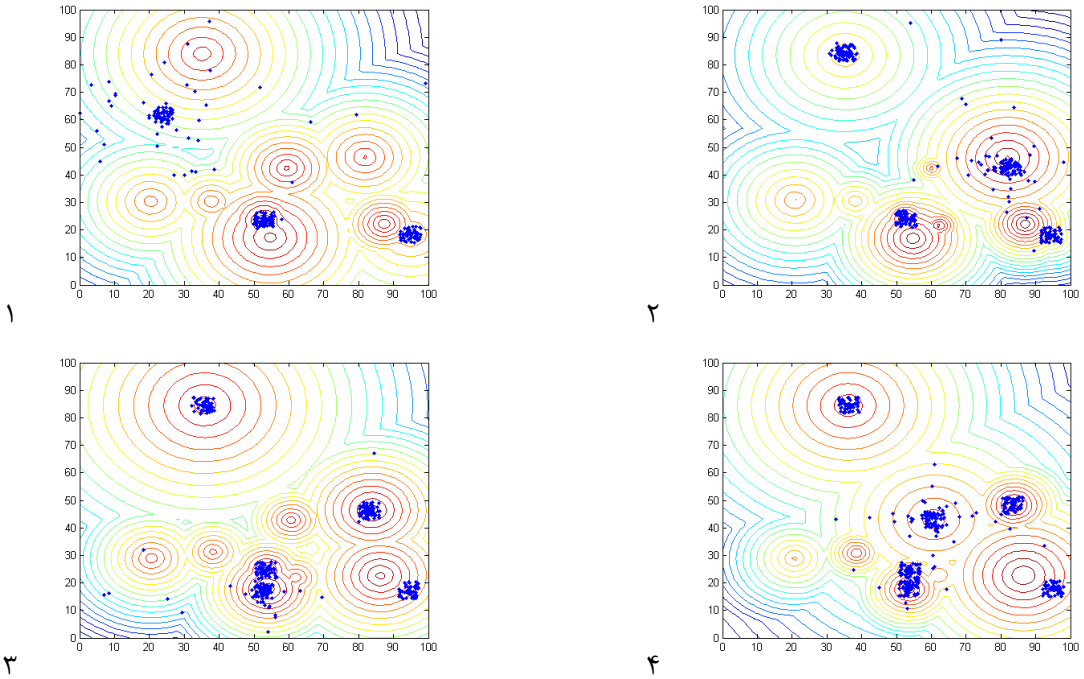
آزمایش‌های مختلف بر روی پارامترهای مختلفی برای الگوریتم انجام گرفته است که نتایج این آزمایش‌های در ادامه در جداول مختلف مورد ارزیابی و مقایسه با روش‌های دیگر قرار گرفته است. در آزمایش‌های انجام‌گرفته اندازه جمعیت به‌صورت ثابت و برابر ۱۰۰ است (این پارامتر قابل‌تغییر است) و مقایسه بر اساس میانگین‌گیری از ۳۰ بار اجرای مکرر الگوریتم انجام شده است. به‌طور مثال در فرکانس ۵۰۰ و با ۱۰ قله و ۳۰ بار اجرای مکرر الگوریتم مقدار میانگین خطای برون‌خطی برابر مقدار ۰/۸۹۵۸ است که پایین بودن این مقدار به‌منزله بهتر بودن الگوریتم از نظر سازگاری با محیط است. شکل ۱۱ روش پیشنهادی را تشریح می‌نماید. در این شکل همان‌گونه که مشخص است روش پیشنهادی از ۳ خوشه تشکیل شده است. در هر خوشه جمعیت اصلی و جمعیت حافظه قرار دارند. نقش حافظه در این شکل به‌خوبی نمایان است. در این شکل مشخص است که در صورت جابه‌جایی یک قله، حافظه‌ای که بالاترین کارایی را در آن خوشه داراست به‌سرعت قله جدید را ردیابی نموده و جمعیت را به سمت آن سوق می‌دهد. در شکل ۱۱ بهترین فرد از هر خوشه فردی است که به مرکز جدید قله نزدیک‌تر است. بهترین حافظه نیز حافظه‌ای است که به مرکز جدید قله نزدیک‌تر است. بدترین فرد از هر خوشه فردی است که از مرکز جدید قله بیش‌ترین فاصله را داشته باشد. به‌روزرسانی مناسب حافظه باعث می‌شود در هر زمان بهترین افراد در حافظه قرار گیرند و همین موضوع به افزایش سرعت همگرایی در الگوریتم کمک شایانی می‌نماید. در این روش بدترین فرد از هر خوشه مشخص شده و این فرد از خوشه با بهترین فرد حافظه جایگزین می‌شود. مزایای روش پیشنهادی نسبت به روش‌های مشابه در نحوه به‌روزرسانی حافظه و نحوه خوشه‌بندی استفاده شده است. در این روش علاوه بر خوشه‌بندی جمعیت اصلی، حافظه نیز خوشه‌بندی می‌شود. تعداد خوشه‌ها در حافظه و جمعیت اصلی برابرند. تبادل اطلاعات میان خوشه‌های متناظر باهم در جمعیت اصلی و حافظه صورت می‌گیرد. این روش خوشه‌بندی علاوه بر این‌که تنوع را به‌خوبی در الگوریتم حفظ می‌نماید، باعث بهبود جستجوی محلی نیز می‌شود.

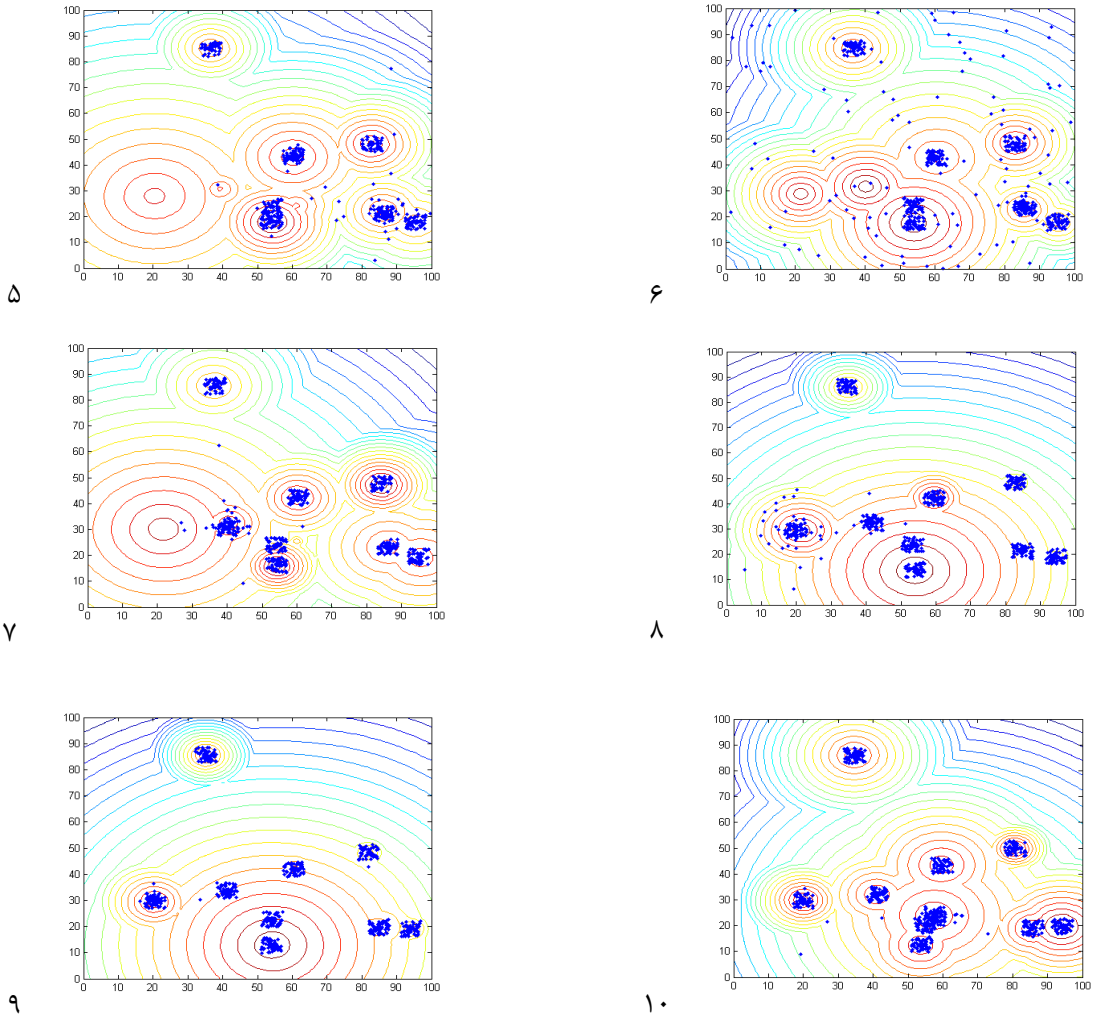
شکل ۱۲ روند همگرایی افراد جمعیت به مرکز قله‌ها را در روش پیشنهادی در ۱۰ مرحله نشان می‌دهد. همان‌گونه که در شکل ۱۲ دیده می‌شود روش پیشنهادی به‌سرعت به مراکز قله‌ها همگرا شده و نه‌تنها دارای سرعت همگرایی بالایی است بلکه درصد خیلی بالایی از

قله‌ها نیز توسط افراد جمعیت پوشش داده شده‌اند که این موضوع یکی از نقاط قوت الگوریتم‌های تکاملی در محیط‌های پویا است. شکل ۱۲ به‌وضوح مشخص است که درصد خیلی بالایی از قله‌ها توسط افراد جمعیت پوشش داده شده‌اند. این پوشش قله‌ها توسط افراد به دلیل تنوع ایجادشده توسط روش خوشه‌بندی مناسبی است که در این الگوریتم به کار گرفته شده است. همان‌طور که قبلاً نیز بیان شد در یک محیط پویا هدف تنها یافتن بهترین راه‌حل نیست، بلکه باید تمامی راه‌حل‌ها در طول اجرای الگوریتم ردیابی شوند. در یک محیط پویا یک راه‌حل که قبلاً بهینه محلی بوده ممکن است در صورت تغییر در محیط به بهینه سراسری تبدیل شود. بنابراین این پوشش تمامی قله‌ها در فضای مسئله این چالش اساسی را به‌خوبی حل نموده و این الگوریتم را به یک الگوریتم کارا و مناسب تبدیل نموده است.



شکل ۱۱: تشریح روش پیشنهادی با شکل مناسب





شکل ۱۲: نحوه همگرا شدن افراد جمعیت به مرکز قله‌ها در حین اجرای الگوریتم

۸-۱- بررسی پارامتر فرکانس تغییرات مختلف با تعداد قله‌های مختلف برای روش پیشنهادی

جداول ۳، ۴ و ۵ به ترتیب مقدار میانگین خطای برون‌خطی را برای روش پیشنهادی در فرکانس‌های تغییر ۵۰۰، ۱۰۰۰ و ۵۰۰۰ و با تعداد قله‌های مختلف نشان می‌دهند. درواقع روش پیشنهادی در فرکانس‌های تغییر پایین و فرکانس‌های تغییر بالا آزمایش شده است. در فرکانس‌های تغییر پایین محیط در تعداد تکرارهای کم‌تری دچار تغییر می‌شود و ردیابی بهینه تغییریافته را برای الگوریتم مشکل‌تر می‌نماید. در فرکانس‌های تغییر بالا الگوریتم‌ها زمان بیش‌تری برای جستجوی بهینه تغییریافته در اختیار دارند و درنتیجه ردیابی بهینه موردنظر برای الگوریتم راحت‌تر است. نتایج به‌دست‌آمده در جدول‌های ۳، ۴ و ۵ نشان می‌دهد روش پیشنهادی در فرکانس‌های تغییر مختلف و با تعداد قله‌های مختلف دارای کارایی مناسب‌تری نسبت به سایر روش‌ها است.

جدول ۳: میانگین خطای برون‌خطی با تعداد قله‌های مختلف و فرکانس ۵۰۰

تعداد قله‌ها	الگوریتم پیشنهادی	FMSO	mQSO	CPSO	rPSO	Cellular PSO
۱	۱/۱۱۵۰	۲۷/۵۸	۳۳/۶۷	۱۴/۲۵۲۱	۴/۲۷	۱۳/۴
۵	۰/۹۶۵۴	۱۹/۴۵	۱۱/۹۱	۳۶/۴۰۹۴	۱۶/۱۹	۹/۶۳
۱۰	۰/۸۹۵۸	۱۸/۲۶	۹/۶۲	۲۰/۹۱۲۹	۱۷/۳۴	۹/۴۲
۲۰	۰/۸۶۹۷	۱۷/۳۴	۸/۷۹	۱۳/۱۱۵۵	۱۷/۰۶	۸/۸۴
۳۰	۰/۸۸۱۸	۱۶/۳۹	۸/۸۰	۱۰/۸۳۰۷	۱۶/۹۸	۷/۸۸
۴۰	۰/۹۳۹۷	۱۵/۳۴	۸/۵۵	۱۰/۱۲۳۹	۱۶/۴۶	۷/۸۳
۵۰	۰/۸۴۲۹	۱۵/۵۴	۸/۵۶	۹/۲۸۷۱	۱۵/۷۷	۸/۶۲
۱۰۰	۰/۷۵۹۸	۱۲/۸۷	۸/۵۴	۲/۷۷۲۶	۱۴/۵۵	۱۱/۳۸
۲۰۰	۰/۷۴۴۵	۱۱/۵۲	۸/۱۹	۶/۸۳۳۳	۱۳/۴۰	۱۱/۳۴

جدول ۴: میانگین خطای برون‌خطی با تعداد قله‌های مختلف و فرکانس ۱۰۰۰

تعداد قله‌ها	الگوریتم پیشنهادی	FMSO	mQSO	CPSO	rPSO	Cellular PSO
۱	۰/۲۰۱۸	۱۴/۴۲	۱۸/۶	۸/۹۳۵۳	۱/۹۴	۶/۷۷
۵	۰/۲۳۶۵	۱۰/۵۹	۶/۵۶	۸/۶۲۲۲	۱۳/۷۷	۵/۳۰
۱۰	۰/۲۰۶۵	۱۰/۴۰	۵/۷۱	۷/۴۸۱۹	۱۵/۵۵	۵/۱۵
۲۰	۰/۱۹۴۶	۱۰/۳۳	۵/۹۰	۶/۱۰۳۱	۱۵/۵۴	۵/۳۲
۳۰	۰/۱۹۵۲	۱۰/۰۶	۶/۰۲	۵/۴۴۸۵	۱۴/۳۸	۵/۹۳
۴۰	۰/۱۹۴۷	۹/۸۵	۶/۲۲	۵/۵۷۱۲	۱۴/۱۱	۵/۹۵
۵۰	۰/۱۸۲۰	۹/۵۴	۵/۹۰	۵/۱۷۱۸	۱۳/۷۵	۵/۵۵
۱۰۰	۰/۱۹۴۳	۸/۷۷	۶/۴۸	۴/۲۶۲۲	۱۲/۲۷	۶/۲۷
۲۰۰	۰/۱۵۳۹	۸/۰۶	۶/۱۸	۳/۷۴۲۲	۱۱/۳۲	۶/۰۱

جدول ۵: میانگین خطای برون‌خطی با تعداد قله‌های مختلف و فرکانس ۵۰۰۰

تعداد قله‌ها	الگوریتم پیشنهادی	CPSOR	FMSO	mQSO	CPSO	rPSO	Cellular PSO
۱	۰/۰۱۲۳	۰/۰۳۵۶	۳/۴۴	۵/۰۷	۰/۱۴	۰/۵۶	۲/۵۵
۵	۰/۰۱۱۵	۰/۰۵۴۹	۲/۹۴	۱/۸۱	۰/۷۲	۱۲/۲۲	۱/۶۸
۱۰	۰/۰۷۹۶	۰/۵۹۹	۳/۱۱	۱/۷۵	۱/۵۶	۱۲/۹۸	۱/۷۸۱
۲۰	۰/۰۶۴۲	۰/۷۹۶	۳/۳۶	۲/۷۴	۱/۵۹	۱۲/۷۹	۲/۶۰
۳۰	۰/۰۵۸۲	۱/۰۵	۳/۲۸	۳/۲۷	۱/۵۸	۱۲/۳۵	۲/۹۳
۴۰	۰/۰۵۴۲	–	۳/۲۶	۳/۶۰	۱/۵۱	۱۱/۳۷	۳/۱۴
۵۰	۰/۰۶۵۹	۰/۹۸۶	۳/۲۲	۳/۶۵	۱/۵۴	۱۱/۳۴	۳/۲۶
۱۰۰	۰/۰۶۲۹	۱/۰۶	۳/۰۶	۳/۹۳	۱/۴۱	۹/۷۳	۳/۴۱
۲۰۰	۰/۰۵۶۹	۰/۹۴۹	۲/۸۴	۳/۸۶	۱/۲۴	۸/۹۰	۳/۴۰

۸-۲- بررسی پارامتر شدت تغییرات بر روی روش پیشنهادی در مقایسه با سایر روش‌ها

شدت تغییرات در قله‌ها بیانگر این است که طول تغییرات در قله‌ها به چه میزان باشد و درواقع هر چه شدت تغییرات در طول قله‌های در حال تغییر بیش‌تر باشد کار الگوریتم برای ردیابی پاسخ بهینه نیز مشکل‌تر می‌شود. نتایج حاصل از جدول ۶ کارایی مناسب روش پیشنهادی را با طول تغییرات مختلف در قله‌های در حال تغییر در مقایسه با سایر روش‌ها نشان می‌دهد.

جدول ۶: میانگین خطای برون‌خطی برای روش پیشنهادی با طول تغییرات مختلف برای قله‌ها

الگوریتم	طول تغییرات قله‌ها					
	۰	۱	۲	۳	۴	۵
روش پیشنهادی	۰/۱۰۴۷	۰/۰۷۹۶	۰/۰۷۶۰	۰/۰۶۴۲	۰/۰۵۴۱	۰/۰۵۳۱
CPSOR	۰/۴۱۸	۰/۵۹۹	۰/۸۴۹	۰/۹۶۴	۱/۳۸	۱/۶۹
CPSO	۰/۴۶۵	۰/۷۱۵	۰/۸۴۳	۰/۹۱۱	۰/۹۹۷	۱/۰۸
mQSO	۱/۱۸	۱/۷۵	۲/۴۰	۳/۰	۳/۵۹	۴/۲۴
rSPSO	۰/۷۴	۱/۵۰	۱/۸۷	۲/۴	۲/۹۰	۳/۲۵
ESCA	۱/۷۲	۱/۵۳	۱/۵۷	۱/۶۷	۱/۷۲	۱/۷۸
CESO	۰/۵۸	۱/۳۸	۱/۷۸	۲/۰۳	۲/۲۳	۲/۵۲

۸-۳- بررسی پارامتر اندازه حافظه بر روی روش پیشنهادی

اندازه حافظه برای ذخیره راه‌حل‌های مناسب یکی از مهم‌ترین پارامترهای روش پیشنهادی است. جدول ۷ روش پیشنهادی را با اندازه‌های حافظه مختلف و در فرکانس‌های تغییر مختلف مورد بررسی قرار داده و از خطای برون‌خطی به‌عنوان معیاری برای اندازه‌گیری کارایی روش پیشنهادی در اندازه‌های مختلف استفاده می‌کند. پارامترهای مورداستفاده در مسئله محک قله‌های متحرک همان پارامترهای پیش‌فرض موجود در جدول ۱ است. همان‌گونه که از جدول ۶ مشاهده می‌شود در روش پیشنهادی با افزایش اندازه حافظه و در فرکانس‌های تغییر پایین خطای برون‌خطی نیز افزایش یافته و کارایی برای روش پیشنهادی پایین می‌آید.

جدول ۷: میانگین خطای برون‌خطی برای روش پیشنهادی با اندازه‌های مختلف حافظه و در فرکانس‌های تغییرات مختلف

فرکانس تغییرات	اندازه حافظه					
	۱۰	۲۰	۳۰	۴۰	۵۰	۱۰۰
۱۰۰	۴/۰۲۳	۶/۵۱۰	۱۰/۰۲۳	۱۴/۲۳	۱۶/۱۷	۲۰/۱۳
۲۰۰	۴/۰۱۹	۶/۳۱۰	۹/۷۱۱	۱۴/۱۰	۱۵/۹۶	۱۹/۲۰
۵۰۰	۲/۶۰۲	۴/۱۲۹	۶/۲۰۸	۸/۷۰۹	۸/۹۰۹	۱۲/۱۶
۲۵۰۰	۱/۰۴۱۲	۱/۹۰۶	۳/۱۲۴	۵/۸۰۱	۷/۰۰۲	۹/۶۲۰
۵۰۰۰	۰/۰۷۹۶	۰/۱۲۹۶	۰/۷۷۱۰	۱/۰۶۱	۲/۱۲۰	۶/۷۰۸
۱۰۰۰۰	۰/۰۵۱۸	۰/۱۶۴۴	۰/۵۰۴۲	۰/۹۲۸۱	۱/۱۰۶۸	۳/۰۴۰۴

۸-۴- بررسی تأثیر پارامتر تعداد خوشه‌ها در روش پیشنهادی

جدول ۷ روش پیشنهادی را با تعداد خوشه‌های مختلف و با تعداد قله‌های مختلف مورد بررسی قرار می‌دهد. دیگر پارامترهای مورداستفاده در مسئله محک قله‌های متحرک همان پارامترهای پیش‌فرض موجود در جدول ۱ است. همان‌گونه که در بخش‌های گذشته بیان شد، تعداد خوشه‌ها در حافظه و جمعیت اصلی برابر می‌باشند و نتایج حاصل از جدول ۸ نشان می‌دهد که افزایش تعداد خوشه‌ها و افزایش تعداد قله‌ها منجر به کاهش کارایی الگوریتم پیشنهادی می‌گردد. با افزایش تعداد خوشه‌ها تنوع در میان خوشه‌ها کم‌تر شده و درنتیجه سرعت همگرایی نیز کاهش می‌یابد.

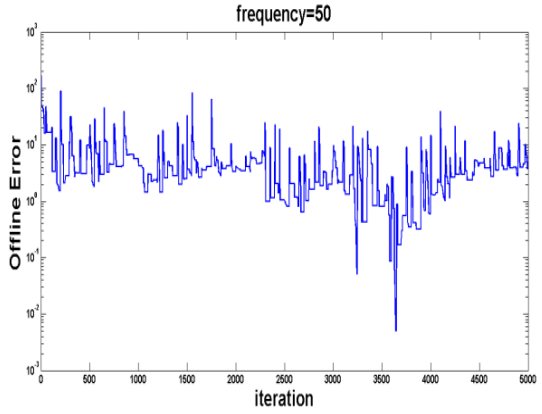
نتایج حاصل از جدول ۸ نشان می‌دهد بهترین پارامتری که می‌توان به‌عنوان پارامتر تعداد خوشه‌ها برای این روش در نظر گرفت تعداد ۲ خوشه است. با افزایش تعداد قله‌ها الگوریتم باید در میان تعداد قله‌های بیشتری به دنبال بزرگ‌ترین قله (بهینه سراسری) بگردد. افزایش تعداد قله‌ها کار الگوریتم‌ها را در ردیابی بهینه سراسری مشکل‌تر می‌نماید.

جدول ۸: میانگین خطای برون‌خطی برای روش پیشنهادی با تعداد خوشه‌های مختلف

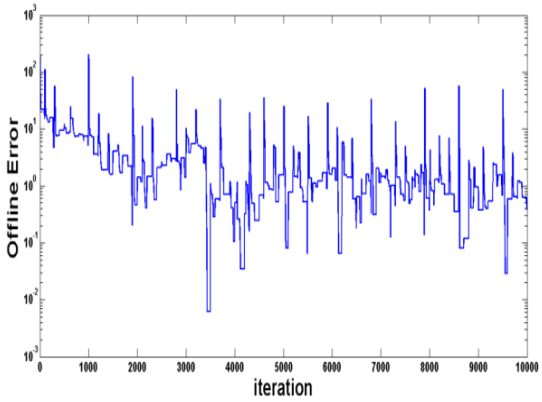
تعداد قله‌ها	تعداد خوشه‌ها					
	۲	۴	۶	۸	۱۰	۲۰
۱	۰/۰۱۲۳	۰/۹۱۳	۱/۲۲۳	۲/۷۲۳	۴/۰۲۷	۶/۰۵۳
۱۰	۰/۰۷۹۶	۱/۷۴۷	۲/۲۷۱	۴/۱۷۰	۵/۰۹۶	۷/۴۲۴
۲۰	۰/۰۶۴۲	۱/۹۶۹	۳/۰۲۸	۴/۶۶۹	۶/۱۲۹	۸/۰۴۰
۳۰	۰/۰۵۸۲	۱/۹۲۸	۳/۱۵۶	۵/۹۶۹	۷/۵۰۲	۹/۱۶۵
۵۰	۰/۰۶۵۹	۱/۷۲۷	۳/۴۷۹	۵/۸۱۰	۷/۸۲۰	۹/۷۲۹
۱۰۰	۰/۰۶۲۹	۱/۶۵۴	۳/۵۲۹	۵/۶۲۱	۷/۱۵۹	۹/۰۴۹

۸-۵- روند تولید خطای برون‌خطی برای روش پیشنهادی

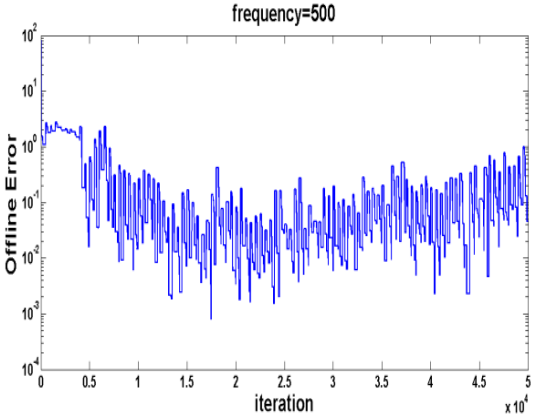
نمودارهای ترسیم‌شده در شکل ۱۳، ۱۴، ۱۵، ۱۶، ۱۷ و ۱۸ به ترتیب روند تولید خطای برون‌خطی را برای روش پیشنهادی در فرکانس‌های تغییرات ۵۰، ۱۰۰، ۵۰۰، ۱۰۰۰، ۵۰۰۰، ۱۰۰۰۰ و با تعداد ۱۰ قله نشان می‌دهند. دیگر پارامترهای استفاده‌شده همان پارامترهای استاندارد سناریوی دوم برانک می‌باشند. شکل ۱۳ نشان‌دهنده تعداد تغییرات برای ۵۰۰۰ تولید نسل با فرکانس تغییر ۵۰ است، به ازای هر ۵۰ نسل تولیدشده یک تغییر رخ می‌دهد. شکل ۱۴ نشان‌دهنده تعداد تغییرات برای ۱۰۰۰۰ تولید نسل و با فرکانس تغییر ۱۰۰ است، به ازای هر ۱۰۰ نسل تولیدشده یک تغییر دیده می‌شود. شکل ۱۵ نشان‌دهنده تعداد تغییرات برای ۵۰۰۰۰ تولید نسل و با فرکانس تغیJیر ۵۰۰ است، به ازای هر ۵۰۰ نسل تولیدشده یک تغییر دیده می‌شود. شکل ۱۶ نشان‌دهنده تعداد تغییرات برای ۱۰۰۰۰۰ تولید نسل و با فرکانس تغییر ۱۰۰۰ است، به ازای هر ۱۰۰۰ نسل تولیدشده یک تغییر دیده می‌شود. شکل‌های ۱۷ و ۱۸ نشان‌دهنده تغییرات برای ۵۰۰۰۰ و ۱۰۰۰۰۰ تولید نسل و با فرکانس ۵۰۰۰ و ۱۰۰۰۰ و تعداد ۴۰ تغییر در محیط است.



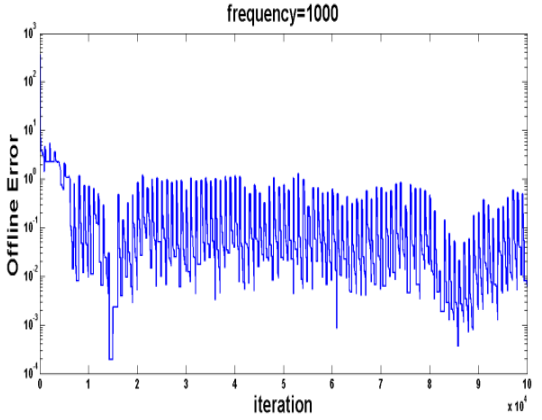
شکل ۱۳: روند تولید خطای برون‌خطی در فرکانس تغییرات ۵۰ و با ۱۰ قله



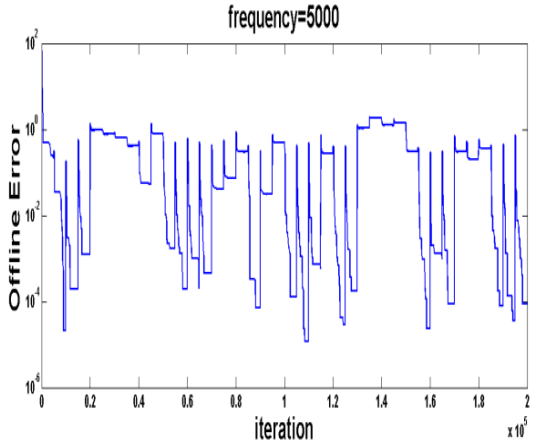
شکل ۱۴: روند تولید خطای برون‌خطی در فرکانس تغییرات ۱۰۰ و با ۱۰ قله



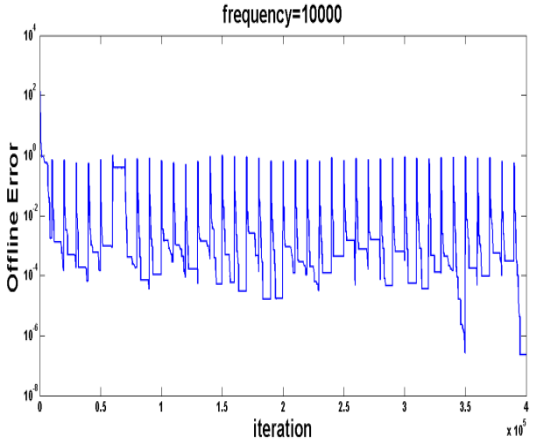
شکل ۱۵: روند تولید خطای برون‌خطی در فرکانس تغییرات ۵۰۰ و با ۱۰ قله



شکل ۱۶: روند تولید خطای برون‌خطی در روش پیشنهادی در فرکانس ۱۰۰۰ و با ۱۰ قله



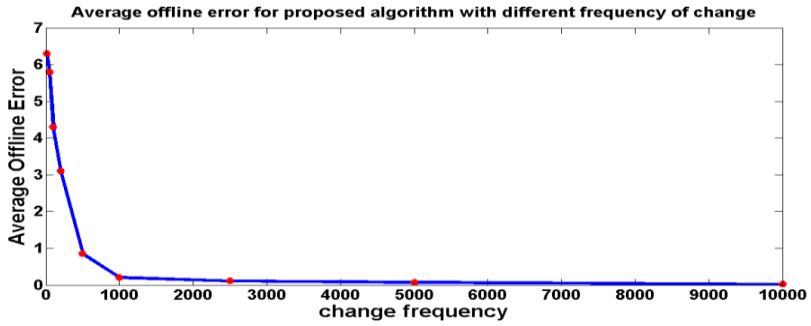
شکل ۱۷: روند تولید خطای برون‌خطی در روش پیشنهادی در فرکانس ۵۰۰۰ و با ۱۰ قله (۴۰ تغییر)



شکل ۱۸: روند تولید خطای برون‌خطی در روش پیشنهادی در فرکانس ۱۰۰۰۰ و با ۱۰ قله (۴۰ تغییر)

۸-۶- بررسی روش پیشنهادی در فرکانس‌های تغییر مختلف و با ۱۰ قله

نمودار ترسیم‌شده در شکل ۱۹ میانگین خطای برون‌خطی را برای روش پیشنهادی در فرکانس‌های تغییر مختلف و با ۱۰ قله و دیگر پارامترهای استاندارد ذکرشده در جداول ۱ و ۲ نشان می‌دهد. شکل ۱۹ بیانگر این موضوع است که روش پیشنهادی در فرکانس‌های مختلف دارای کارایی مناسبی بوده و قابلیت سازگاری با تغییرات مختلف محیطی را دارا است.



شکل ۱۹: میانگین خطای برون‌خطی در روش پیشنهادی در فرکانس‌های تغییرات مختلف با ۱۰ قله

۸-۷- مقایسه روش پیشنهادی با روش‌های مبتنی بر حافظه Memory/Search و SEAm و RIm

جدول ۷ روش پیشنهادی را در شرایط برابر و با پارامترهای سناریوی دوم برانک برای مسئله محک قله‌های متحرک با روش‌های مبتنی بر حافظه Memory/Search و SEAm [۱۷] [۱۹] و RIm [۱۰] [۲۰] مقایسه می‌نماید.

جدول۷: میانگین خطای برون‌خطی برای روش پیشنهادی و سایر روش‌ها

الگوریتم	خطای برون‌خطی
روش پیشنهادی	۰/۰۷۹۶
Memory/Search	۷/۷۶
SEAm	۱۰/۳۸
RIm	۱۱/۱۸

۸-۸- بررسی تأثیر پارامتر ضریب آشوب بر روی روش پیشنهادی

جدول ۸ تأثیر پارامتر ضریب آشوب بر روی کارایی روش پیشنهادی را نشان می‌دهد. پارامتر A ضریب آشوب است. تغییر در این پارامتر می‌تواند نتایج متفاوتی برای کارایی روش پیشنهادی تولید نماید. پس این پارامتر یکی از پارامترهای مهم در این روش محسوب می‌شود.

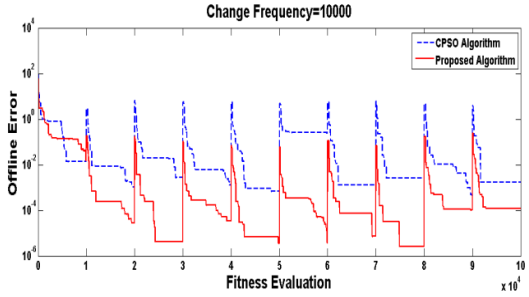
جدول۸: تأثیر پارامتر ضریب آشوب بر روی کارایی روش پیشنهادی

ضریب آشوب (A)	میانگین خطای برون‌خطی
۱	۰/۲۷۱۰
۲	۰/۲۰۹۰
۳	۰/۱۲۲۳
۴	۰/۰۷۹۶

نتایج حاصل از جدول ۸ نشان می‌دهد بهترین مقدار برای پارامتر ضریب آشوب در این روش پیشنهادی مقدار ۴ است.

۸-۹- مقایسه روش پیشنهادی با روش CPSO

در این بخش روش پیشنهادشده را با یکی از روش‌های مشابه که از خوشه‌بندی استفاده نموده مقایسه شده است و نتایج حاصله نشان از برتری روش پیشنهادی نسبت به روش مشابه دارد. شکل ۲۰ مقایسه بین روش پیشنهادی و روش CPSO [۵] را ازنظر خطای برون‌خطی در هر ارزیابی نشان می‌دهد (شرایط مسئله برای دو روش کاملاً برابر در نظر گرفته شده است). روش CPSO یکی از معروف‌ترین روش‌هایی است که در سال ۲۰۱۰ توسط آقای یانگ معرفی شده و اکثر محققین از این روش به‌عنوان مرجع استفاده می‌کنند. این روش به دلیل شباهتی که به روش پیشنهادی دارد (هر دو روش از خوشه‌بندی استفاده نموده‌اند) می‌تواند مرجع مناسبی برای مقایسه با روش پیشنهادی در این مقاله باشد.



شکل ۲۰: مقایسه روش پیشنهادی با روش مشابه CPSO در شرایط برابر (فرکانس ۱۰۰۰۰، تعداد ۱۰ قله و طول تغییرات ۱)

۸-۱۰- مقایسه روش پیشنهادی با دیگر روش‌ها ازنظر پیچیدگی محاسباتی

جدول ۹ روش پیشنهادی را ازنظر پیچیدگی محاسباتی با دو روش CPSO، CPSOR و mQSO مقایسه می‌نماید (M اندازه جمعیت، ite تعداد محاسبه کارایی و cf فرکانس تغییرات محیط است).

جدول۹: مقایسه روش پیشنهادی با دیگر روش‌ها ازنظر پیچیدگی محاسباتی

الگوریتم	پیچیدگی محاسباتی
روش پیشنهادی	$O(M \log M)$
CPSO	$O(M^3)$
CPSOR	$O(M^3)$
mQSO	$O(M^2)$

۹- نتیجه‌گیری

در این مقاله یک الگوریتم جدید برای بهینه‌یابی محیط پویا بر پایه ترکیب الگوریتم ژنتیک آشوب‌گونه با حافظه صریح و خوشه‌بندی پیشنهاد گردید. الگوریتم پیشنهادی دارای افزایش سرعت همگرایی برای اکثر پارامترهای مسئله است. با استفاده از نظریه آشوب برای ایجاد جمعیت اولیه، سرعت همگرایی افزایش پیدا کرده است. همچنین با استفاده از حافظه صریح برای حفظ راه‌حل‌های مناسب گذشته و یک راه‌کار مناسب

برای به‌روزرسانی حافظه، بعد از تغییر در محیط بر چالش حافظه نامعتبر افراد در ردیابی بهینه جدید غلبه شده است. خوشه‌بندی جمعیت در حافظه و جمعیت اصلی هم تنوع را در حین اجرای الگوریتم افزایش داده و هم جستجوی محلی را برای الگوریتم بهبود داده است. تنوع ایجادشده باعث می‌شود پراکندگی جمعیت در فضای مسئله حفظ شود. هر فرد از جمعیت، جستجوی محلی را در خوشه مربوط به خود با سرعت بالایی انجام می‌دهد. در روش پیشنهادی دو جنبه نوآوری اساسی پیشنهاد گردید: یکی خوشه‌بندی استفاده‌شده که دو جمعیت اصلی و حافظه را خوشه‌بندی نموده و با ایجاد تناظر میان خوشه‌های دو جمعیت تنوع را افزایش داده و جستجوی محلی را نیز بهبود داده است. دومین جنبه نوآوری اساسی در این روش نحوه به‌روزرسانی حافظه است که باعث می‌شود در هر زمان مفیدترین اطلاعات در حافظه قرار گرفته و زمانی که محیط تغییر می‌نماید از این اطلاعات در محیط جدید استفاده شود. در مقایسه با الگوریتم ژنتیک استاندارد به‌صورت منفرد و روش‌های مشابه با روش پیشنهادی، این الگوریتم کارایی قابل قبولی از خود نشان داده است، ولی هنوز هم امکان افزایش سرعت همگرایی وجود دارد. ترکیب روش پیشنهادی با الگوریتم‌های دیگر ازجمله روش جستجوی تپه‌نوردی برای جستجوی محلی در هر خوشه می‌تواند گزینه مناسبی برای کارهای آتی باشد.

مراجع

[1] S. Yang and C. Li, "A Clustering Particle Swarm Optimizer for Locating and Tracking Multiple Optima in Dynamic Environments," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 6, pp. 959–974, 2010.

[2] L. Liu, S. Yang and D. Wang, "Particle Swarm Optimization With Composite Particles in Dynamic Environments," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 40, no. 6, pp. 1634–1648, 2010.

[3] S. Yang, "Explicit memory schemes for evolutionary algorithms in dynamic environments," *Studies in Computational Intelligence*, vol. 51, pp. 3–28, 2007.

[4] S. Yang, "Genetic algorithms with elitism-based immigrants for changing optimization problems," *Lecture Notes in Computer Science 4448*, pp. 627–636, 2007.

[5] S. Yang and C. Li, "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 6, pp. 959–974, 2010.

[6] J. Branke, T. Kaußler, C. Schmidt and H. Schmeck, "A multi population approach to dynamic optimization problems," in *Adaptive Computing in Design and Manufacturing*, pp. 299–308, 2000.

[7] T. Blackwell and J. Branke, "Multi-Swarms, Exclusion, and Anti-Convergence in Dynamic Environments," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 459–472, 2006.

[8] T. Blackwell, J. Branke and X. Li, "Particle swarms for dynamic optimization problems," *Swarm Intelligence*, pp. 193–217, 2008.

[9] J. Branke, *Evolutionary Optimization in Dynamic Environment*, Kluwer Academic Publishers Norwell, 2002.

[10] J. J. Grefenstette and C. L. Ramsey, "An approach to anytime learning," in *International Conference on Machine Learning*, pp. 189–195, 1992.

[11] S. Yang and C. Li, "A clustering particle swarm optimizer for dynamic optimization," in *IEEE Congress on Evolutionary Computation*, pp. 439–446, 2009.

[12] S. Yang, "Memory-based immigrants for genetic algorithms in dynamic environments," in *Seventh International Genetic and Evolutionary Computation Conference (GECCO)*, vol. 2, pp. 1115–1122. 2005.

[13] S. Yang and C. Li, "Fast Multi-Swarm Optimization for Dynamic Optimization Problems," in *Fourth International Conference on Natural Computation*, pp. 624–628, 2008.

[۱۴] رحمت‌اله هوشمند، حسین محکمی، امین خدابخشیان، «روشی جدید در جایابی بهینه خازن‌ها و ژنراتورهای توزیع‌شده در شبکه-های توزیع با استفاده از الگوریتم جستجوی باکتریای جهت داده‌شده با pso»، مجله مهندسی برق دانشگاه تبریز. جلد ۳۹. شماره ۲. ۱۳۸۹.

[۱۵] الهام شکرانی‌پور، مسعود افتخاری‌مقدم، «یک الگوریتم جدید بهینه‌سازی گروه ذرات تعاونی با قابلیت به‌روزرسانی تطبیقی پارامترها»، مجله مهندسی برق دانشگاه تبریز. جلد ۴۰. شماره ۲. زمستان ۱۳۸۹.

[16] R. Morrison and K. DeJong. "A test problem generator for non-stationary environments," *In Congress on Evolutionary Computation*, pp. 2047–2053, 1999.

[17] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," *In Congress on Evolutionary Computation*, pp. 1875–1882, 1999.

[18] J. Holland, "Adaptation in Natural and Artificial Systems," University of Michigan Press, Ann Arbor, MI, 1975.

[19] N. Krasnogor and j. Smith "A Tutorial for Competent Memetic Algorithms: Model, Taxonomy, and Design Issues," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 474–488, 2005.

[20] C. Ryan, "Diploidy without dominance," *In Nordic Workshop on Genetic Algorithms*, pp. 45–52, 1997.

[21] S. Yang, "Genetic algorithms with elitism-based immigrants for changing optimization problems," *In Applications of Evolutionary Computing, Lecture Notes in Computer Science 4448*, pp. 627–636, 2007.

[22] C. Ramsey and J. Grefenstette, "Case-based initialization of genetic algorithms," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 84–91, 1993.

[23] K. Trojanowski and Z. Michalewicz, "Searching for optima in non-stationary environments," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999)*, pp. 1843–1850, 1999.

[24] A. Simoes and E. Costa, "Improving memory's usage in evolutionary algorithms for changing environments," in *IEEE congress on evolutionary computation*, pp. 276–283, 2007.

[25] D. Parrott and X. Li, "Locating and Tracking Multiple Dynamic Optima by A Particle Swarm Model Using peciation," *IEEE Transaction on Evolutionary Computation*, vol. 10, no. 4, pp. 440–458, 2006.

[26] B. Hashemi and M. R. Meybodi, "Cellular PSO: A PSO for Dynamic Environments", in *Advances in Computation and ntelligence, Lecture Notes in Computer Science*, vol. 5821, pp. 422–433, 2009.

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

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

[29] G. J. Barlow, *Improving memory for optimization and learning in dynamic environments*, Ph.D. Thesis, Carnegie Mellon University, pp. 65–82, 2011.

زیرنویس‌ها

[1] Diversity
[2] Elitism-based Immigrants GA
[3] Elitism
[4] Selection
[5] Crossover
[6] Memory Immigrant Genetic Algorithm
[7] Overlap degree
[8] Quantum particles
[9] Random Immigrants
[10] Memory Enhanced GA
[11] Adaptive Particle Swarm Optimization
[12] Cooprative Particle Swarm Optimization
[13] Moving Peaks Benchmark
[14] Chaos theory
[15] Explicit memory
[16] Implicit memories
[17] K-means
[۱۸] Moore
[19] Standard Evolutionary Algorithm with Memory
[20] Random Immigrants memory