علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۱۱، شماره ۳ (الف)، ۱۳۹۲
صفحات ۱۰-۲۱
مقاله عادی

حل مسئله قله‌های متحرک با استفاده از یک الگوریتم ممتیکی مبتنی بر الگوریتم بهینه‌سازی ازدحام ذرات فازی

مرتضی علی‌زاده[۱] محمدرضا میبدی[۲] علیرضا رضوانیان[۲]

[۱]دانشکده مهندسی برق، رایانه و فناوری اطلاعات، دانشگاه آزاد اسلامی واحد قزوین، قزوین، ایران
[۲]دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)، تهران، ایران

چکیده

بسیاری ازمسائل دنیای واقعی به صورت یک مسئله بهینه‌سازی با ماهیتی پویا هستند، به طوری که موقعیت و مقدار بهینه‌ی سراسری آنها در طول زمان تغییر می‌کنند. بنابراین در چنین مسائلی لازم است، ابتدا تغییرات محیط پویا را شناسایی کرده، سپس راه‌کارهایی برای بهینه‌سازی در محیط پویا ارائه نمود. یکی از مسائل معروف در بهینه‌سازی محیط‌های پویا، مسئله محک قله‌های متحرک می‌باشد، که رفتاری شبیه به مسائل پویا در دنیای واقعی را شبیه‌سازی می‌کند. در این مقاله یک رویکرد ممتیکی مبتنی بر بهینه‌سازی ازدحام ذرات فازی برای بهینه‌سازی محیط‌های پویا ارائه شده است. با توجه به آنکه قله‌ها در حالت پویا جابجا شده و تغییر پیدا می‌کنند، بنابراین ایده‌ی اصلی روش پیشنهادی به این صورت است که، پس از شناسایی تغییرات در محیط، الگوریتم قابلیت جستجوی محدوده‌ی نزدیک به قله‌های قبلی را با رویکرد ممتیکی دنبال می‌کند. در این حالت، نحوه طراحی رویکرد ممتیکی دارای اهمیت ویژه‌ای می‌شود که در این مقاله از الگوریتم بهینه‌سازی ازدحام ذرات فازی استفاده شده است. برای ارزیابی روش پیشنهادی، الگوریتم بهینه‌سازی ازدحام ذرات فازی بر روی تابع محک معروف قله‌های متحرک در سناریوهای مختلفی آزمایش شده و با چندین الگوریتم مهم دیگر مقایسه شده است، نتایج شبیه‌سازی آزمایشات حاکی از موفقیت رویکرد ممتیکی نسبت به الگوریتم‌های معروف دیگر می‌باشد.

کلمات کلیدی: بهینه‌سازی، محیط پویا، قله‌های متحرک، ممتیک، بهینه‌سازی ازدحام ذرات فازی.

۱- مقدمه

بهینه‌سازی بدین معنی است که در بین پارامترهای یک تابع به دنبال مقادیری باشیم که مقدار تابع را بیشینه و یا کمینه نمایند. 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. در واقع در این نوع از الگوریتم‌ها پاسخ‌های مناسب در یک بازه زمانی قابل قبولی بدست خواهند آمد، اما هیچ وقت تضمینی جهت بدست آمدن بهترین پاسخ وجود ندارد. در میان روش‌های جستجوی تصادفی، الگوریتم‌های تکاملی که از طبیعت الهام گرفته‌اند، بیشتر مورد استقبال محققان قرار گرفته است و نتایج موفقی نیز برای حل مسائل توسط محققان ارائه شده است. با توجه به پیچیدگی و تنوع مسائل بهینه‌سازی توسعه‌هایی نیز بر روی این الگوریتم‌ها برای بهبود کارایی و حل مسائل مختلف انجام شده است. از بین روش‌هایی که برای برطرف ساختن مشکلات الگوریتم‌های تکاملی ارائه شده است، استراتژی ترکیبی[۱] جایگاه ویژه‌ای دارد [۱، ۲]. استراتژی ترکیبی از به کارگیری تکنیک‌های مختلف در فرایند حل مساله حاصل می‌گردد. الگوریتم‌های ممتیکی از مشهورترین اعضاء این خانواده به شمار می‌آیند که از ترکیب الگوریتم‌های تکاملی با یک جستجوی محلی مکاشفه‌ای[۲] (نظیر روش تپه‌نوردی[۳]) حاصل می‌گردد [۳، ۴]. الگوریتم ممتیک که

توسط بالدوین[۴] و لامارک[۵] [۵] به عنوان نظریه‌پردازان اولیه بیان شده و توسط موسکاتو[۶] در سال ۱۹۸۹ بطور کامل ارائه شده است. با توجه به ماهیت مسائل مختلف، نحوه‌ی طراحی یک الگوریتم ممتیکی حائز اهمیت می‌شود. الگوریتم ممتیک[۷] گونه‌ای از الگوریتم‌های تکاملی[۸] است که در آن جستجوهای ابتکاری محلی با الگوریتم ژنتیک ترکیب می‌شوند تا در زمان کمتری نتایج بهتری به دست آید. این الگوریتم بر این ایده استوار است که هر عضو جمعیت می‌تواند به اندازه اطرافیان خود (همسایگان)، شایستگی[۹] خود را افزایش دهد [۱].

در مسائل بهینه‌سازی پویا، الگوریتم‌های تکاملی به سادگی قابل استفاده نیستند و کارایی لازم را مانند محیط‌های ایستا ندارند. از مشکلات اصلی این الگوریتم‌ها در محیط‌های پویا، می‌توان به حافظه غیرمعتبر و از دست رفتن تنوع اشاره نمود. همچنین از آنجا که اکثر روش‌های پردازش تکاملی به‌دلیل ماهیتشان به یک نقطه همگرا می‌شوند، لذا تنوع در محیط از بین می‌رود و در صورت تغییر در محیط همگرا شدن به نقطه بهینه جدید در صورت امکان بسیار زمان‌گیر است. ساده‌ترین راه‌حل برای حل این مشکل (تغییرات محیط)، در نظر گرفتن هر تغییر به عنوان ورودی یک مساله بهینه‌سازی جدید می‌باشد که باید از ابتدا حل شود، اما این روش نیز کارایی مناسبی ندارد. در صورت داشتن زمان کافی این یک گزینه مناسب می‌باشد. اما در بیشتر مواقع زمان موجود برای بهینه‌سازی مجدد نسبتاً کوتاه می‌باشد. یک تلاش طبیعی برای تسریع فرایند بهینه‌سازی پس از یک تغییر، استفاده از اطلاعات مربوط به فضای جستجوی قبلی برای پیشروی در جستجو پس از تغییر می‌باشد.

نکته اساسی دیگر این است که وقتی یک الگوریتم به یک نقطه همگرا می‌شود، تنوع خود را از دست می‌دهد که این امر باعث کاهش قابلیت انطباق و سازگاری با تغییر محیط می‌شود. بنابراین در کنار انتقال اطلاعات بین عامل‌های الگوریتم‌های بهینه‌سازی از قبل و بعد از تغییر محیط، باید به دنبال راهکارهایی برای افزایش تنوع و سازگاری الگوریتم پس از تغییر محیط بود.

در بین کارهای انجام شده برای بهینه‌سازی محیط‌های پویا می‌توان به تلاش‌های فراوان یانگ و شاگردانش اشاره نمود که تحقیقات گسترده‌ای بر روی الگوریتم‌های ممتیکی و محیط‌های پویا انجام داده‌اند [۶] [۷] [۸] استفاده از ترکیبی از الگوریتم‌های ممتیک و الگوریتم‌های تپه نوردی و ازدحام ذرات از کارهای اخیرشان به حساب می‌آید. در این روش نسخه‌ای محلی از بهینه‌سازی ازدحام ذرات که از توپولوژی حلقه با شعاع همسایگی یک، جهت اشتراک اطلاعات میان ذرات بهره می‌برد، به عنوان عملگر جستجوی سراسری مورد استفاده قرار می‌گیرد و از یک روش جستجوی محلی ادراکی فازی بعنوان تکنیک جستجوی محلی استفاده می‌شود. نسخه‌ی محلی الگوریتم بهینه‌سازی ازدحام ذرات قادر است فشار همگرایی جمعیت ذرات را (حداقل بطور موقت) کاهش دهد و ظرفیت کاوش کافی جهت یافتن نقاط مطلوب را به مدت بیشتری برای الگوریتم حفظ کند. آن‌ها به منظور افزایش قابلیت کاوش نواحی جدید توسط ذرات از مکانیزم مهاجرت تصادفی خود سازمانده[۱۰] استفاده می‌کنند. همچنین از ترکیب الگوریتم‌های ترکیبی با استفاده از اتوماتای سلولی[۱۱] و الگوریتم‌های هوش جمعی با رویکرد ممتیکی [۹] در محیط‌های ایستا و پویا نتایج قابل قبولی را ارائه نموده‌اند [۱۰].

لانگ و دومیترسکو یک رویکرد ترکیبی نوین به منظور بهینه‌سازی در محیط‌های پویا را با عنوان CESO ارائه کرده‌اند [۱۱]. این الگوریتم از دو جمعیت با اندازه‌ی یکسان به منظور پیشبرد فرآیند جستجو استفاده می‌کند. یکی از این دو جمعیت (که از یک الگوریتم تکاملی به نام تفاضل تکاملی[۱۲] مبتنی بر ازدحام استفاده می‌کند) مسئول حفظ پراکندگی ذرات در زمان جستجو می‌باشد و جمعیت دیگر (که از بهینه‌سازی گروه ذرات بهره می‌گیرد) به منظور تعقیب بهینه‌ی سراسری مورد استفاده قرار می‌گیرد. CESO مکانیزمی همکارانه را به منظور جلوگیری از هم‌گرایی پیش از موعد و تعقیب کارآمد بهینه‌ی مسئله، میان دو جمعیت ارائه می‌کند. همچنین همین نویسندگان، کار مشابه دیگری را در [۱۲] گزارش نمودند که در این الگوریتم، جمعیت دیگری را به این روش افزوده‌اند و سه مکانیزم همکارانه‌ی مختلف را به زیرگروه‌های ذرات افزوده‌اند. نتایج بدست آمده نشان دهنده‌ی آن هستند که این روش در مقابل تغییرات شدید در چشم‌انداز شایستگی محیط، عکس‌العمل بهتری را نسبت به رویکرد پیشین از خود نشان می‌دهد.

لی و دیگران [۱۳]، بهینه‌سازی گروه ذرات مرکب[۱۳] را معرفی کرده‌اند که در آن به‌منظور هدایت فرآیند بهینه‌سازی از ذرات جدیدی به‌نام ذرات مرکب استفاده شده است. این ذرات قادرند تا پس از رخداد تغییر در محیط بصورتی دقیق‌تر به اکتشاف فضای جستجو بپردازند. در تحقیق دیگری [۱۴]، آنها از ذرات مرکب و یک عملگر پخش‌کننده برای حفظ پراکندگی ذرات استفاده می‌کنند.

هو و ابرهارت تعدادی از مکانیزم‌های پراکندن مجدد را مورد مطالعه قرار داده‌اند [۱۵]. تمامی این مکانیزم‌ها شامل تصادفی‌سازی کل یا بخشی از گروه ذرات، پس از رخداد تغییر در محیط می‌باشند.

بلکول و بنتلی بهینه‌سازی گروه ذرات باردار[۱۴] را معرفی کرده‌اند [۱۶]، که در آن تعدادی از ذرات باردار به دور هسته‌ای از ذرات خنثی در حال گردش هستند. این ذرات باردار بطور متقابل یکدیگر را دفع می‌کنند و از این طریق باعث حفظ پراکندگی ذرات در هنگام فرآیند بهینه‌سازی می‌شوند. علاوه بر این بلکول و دیگران ایده‌ی ذرات باردار را گسترش داده‌اند و با ایجاد یک مدل کوانتومی یک روش چندگروهی را ارائه نموده‌اند [۱۷].

هاشمی و میبدی مدلی ترکیبی از بهینه‌سازی گروه ذرات و اتوماتای سلولی ارائه کرده‌اند که بهینه‌سازی ازدحام ذرات سلولی[۱۵] نام دارد [۱۸]. در این مدل آنها با استفاده از یک اتوماتای سلولی فضای جستجو را به تعدادی سلول افراز می‌کنند و در هر زمان، گروهی از ذرات در برخی از سلول‌ها به‌جستجوی نقاط بهینه‌ی محلی می‌پردازند. به منظور حفظ پراکندگی ذرات آنها تعداد ذرات مجاز در هر سلول را به یک حد آستانه محدود می‌کنند. علاوه بر این در مرجع [۱۹] به منظور تعقیب توالی تغییرات بهینه‌ی مسئله، پس از شناسایی رخداد تغییر در محیط، ذرات در اطراف نقاط بهینه‌ای که به‌تازگی یافت شده‌اند نقش خود را به‌ذرات کوانتومی تغییر می‌دهند و برای چند تکرار به جستجوی تصادفی در حوالی این نقاط می‌پردازند.

لی و یانگ یک روش چندگروهی سریع را معرفی کرده‌اند که پراکندگی ذرات را در حین اجرا حفظ می‌کند [۲۰]. در این روش گروهی از ذرات والد میزان پراکندگی ذرات را حفظ می‌نمایند و نواحی نوید بخش فضای جستجو را با استفاده از الگوریتم برنامه‌ریزی تکاملی سریع شناسایی می‌کنند و دسته‌ای از گروه‌های ذرات فرزند ناحیه‌ای محلی را با استفاده از الگوریتم بهینه‌سازی گروه ذرات سریع[۱۶] با هدف یافتن بهینه‌ی مسئله جستجو می‌کنند. علاوه بر این آنها، در [۲۱]، یک روش خوشه‌بندی را معرفی کرده‌اند که گروه ذرات را به زیرگروه‌های مختلفی افراز می‌کند و هر یک از آنها به‌جستجوی یک ناحیه‌ی محلی در فضای جستجو می‌پردازند.

نوروزی و همکاران با هدف حفظ تنوع جمعیت در فضای جستجو و همچنین انجام جستجوی محلی کارا، پیشنهاد ترکیب الگوریتم تفاضل تکاملی را با اتاماتای سلولی ارائه نمودند [۲۲]. ایده‌ی بکار گرفته شده توسط آنها، مشابه با بهینه‌سازی ازدحام ذرات سلولی [۱۸] می‌باشد.

در این مقاله یک الگوریتم ممتیکی مبتنی بر بهینه‌سازی ازدحام ذرات فازی، برای محیط‌های پویا ارائه شده است. در این مقاله ابتدا در بخش دوم در مورد ویژگی‌های محیط پویا بحث شده است است، مسئله قله‌های متحرک به عنوان تابع محکی برای شبیه‌سازی محیط‌های پویا در بخش دوم معرفی شده است. سپس در

بخش پنجم الگوریتم پیشنهادی مطرح شده است. نتایج آزمایشات در مقایسه با روش‌های معروف برای بهینه‌سازی محیط‌های پویا در بخش ششم ارائه شده است.

۲- محیط پویا[17]

هرگاه تغییری در هدف انجام بهینه‌یابی، نمونه مسئله یا بعضی تغییرات در محدودیت[18] یک مسئله بهینه‌یابی رخ دهد، ممکن است بهینه آن مسئله تغییر کند به عبارت دیگر محیط تغییر کرده است و بهینه‌سازی به صورت پویا می‌باشد [۲۳] که چنین شرایطی در خیلی از مسائل دنیای واقعی وجود دارد. در بهینه‌سازی محیط‌های پویا، هدف یک الگوریتم بهینه‌سازی تنها به یافتن پاسخ‌های مسئله محدود نمی‌شود، بلکه الگوریتم بایستی بتواند پاسخ‌های مسئله را دنبال کند. با توجه به آنکه پویایی در دنیای واقعی وجود دارد، در ادامه ویژگی‌های بهینه‌سازی در محیط‌های پویا مختصراً ارائه شده است.

۲-۱- معیارهای طبقه بندی

در این قسمت معیارهایی را که در طبقه بندی بهینه‌سازی محیط‌های پویا مطرح هستند، معرفی شده است.

۲-۱-۱- فرکانس تغییرات

موضوع فرکانس تغییرات زمانی مطرح می‌شود که با سوال‌های زیر برخورد می‌کنیم. محیط غالباً چگونه تغییر می‌کند؟ ابتدا از تغییرات نادر تا تغییرات مداوم؟ چه مدت زمان اختصاص می‌یابد تا یک الگوریتم به یک راه‌حل تطبیق شده برسد؟ از آنجا که مقایسه‌های زمانی، بسیار به جزئیات سخت‌افزاری و پیاده‌سازی بستگی دارد و معمولا تعداد ارزیابی‌ها در الگوریتم، فاکتور تغییر زمان است، میانگین تعداد ارزیابی‌ها بین تغییرات یک معیار مناسب خواهد بود زمان واقعی بین تغییرات محیط نیز معیار دیگری برای محاسبه زمانی نسبت به ارزیابی و برای مقایسه بین روش‌ها می‌باشد.

۲-۱-۲- شدت (سختی) تغییرات

سوال اصلی این است که سیستم با چه شدتی تغییر می‌کند؟ آیا تغییرات بسیار ناچیز است یا تغییرات به حدی زیاد است که یک وضعیت کاملا جدید رخ می‌دهد؟ این موضوع بوسیله فاصله ژنومی از بهینه قدیمی تا بهینه جدید مشخص می‌شود ولی ممکن است جنبه‌های دیگری (مثل اندازه فضای جستجو، احتمال رسیدن به بهینه جدید از بهینه قدیمی به کمک یک جهش) نیز وجود داشته باشد مثل زمانی که بهینه جدید ممکن است از بهینه قدیمی به وسیله یک تپه‌نوردی ساده پیدا شود.

۲-۱-۳- طول چرخه یا دقت چرخه

آیا بهینه به مکان‌های قبل باز خواهد گشت یا دست کم به آن‌ها نزدیک خواهد شد و اگر این طور است چگونه نزدیک می‌شود؟ در این‌جا میانگین تعداد وضعیت‌های محیطی که ممکن است میان دو وضعیت یکسان پشت سر هم رخ دهد را اندازه‌گیری می‌کنند اگر وضعیت جدید دقیقا یکسان نباشد فاصله راه‌حل جدید و قدیمی اهمیت پیدا می‌کند. طول و دقت چرخه معین می‌کند که راه‌حل قبلی یک استراتژی مورد استفاده است.

۲-۲- مسائل محک جهت ارزیابی

تا به حال، ده‌ها نوع متفاوت از مسائل بهینه‌سازی پویا برای ارزیابی الگوریتم‌های تکاملی مورد استفاده قرار گرفته‌اند. که شامل توابع ساده ریاضی تا انواع مسائل کاربردی مانند زمانبندی شبیه به واقعیت هستند. معروف‌ترین تابع محک برای بهینه‌سازی محیط‌های پویا، مسئله قله‌های متحرک [۵] است، که توسط برنک معرفی شد و در محافل علمی مورد پذیرش محققان مختلفی قرار گرفت. در این تابع محک، قله‌ها بطور پیوسته با زمان در حال تغییر هستند. در شکل ۱ نمونه‌ای از این تابع محک نمایش داده شده است. نکته مهم در این مسئله یافتن بهینه در این تغییرات است. تابع قله‌های متحرک شامل یک فضای چند بعدی، چندین قله با ارتفاع و پهنای متفاوت است، که ارتفاع، پهنا و موقعیت هر قله در طول زمان در حال تغییر است. تابع چند قله‌ای متحرک شامل m قله در n بعد و پارامترهای حقیقی است که هدف، یافتن بیشینه در هر زمان در میان m قله تا تغییر بعدی می‌باشد که به صورت رابطه (۱) تعریف می‌شود:

$$F(\vec{x}, t)=\mathrm{Max}(B(\vec{x}),\mathrm{Max}_{i=1..m}P(\vec{X},H_i(t),W_i(t),\vec{p}(t))) \qquad (۱)$$

در این تابع $B(\vec{x})$ یک شمای پایه ثابت در زمان و p تابع چند قله‌ای است که در هر Δe ارزیابی، H و W ارتفاع و پهنای قله‌ها بوسیله افزودن یک عدد تصادفی گوسین با میانگین صفر و واریانس σ و مکان هر قله بوسیله افزودن بردار v یا طول ثابت s (سختی) طبق رابطه (۲) تغییر می‌کند. در این تابع می‌توان پیچیدگی را با افزایش و کاهش Δe در زمان تغییر داد [۵].

$$\begin{cases} \sigma \in N(0,1) \\ hi(t) = hi(t-1) + height\_severity.\sigma \\ Wi(t) = Wi(t-1) + width\_severity.\sigma \\ \vec{p}i(t) = \vec{p}i(t-1) + \vec{v}i(t) \end{cases} \qquad (۲)$$

بردار v را می‌توان وابسته به تغییر قبلی آن ایجاد کرد که در این صورت تغییر موقعیت قله‌ها همسو با تغییرات قبل آن می‌شود و یا به صورت تصادفی آنرا ایجاد نمود که موجب می‌شود موقعیت قله‌ها به صورت تصادفی تغییر کند و هیچگونه وابستگی به تغییر قبلی نداشته باشد [۵].

۳- ازدحام ذرات

منبع الهام این الگوریتم، رفتار اجتماعی حیوانات، همانند حرکت دسته جمعی پرندگان و ماهی‌ها بوده است. هر عنصر جمعیت، یک ذره نامیده می‌شود. [۲۴] در واقع الگوریتم ازدحام ذرات از تعداد مشخصی از ذرات تشکیل می‌شود که به طور تصادفی، مقدار اولیه می‌گیرند. برای هر ذره دو مقدار وضعیت و سرعت، تعریف می‌شود که به ترتیب با یک بردار مکان و یک بردار سرعت، مدل می‌شوند. این ذرات، بصورت تکرار شونده‌ای در فضای n بعدی مسئله حرکت می‌کنند تا با محاسبهٔ مقدار بهینگی به عنوان یک ملاک سنجش، گزینه‌های ممکن جدید را جستجو کنند. بُعد فضای مسئله، برابر تعداد پارامترهای موجود در تابع مورد نظر برای بهینه‌سازی می‌باشد. یک حافظه به ذخیرهٔ بهترین موقعیت هر ذره در گذشته و یک حافظه به ذخیرهٔ بهترین موقعیت پیش آمده در میان همهٔ ذرات، اختصاص

می‌یابد. با تجربهٔ حاصل از این حافظه‌ها، ذرات تصمیم می‌گیرند که در نوبت بعدی، چگونه حرکت کنند.

در هر بار تکرار، همهٔ ذرات در فضای n بعدی مسئله حرکت می‌کنند تا در نهایت نقطهٔ بهینهٔ عام، پیدا شود. حال ذرات، سرعت‌هایشان و موقعیت‌شان را برحسب بهترین جواب‌های مطلق و محلی به‌روز می‌کنند. معادله ۳ نشان دهنده تغییر وضعیت و به‌روز رسانی سرعت ذره می‌باشد [۲۵].

۴- ممتیک

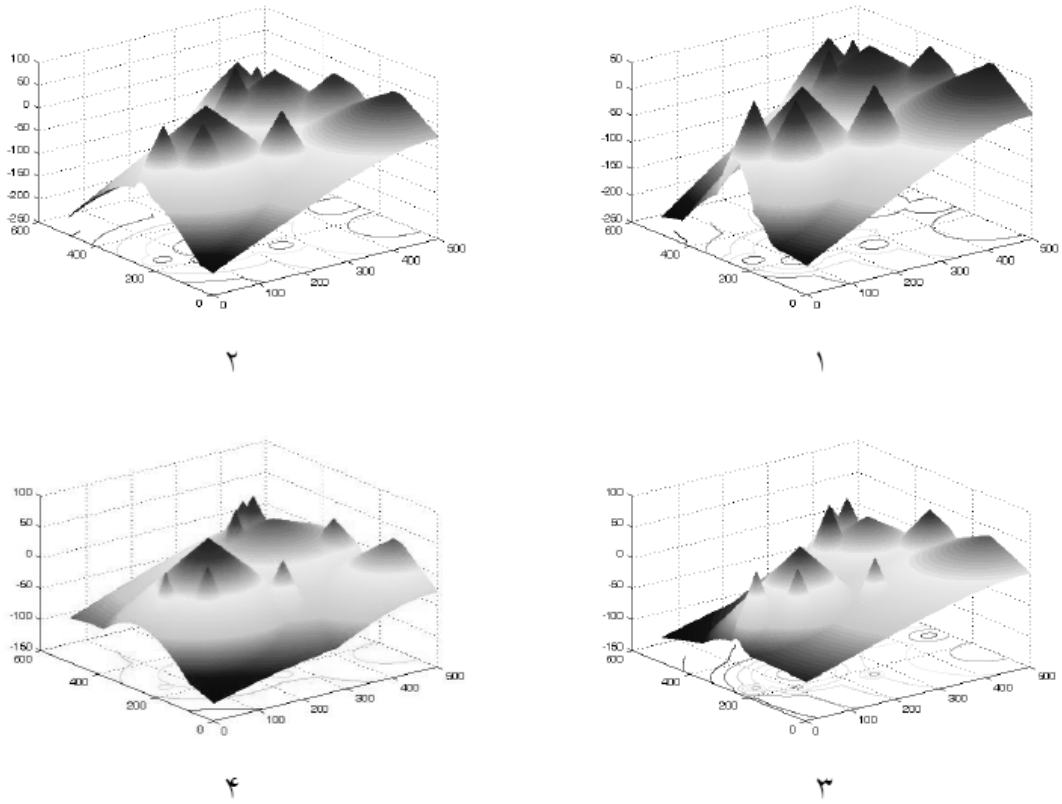
الگوریتم‌های ژنتیکی همانند سایر الگوریتم‌های مکاشفه‌ای در گام‌های نخست اجرای الگوریتم ناحیه‌هایی از فضای حالت مسئله که بهینه‌های سراسری و محلی در آن واقع شده‌اند را به خوبی شناسایی می‌کنند اما در ادامه مسیرشان به سمت بهینه سراسری بسیار کند عمل می‌نمایند [۱].

عیب دیگری که این الگوریتم‌ها با آن مواجه می‌باشند عدم پایداری این الگوریتم‌ها است. به این معنی که کیفیت پاسخ‌هایی که از اجراهای مختلف الگوریتم به دست می‌آید ممکن است تفاوت‌های بسیاری داشته و حتی غیرقابل اعتماد باشند. از بین روش‌هایی که برای برطرف ساختن مشکلات الگوریتم‌های مکاشفه‌ای ارائه شده است، استراتژی ترکیبی جایگاه ویژه‌ای دارند [۱] [۳].

استراتژی پیوندی از به کارگیری روش‌های مختلف در فرایند حل مسئله حاصل می‌گردد. الگوریتم‌های ممتیک از مشهورترین اعضاء این خانواده به شمار می‌آیند که از ترکیب الگوریتم‌ها با یک جستجوی محلی مکاشفه‌ای مانند روش تپه نوردی حاصل می‌گردد [۳] [۴]. در این الگوریتم‌ها یک عملگر جستجوی محلی پس از عملگرهای ژنتیکی شایستگی را بهبود می‌بخشد (تقلید) و پس از آن اعمال ارزیابی و جایگزینی جمعیت صورت می‌گیرد. اینکه برای پیاده‌سازی یک الگوریتم ممتیک برای جستجوی محلی از کدام روش استفاده شود، نتایج اجرای بسیار متفاوتی خواهد داشت و وابسته به نوع مسئله است، اما در بسیاری از مسائل به خصوص مسائلی با محیط‌های گسسته نیازمند تعریف یک روش جستجوی محلی جدید است.

انواع مختلفی برای الگوریتم ممتیک معرفی شده است. اخیراً نیز تحقیقات جدید حول الگوریتم ممتیکی ارائه شده است. الگوریتم‌های ممتیکی در طی اجرا، به علت بهبود و تصحیح محلی موثری که بر روی ژن‌ها انجام می‌دهند، یک ظرفیت جستجو با قدم کوچک[۱۹] قوی ارائه می‌کنند، اما ممکن است ظرفیت جستجو با قدم بزرگ[۲۰] را از دست بدهند مانند همگرا شدن جمعیت به یک بهینه که باید در محیط‌های پویا اجتناب شود. بنابراین، این می‌تواند یک ایده پژوهشی، جالب برای امتحان کارایی الگوریتم‌های ممتیکی که با روش‌های تنوعی مناسب، بهبود یافته‌اند در محیط‌های پویا باشد.

قبلاً گزارش شده بود که عملگرهای مِم چندگانه در چارچوب یک الگوریتم ممتیکی می‌تواند بکار گرفته شود. این به آن خاطر است که هر مِم یک جستجوی اساسی انجام می‌دهد، که یک روش را برای برخی کلاس از مسائل کارآمدتر می‌کند و برای برخی دیگر این گونه نیست. از این‌رو جستجوی محلی (مِم) وابسته به مسئله است. بنابراین این‌که چگونه عملگرها مِم بهینه را یافت و از روش‌های جستجوی محلی نامناسب جلوگیری نمود، یکی از موضوعات بسیار مهم شده است. به منظور بررسی این مسئله، بسیاری از محققان از روش‌های جستجوی چندگانه در الگوریتم‌های ممتیکی استفاده می‌کنند. در مقایسه با روش‌های قدیمی الگوریتم‌های ممتیکی که از یک مِم تنها در سراسر اجرا استفاده می‌کردند، الگوریتم‌های ممتیکی با چند مِم می‌تواند کارآیی بهتری را بدست آورد.



شکل ۱- تغییرات محیط قله‌ها در مسئله قله‌های متحرک

ایده کلیدی استفاده از چند عملگر جستجوی محلی در الگوریتم‌های ممتیکی، بالا بردن مشارکت و رقابت مم‌های متفاوت است، در حالی‌که قادر باشند با یکدیگر کار کنند تا به هدف بهینه اشتراکی دست یابند. برخی از محققان پیشنهاد نموده‌اند که عملگرهای مِم باید همزمان روی آن ژن‌هایی که برای بهبود محلی انتخاب شده‌اند، اجرا شوند و یک مکانیزم یادگیری دقیق باید شانس انتخاب آنها را در مرحله بعد تطبیق دهد. هر چند نری[۲۱] و دیگران، یک الگوریتم ممتیکی با مِم چندگانه با یک روش غیر رقابتی پیشنهاد کردند که روش‌های جستجوی محلی متفاوت در طی دوره‌های تکاملی جمعیت‌های متفاوت می‌تواند فعال گردد [۲].

در ادامه مفهوم‌های ممتیک و تفاوت‌های آن با دیگر الگوریتم‌ها و کاربردش در محیط‌های پویا بحث شده است.

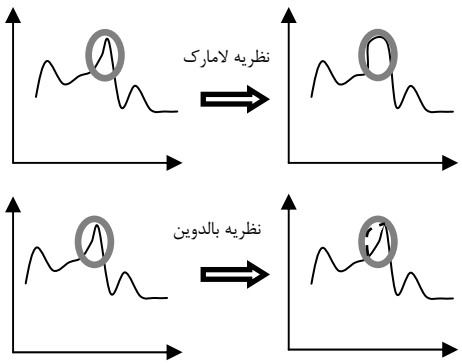
۴-۱- الگوریتم‌های ممتیک مبتنی بر نظریه لامارک

برخی از متخصصان معتقدند آنچه که یک موجود در طول زندگی از محیط و جامعه خود می‌آموزد در ساختار ژنی آن موجود نیز کد می‌شود. این اعتقاد مبنای رویکرد لامارک است. در این رویکرد پس از آن که در همسایگی یک عضو جمعیت جستجوی محلی صورت گرفت، بهترین همسایه جایگزین آن عضو می‌شود، به این ترتیب بهبود در شایستگی اعضاء جمعیت هم توسط عملگرهای ژنتیک و هم توسط عملگر ممتیک (جایگزینی بهترین همسایه) صورت می‌گیرد [۲۶].

۴-۲- الگوریتم‌های ممتیک مبتنی بر نظریه بالدوین

بعضی از دیگر صاحب‌نظران معتقدند دلایل کافی برای این مدعا که آموخته‌های موجود در ژن او کد می‌شود وجود ندارد و آموخته‌های موجود تنها در طول حیات خود او در اختیار هستند و فقط توسط آموزش (تقلید) به موجود دیگر انتقال می‌یابند، بنابراین در رویکرد بالدوین پس از آن که در همسایگی یک عضو جمعیت جستجوی محلی صورت گرفت، شایستگی بهترین همسایه یک عضو جمعیت را جایگزین شایستگی آن عضو می‌نمایند.

در واقع در این روش یک عضو جمعیت در تجربه همسایگانش سهیم می‌شود به همین علت به آن اثر بالدوین نیز می‌گویند [۳]. در برخی از روش‌ها از هر دو رویکرد به صورت همزمان نیز بهره برده شده است که به آنها متا لامارکی گفته می‌شود [۲۶].



شکل۲- شمای یاز نظریه‌هایلامارک و بالدوین در فضای دو بعدی

در شکل ۲ قسمت بالا و نظریه لامارک جمعیت و ارزش آنها جایگزین شده‌اند ولی در نظریه بالدوین فقط ارزش جمعیت همسایه برابر بهینه محلی می‌باشد.

۵- الگوریتم پیشنهادی

در این مقاله با ترکیب الگوریتم ازدحام ذرات[۲۲] و تکنیک جستجوی محلی به صورت یک الگوریتم ممتیکی، راه حلی برای محیط‌های پویا ارائه شده است. برای افزایش کارایی اکتشاف قله‌های جدید برای فضاهای جستجوی جدید از طرح مهاجرت ذرات تصادفی خودسازمانده بهره‌مند شده است. مطالعات تجربی و پیاده‌سازی و مقایسه بر روی قله‌های متحرک نشان می‌دهد که الگورتیم ممتیک بر پایه ازدحام ذرات در محیط‌های پویا بسیار قوی عمل کرده است. ازدحام ذرات دارای دو پارامتر محلی و سراسری می‌باشد. در این‌جا از پارامتر محلی به همراه جستجوی محلی استفاده شده است. با توجه به شبه کد ازدحام ذرات مبتنی بر ممتیک، چند مفهوم ضروری در ادامه توضیح داده شده است.

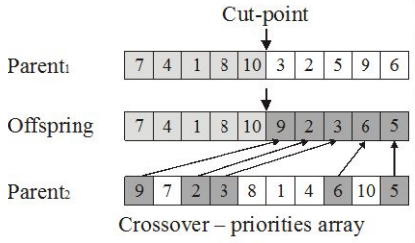
شبه کد زیر نمای کلی از الگوریتم ممتیک می‌باشد، همانطور که می‌بینید شباهت زیادی به الگوریتم ژنتیک داشته ولی بهمراه یک جستجوی محلی می‌باشد که در شبه کد شکل ۳ و ۶، این جستجوی محلی آمده است.

```
1.  Function Mempso
2.  Begin
3.      Initialize population
4.      Evaluation population
5.      While population size
6.        Begin
7.          New pop=Local search (using PSO)
8.          Replace new pop
9.          Crossover
10.         Mutation
11.         Selection
12.       End
13.     Population = New pop
14.     Best fitness=max (Evaluation population)
15. End
```

شکل ۳- شبه کد رویکرد کلی برای الگوریتم پیشنهادی

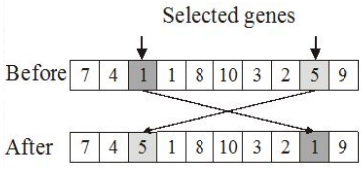
در سطر ۳ الگوریتم ابتدا بایستی جمعیت اولیه مقداردهی شوند، که معمولاً بصورت تصادفی یا بوسیله یک تابع توزیع گوسی در فضای مساله مقداردهی می‌شوند و در خط ۴ ارزش این جمعیت محاسبه می‌شود. سپس در سطرهای ۷ و ۸ پس از بهبود جمعیت توسط الگوریتم ازدحام ذرات، جستجو بین همسایه‌ها انجام شده و جایگزینی انجام می‌شود.

عملگر تبادل، مهمترین عملگر تولید نسل در الگوریتم ژنتیک است. این عملگر به نحوی بخشی از یک کروموزوم را به بخشی از یک کروموزوم دیگر الحاق می‌کند که کروموزوم جدید که فرزند نامیده می‌شود، با والدین خود هم‌اندازه باشد. برای انتخاب تعداد و محل نقاط انجام تبادل روی والدین، روش‌های متفاوتی مانند تبادل تک‌نقطه، دونقطه و تبادل یکنواخت وجود دارد که هر یک می‌تواند کاربرد خود را داشته باشد. در الگوریتم پیشنهادی مانند شکل ۴ از عملگر تبادل تک نقطه استفاده شده است اما آنچه بیش از نوع عملگر تبادل اهمیت دارد، انتخاب والدین برای انجام تبادل است. بر اساس بررسی، بهترین تبادل زمانی رخ می‌دهد که دو والد انتخاب شده برای تبادل، از نظر مقادیر ژن‌های خود، حداکثر اختلاف را با یکدیگر داشته باشند. ابتدا کروموزومی که دارای حداقل اختلاف با دیگر کروموزوم ها باشد به عنوان والد اول انتخاب می‌شود. سپس از بین سایر کروموزوم‌ها آن که دارای حداکثر میزان اختلاف با والد انتخابی اول باشد بعنوان والد دوم انتخاب شده و تبادل روی آن‌ها انجام می‌شود.



شکل ۴- عملگر تبادل (تقاطع)

جهش نقش ثانویه بسیار مهمی در عملکرد الگوریتم ژنتیک دارد. در سیستم های ژنتیک هوشمند عملکرد جهش از به دام افتادن در کمینه محلی جلوگیری می‌کند. جهش یک گردش تصادفی در فضای جستجو است. عملگر جهش را می توان به شیوه‌های متفاوت برروی کروموزوم‌ها اجرا نمود اما آن‌چه در الگوریتم پیشنهادی استفاده شده است یک عملگر جهش با انتخاب تصادفی دو نقطه از یک کروموزوم است. شکل ۵ نمایش یک عملگر جهش می‌باشد.



شکل ۵- عملگر جهش

حال با متحول شدن جواب‌ها، برای انتخاب بهترین‌ها از انتخاب مسابقه‌ای استفاده می‌کنیم.

```
1.  Local Search(population)
2.      While (size of population)
3.          Begin
4.          Best=Select some Best of population
5.          Generated neighbor from best
6.          Compare each individual with them neighbors
7.          If fitness (neighbor (i))>fitness(best(i))
8.                  Lamarkian{(Replace best by neighbor
                    and fitness)or Baldvinian method
                    {(Replace fitness of best neighbor)}
9.          End if
10.         Insert best in new population
11.         Return new population
12. End
```

شکل ۶- شبه کد جستجوی محلی ممتیک

در شبه کد ۶ همسایگی‌ها توسط یکی از دو نظریه مورد بررسی قرار می‌گیرند که چگونگی آن در بخش بعدی توضیح داده شده است.

۵-۱- ممتیک- ازدحام ذرات فازی

انواع مختلفی از الگوریتم‌های تکاملی و هوش جمعی وجود دارند که برای جستجو و بهینه‌یابی استفاده می‌شوند، که در اینجا از الگوریتم ازدحام ذرات فازی به عنوان جستجوگر بین همسایه‌های یک مم استفاده شده است.

هر عنصر جمعیت، یک ذره نامیده می‌شود است. در واقع الگوریتم ازدحام ذرات از تعداد مشخصی از ذرات تشکیل می‌شود که به طور تصادفی، مقدار اولیه می‌گیرند. برای هر ذره دو مقدار وضعیت و سرعت، تعریف می‌شود، همهٔ ذرات در فضای n بعدی مسئله (همسایگی) حرکت می‌کنند تا در نهایت نقطهٔ بهینهٔ عام، پیدا شود. حال ذرات، سرعت‌هایشان و موقعیت‌شان را بر حسب بهترین جواب‌های مطلق و محلی به‌روز می‌کنند [۲۴]. در واقع به سمت بهترین همسایه پیش می‌روند.

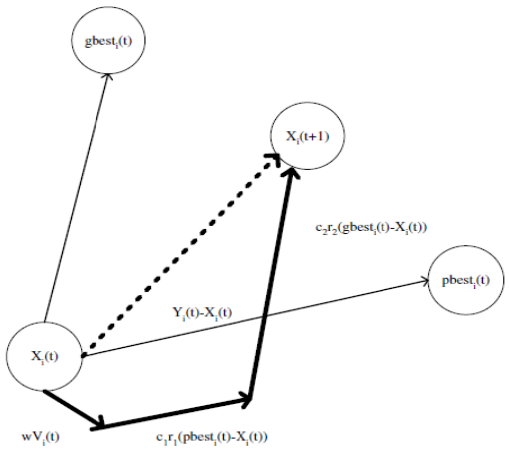
الگوریتم ازدحام ذرات استاندارد از سه مرحله‌ی زیر تشکیل شده است:

- مقدار دهی اولیه موقعیت و سرعت ذرات
- به روز رسانی بردار سرعت
- به روز رسانی موقعیت

$$v_{id}(t+1) = \omega v_{id}(t) + C_1 r_1 (p_{ld} - x_{id}(t)) + C_2 r_2 (p_{gd} - x_{id}(t))$$
$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \qquad (۳)$$

که در آن ω پارامتر اینرسی، v_{id} سرعت i امین ذره، x_{id} موقعیت i امین ذره، r_1, r_2 اعداد تصادفی مستقل با توزیع یکنواخت، C_1 , C_2 فاکتورهای یادگیری (ثابت‌های C_1 , C_2 به ترتیب، پارامتر ادراکی و پارامتر اجتماعی نامیده می‌شوند)، p_{ld} بهترین جواب محلی و p_{gd} بهترین جواب مطلق می‌باشند. الگوریتم ازدحام ذرات، بردار سرعت هر ذره را به‌روز کرده و سپس مقدار سرعت جدید را به موقعیت و یا مقدار سرعت ذره می‌افزاید که شکل ۷ نشان دهنده این گفته است [۲۴].

به‌روز کردن‌های سرعت، تحت تأثیر هر دو مقدار بهترین جواب محلی و بهترین جواب مطلق قرار می‌گیرند. بهترین جواب محلی و بهترین جواب مطلق، بهترین جوابهایی هستند که تا لحظه‌ی جاری اجرای الگوریتم، به ترتیب توسط یک ذره و در کل جمعیت به دست آمده‌اند. مزیت اصلی ازدحام ذرات این است که پیاده‌سازی این الگوریتم ساده بوده و نیاز به تعیین پارامترهای کمی دارد.



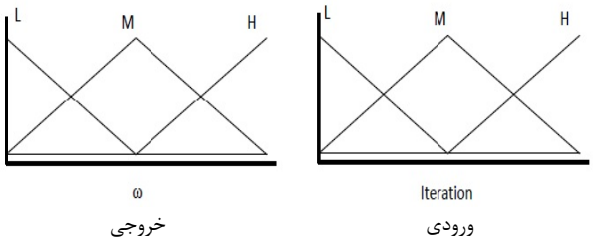
شکل ۷- چگونگی حرکت ذره و بروز رسانی موقعیتش

تعیین پارامتر الگوریتم ازدحام ذرات، با سیستم فازی، سبب بهبود عملکرد الگوریتم ازدحام ذرات گشته است.در این مقاله پارامتر اینرسی، در طی دوره‌ی اجرای الگوریتم، به واسطه‌ی کنترل کننده‌ی فازی، تعیین و تطبیق داده می‌شود. کنترل کننده‌ی فازی دارای یک ورودی و یک خروجی می‌باشد که تعیین پارامتر وزن اینرسی به عنوان خروجی سیستم فازی در نظر گرفته شده است. پیاده‌سازی انجام شده بهبود سیستم ارائه شده، نسبت به الگوریتم استاندارد ازدحام ذرات را نشان می‌دهد. این الگوریتم از طریق تغییر یکی از سه پارامتر زیر قابل بهبود است:

- ضریب وزن اینرسی
- ضریب پارامتر اجتماعی
- ضریب پارامتر ادراکی

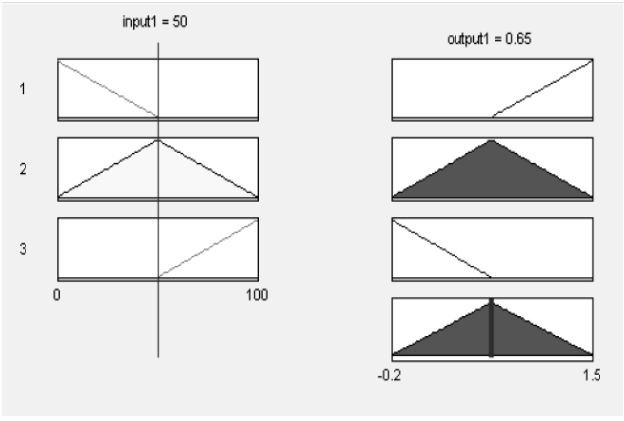
در این مقاله سیستم فازی به عنوان یک سیستم مجزا به الگوریتم اضافه شده است. پارامتر ورودی این سیستم فازی عدد تکرار[۲۳] الگوریتم و خروجی سیستم فازی پارامتر ضریب وزنی اینرسی می باشد. ایده‌ی اصلی روش پیشنهادی مبتنی بر بهبود حرکات ذرات در فضای مسئله با تطبیق آن با دفعات تکرار می‌باشد. تعیین هدفمند پارامترهای الگوریتم ازدحام ذرات سبب کارایی بهتر الگوریتم شده است. در سال ۲۰۰۱، ابرهات سیستمی مبتنی بر فازی ارائه داد [۲۷] اما در این مقاله سعی شده تا با استفاده از سیستم فازی با توجه به تعداد دفعات تکرار مقدار سیستم فازی تعیین گردد [۲۴].

ورودی سیستم فازی شماره تکرار در الگوریتم ازدحام ذرات بکار رفته می‌باشد و با تکرار کل الگوریتم رابطه‌ای ندارد. مقدار خروجی سیستم فازی برای الگوریتم ازدحام ذرات ضریب پارامتر اینرسی می‌باشد که در شکل ۸ آمده است. توابع سیستم فازی برای الگوریتم ازدحام ذرات در شکل ۹ و۱۰ آمده است، تعریف می‌شود.



شکل ۸- نمونه ورودی و خروجی سیستم فازی

در این سیستم از سیستم فازی ممدانی و نیز از غیر فازی‌ساز میانگین مراکز استفاده شده است.

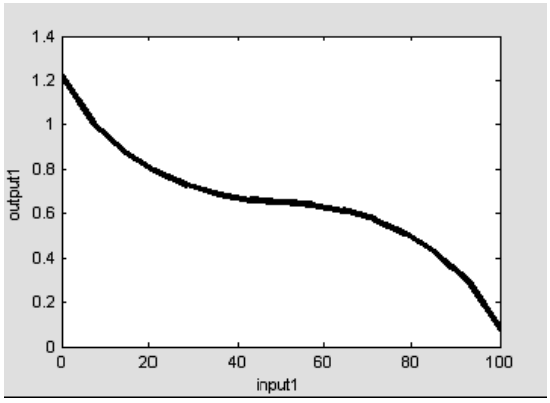


شکل ۹- پنجره نمایش قوانین

که در آن متغیر ورودی توسط چهارتایی
<Iteration, {Low, Middle, High}, [0 100], R> مشخص می‌گردد:
نام متغیر زبانی Iteration
مجموعه مقادیر زبانی {Low, Middle, High}
دامنه فیزیکی واقعی متغیر [0 100]
تابع نگاشت L= [0 50], M= [0 100], H =[50 100]
و نیز متغیر خروجی توسط چهارتایی
<Inertia, {Low, Middle, High}, [-.2 1.5], R> مشخص می‌گردد:

- نام متغیر زبانی Inertia
- مجموعه مقادیر زبانی {Low, Middle, High}
- دامنه فیزیکی واقعی متغیر [-.2 1.5]
- تابع نگاشت L = [-.2 .65], M = [-.2 1.5], H =[.65 1.5]

قوانین مورد استفاده نیز به شرح زیر است.



Rules
If itr = L then ω = H
If itr = M then ω = M
If itr = H then ω = L

شکل ۱۰- قوانین فازی

به همین صورت برای الگوریتم ممتیک، سیستم فازی تعریف می‌شود ولی با این فرق که خروجی ضریب پارامترهای جهش و تقاطع است. فرض این مقاله این است که ابتدای الگوریتم بایستی با تنوع بالا شروع به کار کند، به همین منظور نرخ تقاطع در ابتدا زیاد و نرخ جهش کم در نظر گرفته می‌شود و برای افزایش کارایی از فازی‌سازی استفاده می‌شود.

۵-۲- مواجهه با تغییرات

بطور خلاصه اساس کار الگوریتم ما این است که، جمعیت اولیه پس از تولید توسط الگوریتم ازدحام ذرات فازی بهبود داده می‌شوند، سپس جستجوی محلی بین همسایگی‌ها آغاز شده، این همسایگی‌ها می‌توانند در هر بعدی از مساله باشد و در صورت بهتربودن همسایگی‌ها جایگزین جمعیت می‌شوند (استفاده از یکی از دو رویکرد ممتیکی). حال جواب‌ها توسط دو عملگر جهش و تقاطع برای نسل بعدی آماده می‌شوند.

همانطور که گفته شده این الگوریتم بر اساس جستجوی محلی شکل گرفته شده، و مقدار خروجی بر اساس الگوریتم‌های ازدحام ذرات و ممتیک و مجموعه جوابی که این دو الگوریتم با همدر تعامل هستند، می‌باشد. پس می‌توان گفت پس از تغییر در مکان قله‌ها، بوسیله جستجو محلی تغییرات شناسایی شده یا به عبارتی تعقیب قله‌ها را انجام می‌دهد. حال با احتمال آنکه قله‌های دیگری در حال پیداش و یا افزایش ارتفاع هستند، همچنان اپراتورهای جهش و تقاطع و همچنین درصدی تولید جمعیت تصادفی برای پیدا کردن آنها کمک می‌کنند.

۶- آزمایشات و نتایج

در ادامه تنظیمات مربوط به پارامترهای مولد تابع محک قله‌های متحرک و معیار ارزیابی ذکر شده است. الگوریتم پیشنهادی در آزمایشات مختلف با الگوریتم‌های

[۲۸]mPSO ،[۲۰]FMSO ،[۱۷]mQSO(5,5q) ،[۱۸]Cellular PSO ،[۳۳]FTMPSO ،[۳۲]DynDE ،[۳۱]HmSO ،[۳۰]PSO-AQ ،[۲۹]APSO و CPSOL [۳۴] مورد مقایسه قرار گرفته است.

۶-۱- مولد تابع محک قله‌های متحرک

کارایی الگوریتم پیشنهادی توسط مولد تابع محک قله‌های متحرک مورد ارزیابی قرار گرفته است. در مقالات فراوانی از این تابع محک برای شبیه‌سازی و بررسی کارایی الگوریتم‌ها بهره برده شده است. مقادیر پارامترهای مولد تابع محک قله‌های متحرک مطابق سناریوی II به صورت جدول ۱ در نظر گرفته شده است. قابل توجه است که برای مولد تابع محک قله‌های متحرک ۳ سناریوی مختلف (I, II, III) تعریف شده است که در اکثر مقالات [۲۸-۳۴] از سناریوی II استفاده شده و جنبه‌های پویایی محیط مورد بررسی قرار می‌گیرد.

در جدول ۱، s بیانگر حداکثر 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 مشخص‌کننده‌ی دامنه ارتفاع و پهنای قله‌هاست که با شدت hs و ws تغییر می‌کنند. I مشخص کننده ارتفاع اولیه قله‌هاست و A دامنه فضای جستجو در تمامی ابعاد را مشخص می‌کند.

جدول ۱- تنظیم مقادیر پارامترهای مولد قله‌های متحرک

پارامترها	مقادیر
mتعداد قله‌ها	10
fفرکانس تغییر	5000
h_s سختی ارتفاع	0.7
w_sسختی پهنا	0.1
شکل قله	مخروطی
S طول تغییر	1.0
d ابعاد مسئله	5
H دامنه ارتفاع	[30 70]
W دامنه پهنا	[1 12]
I مقدار اولیه ارتفاع	50.0
A محدوده فضای جستجو	[0 100]

۶-۲- معیار ارزیابی

با توجه به آنکه در مسائل بهینه‌سازی پویا یک پاسخ بهینه ثابت وجود ندارد، هدف یافتن اکسترمم‌های مسئله بصورت دقیق نیست بلکه ردیابی مسیر پیشروی نقاط بهینه در فضای مسئله مطلوب است. برای اثبات برتری یک روش نسبت به روش‌های دیگر، در مقالات یک روش دیگر عمومی نمایش نمودارهای همگرایی مربوط به بهترین یا میانگین شایستگی مربوط به روش‌های مختلف و مقایسه بصری آنهاست. برای یک مقایسه دقیق، محاسبه عددی معمولاً ترجیح داده می‌شود.

از آنجا که در محیط‌های پویا بهترین راه حل بدست آمده معیار مفیدی برای ارزیابی و گزارش نمی‌باشد، از معیارهای خطای جاری[۲۴] و خطای آفلاین[۲۵] جهت مقایسه و نتیجه‌گیری استفاده شده است. خطای آفلاین به عنوان معروف‌ترین معیار ارزیابی کارایی کمی برای بهینه‌سازی محیط‌های پویا به صورت معادله ۴ تعریف می‌شود.

$$Offline - Error = \frac{1}{T}\sum_{t=1}^{T}(\text{Minimum Error After last change}) \quad (۴)$$

۶-۳- نتایج آزمایشات

مقادیر پارامترهای الگوریتم پیشنهادی برای انجام آزمایش‌ها طبق جدول ۲ در نظر گرفته شده است.

جدول ۲- تعیین پارامترهای قسمت ازدحام ذرات الگوریتم

پارامتر ها	مقادیر
C_1	۲
C_2	۲
اندازه جمعیت	۸۰
تکراردر جستجوی محلی	۱۰
Wاینرسی	[0…1] بصورت فازی

در آزمایشات انجام شده اندازه جمعیت بصورت ثابت برابر ۱۰۰ در فرکانس‌های مختلف جواب‌های جداول ۴ و ۵ را تولید کرده و مقایسه بر اساس میانگین‌گیری از ۲۰ بار اجرای مکرر انجام شده است. به عنوان نمونه در فرکانس ۵۰۰، ۱ قله و۲۰ بار اجرا میانگین برابر ۱.۴۸ بدست آمده و انحراف معیار برابر ۰.۳ می‌باشد و همین فرکانس، ۱۰ قله و۲۰ بار اجرا میانگین برابر ۳.۰۱ بدست آمده و انحراف معیار برابر ۰.۶۸ می‌باشد. که این انحراف معیار نشان‌دهنده کارایی الگوریتم‌ها و اثبات نتیجه آنها می‌باشد، هر چه میزان این شاخص پایین بوده به منزله بهتر بودن الگوریتم از لحاظ سازگاری با محیط است.

جدول ۳- میانگین و انحراف معیار در ۲۰ بار اجرا

قله‌ها / فرکانس	۱	۱۰	۲۰	۳۰
۵۰۰	۱.۴±۰.۳۰	۳.۰۱±۰.۶۸	۲.۹۷±۰.۶۱	۳.۱۲±۰.۷۰
۱۰۰۰	۱.۳±۰.۵۰	۲.۷±۰.۴۲	۲.۶۸±۰.۲۱	۳.۲۲±۰.۴۰

شکل ۱۱ نشان دهنده تعداد تغییرات برای ۱۰۰۰ تولید نسل با فرکانس تغییر ۵۰ است، به ازای هر ۵۰ نسل تولید شده، یک تغییر دیده می‌شود که در کل این اجرا تغییر در قله ها ۲۰ بار می‌باشد.

شکل ۱۲ نشان‌دهنده تعداد تغییرات برای ۱۰۰۰ تولید نسل با فرکانس تغییر ۱۰۰ است، به ازای هر ۱۰۰ نسل تولید شده، یک تغییر دیده می‌شود که در کل این اجرا تغییر در قله‌ها ۱۰ بار می‌باشد.

شکل ۱۳ نشان‌دهنده تعداد تغییرات برای ۵۰۰۰۰ تولید نسل و ۱۰ قله و فرکانس تغییر ۵۰۰ است، به ازای هر ۵۰۰ نسل تولید شده، یک تغییر دیده می‌شود که در کل این اجرا تغییر در قله‌ها ۱۰۰ بار می‌باشد.

همچنین مقایسه‌ای هم بین الگوریتم پیشنهادی و الگوریتم‌های دیگر بر اساس پارامترهای استاندارد تعیین شده در جدول ۱ و ۲، نیز در جداول ۴ و ۵ با فرکانس‌های مختلف آمده است.

در ضمن برای مقایسات کارایی الگوریتم براساس نظریه لامارک و بالدوین بر روی تابع قله‌های پویا، نتایج حاکی از آن است که برای نظریه لامارک تعقیب قله‌ها فشرده‌تر بوده ولی در کل براساس میانگین‌گیری از ۲۰ بار اجرا از دو رویکرد کارایی دو رویکرد تقریباً یکسان می‌باشد. اگر بخواهیم در نتایج دقت کنیم، چون

جواب‌ها یکسان هستند، از انحراف معیار برای مقایسه کمک می‌گیریم، که برای رویکرد لامارک این مقدار کمتر بوده است. این نتایج در جدول ۶ آمده‌اند.

جدول ۴- مقایسه خطای آفلایندر فرکانس ۵۰۰

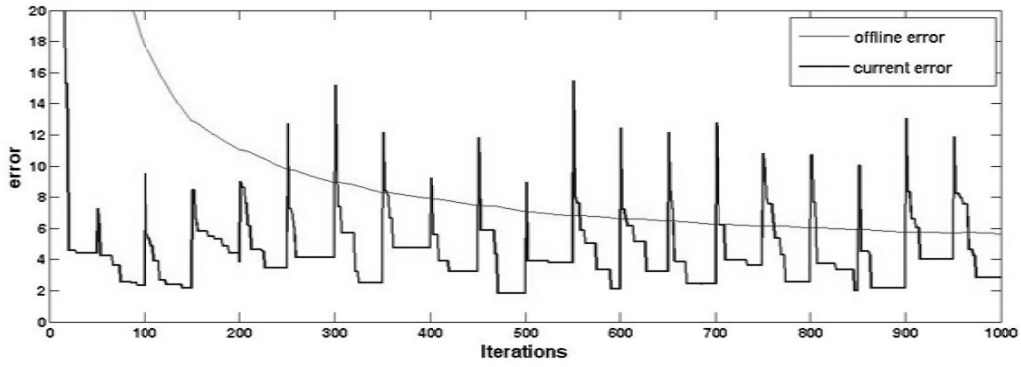
الگوریتم / قله‌ها	MQSO	Cellular PSO	FMSO	CPSOL	Memetic+
۱	۲۸.۳۸±۱.۸۹	۱۱.۴۱±۰.۶۸	۳.۴۴±۰.۱۱	۸.۲۹±۰.۵۵	**۱.۴۰±۰.۳۰**
۱۰	۵.۷۴±۰.۸	۶.۸۱±۰.۱۷	۳.۱۱±۰.۰۶	۵.۴۵±۰.۱۷	**۳.۰۱±۰.۶۸**
۲۰	۶.۸۰±۰.۰۴	۷.۷۳±۰.۰۷	۳.۳۶±۰.۰۶	۵.۴۷±۰.۱۹	**۲.۹۷±۰.۶۱**
۳۰	۷.۲۸±۰.۰۴	۸.۳۹±۰.۱۱	۳.۲۸±۰.۰۵	۵.۵۹±۰.۱۲	**۳.۱۲±۰.۷**

جدول ۵- مقایسه خطای آفلاین در فرکانس ۱۰۰۰

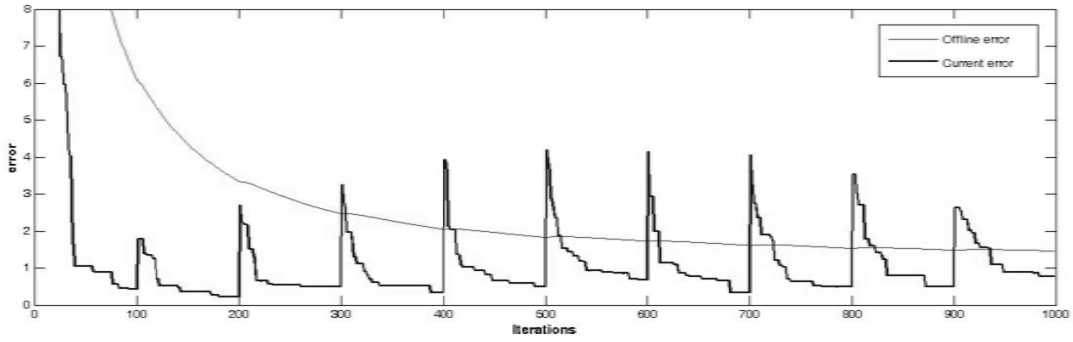
الگوریتم / قله‌ها	MQSO	Cellular PSO	FMSO	CPSOL	Memetic+
۱	۱۵.۰۲±۰.۹۹	۸.۷۴±۰.۴۰	۱۴.۴۲±۰.۴	۴.۷۴±۰.۳۲	**۱.۳۰±۰.۵۰**
۱۰	۴.۰۰±۰.۰۶	۴.۷۸±۰.۱۱	۱۰.۴۰±۰.۱	۳.۲۰±۰.۲۰	**۲.۷۰±۰.۴۲**
۲۰	۵.۱۱±۰.۰۵	۵.۹۵±۰.۱۱	۱۰.۳۳±۰.۱	۳.۵۲±۰.۱۷	**۲.۶۸±۰.۲۱**
۳۰	۵.۷۱±۰.۰۵	۶.۶۵±۰.۱۲	۱۰.۰۶±۰.۱	۳.۹۶±۰.۱۲	**۳.۲۲±۰.۴۰**

جدول ۶- مقایسه دو رویکرد

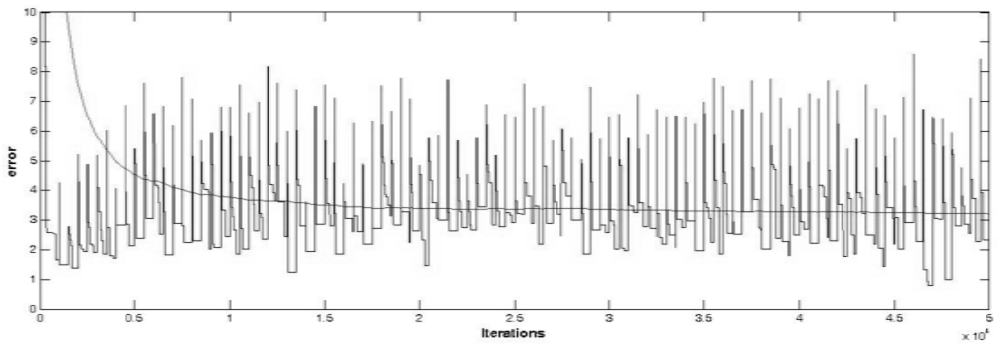
رویکرد	فرکانس	قله	تعداد اجرا	میانگین	انحراف معیار
بالدوین	۱۰۰	۱	۲۰	۲.۰۹	۰.۳۱
لامارک	۱۰۰	۱	۲۰	۲.۰۸	۰.۲۸



شکل ۱۱- نمودار خطای آفلاین برای فرکانس ۵۰ و تک قله



شکل ۱۲- نمودار خطای آفلاین برای فرکانس ۱۰۰ و تک قله



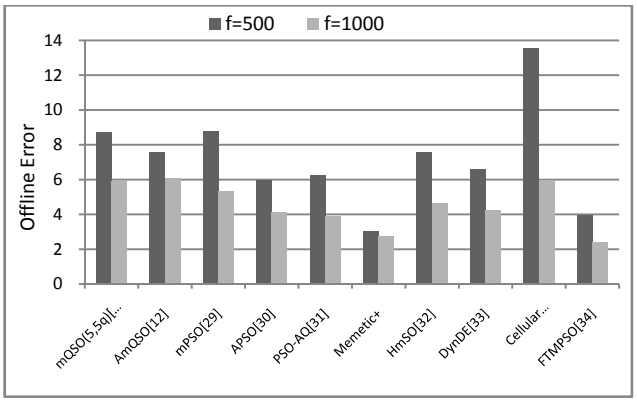
شکل ۱۳- نمودار خطای آفلاین برای فرکانس ۵۰۰ و ۱۰ قله و ۵۰۰۰۰ نسل

جدول۷ - مقایسات با الگوریتم‌های دیگر

Algorithms	***f***	***1 peak***	***10peak***	***20peak***	***30peak***
mQSO(5,5q)[17]	500	8.19(0.17)	8.72(0.20)	8.80(0.21)	9.07(0.25)
AmQSO[15]		7.48(0.19)	7.57(0.32)	7.10(0.39)	6.82(0.34)
mPSO[28]		8.88(0.14)	8.76(0.18)	8.43(0.17)	8.01 (0.19)
APSO[29]		6.20(0.04)	5.95(0.06)	6.03(0.07)	5.81 (0.08)
PSO-AQ[30]		6.13(0.17)	6.21(0.16)	5.43(0.18)	5.71(0.18)
Memetic+		**1.4(0.3)**	**3.01(0.68)**	**2.97(0.61)**	**3.12(0.7)**
HmSO[31]		8.53(0.49)	7.56(0.27)	7.81(0.20)	8.33(0.18)
DynDE[32]		27.68(35.57)	6.59(0.12)	7.45(0.03)	7.97(0.03)
Cellular Pso[18]		22.37(3.8)	13.55(0.5)	12.77(0.3)	12.55(0.4)
FTMPSO[33]		1.76(0.09)	3.91(0.19)	4.83(0.19)	5.05(0.21)
mQSO(5,5q)[17]	1000	5.54(0.11)	5.87(0.13)	5.81(0.15)	5.85(0.15)
AmQSO[15]		5.75(0.26)	6.06(0.14)	5.20(0.38)	5.36(0.47)
mPSO[28]		5.78(0.09)	5.33(0.10)	5.15(0.12)	4.97(0.13)
APSO[29]		4.21 (0.02)	4.11 (0.03)	4.12(0.04)	4.13(0.06)
PSO-AQ[30]		3.98(0.09)	3.87(0.11)	3.98(0.13)	3.66(0.14)
Memetic+		1.30(0.32)	2.70(0.42)	**2.68(0.21)**	3.22(0.4)
HmSO[31]		4.46(0.26)	4.61(0.07)	4.66(0.12)	4.83(0.09)
DynDE[32]		16.84(9.39)	4.25(0.05)	5.34(0.02)	5.80(0.04)
Cellular Pso[18]		7.97(0.54)	5.94(0.23)	6.13(0.17)	6.23(0.15)
FTMPSO[33]		**0.89(0.05)**	**2.36(0.09)**	3.01(0.12)	**3.06(0.10)**

نتایج نشان دهنده آن است که با افزایش فرکانس تغییر، یعنی افـزایش فاصـله تغییرات از هم، الگوریتم زمان بیشتری برای جسـتجو داشـته و کـارایی بـالاتری از خود نشان می‌دهد. نتایج جداول بالا نشان دهنده کارایی بالای الگوریتم پیشنهادی نسبت به الگوریتم های دیگر بیان شده می‌باشد.

جدول ۷ هم مقایسـاتی دیگـری بـا روش‌هـای HmSO[۳۱]، DynDE[۳۲]، Cellular Pso[۱۸] و FTMPSO[۳۳] در دو فرکانس و تعداد قلـه‌هـای متفـاوت آمده است. نتایج حاصل از مقایسات به صورت نمودار میله‌ای برای خطـای آفلایـن در شکل ۱۴ ارائه شده است.



شکل ۱۴- نمودار میله‌ای مقایسه الگوریتم‌ها برمبنای خطای آفلاین

نتایج مشاهده شده در جدول ۷ و شکل ۱۴ نشان دهنده این اصل است که در فرکانس‌های پایین و تغییرات بیشتر کارایی الگوریتم بهتر می‌باشد کـه ایـن نقطـه قوت این الگوریتم می‌باشد.

۷- نتیجه‌گیری

در این مقاله یک الگوریتم جدید برای بهینه‌یابی محیط پویا که بر پایـه ترکیـب دو الگوریتم ممتیک و ازدحام ذرات می‌باشد، پیشـنهاد گردیـد. الگـوریتم پیشـنهادی دارای افزایش سرعت همگرایی و کاهش قرار گـرفتن در بهینـه محلـی مـی‌باشـد. همچنین با فازی‌سازی پارامترهای الگوریتم، سرعت همگرایی افـزایش پیـدا کـرده است. در مقایسه با الگوریتم‌هـای ممتیـک و ازدحـام ذرات و امثـال آنهـا بصـورت منفرد، این الگوریتم کارایی قابل قبولی از خود نشان داده است. ولـی هنـوز امکـان افزایش سرعت همگرایی وجود دارد. امکان اسـتفاده از روش‌هـایی از قبیـل نظریـه آشوب[۲۶] برای تولید جمعیت اولیه همچنان می‌توانند سـرعت همگرایـی را افـزایش دهند.

مراجع

[1] C. Cotta, A. J. Fernandez, and J. E. Gallardo, "On the Hybridization of Memetic Algorithms with Branch-and-

[15] T. Blackwell, J. Branke, and X. Li, "Particle Swarms for Dynamic Optimization Problems," *Swarm Intelligence*, pp. 193-217, 2008.

[16] T. Blackwell, and P. J. Bentley, "Dynamic Search with Charged Swarms," *Proc, Int'l Conf. Genetic and Evolutionary Computation*, pp. 19-26, 2002.

[17] T. Blackwell, and J. Branke, "Multiswarms, Exclusion, and Anti-convergence in Dynamic Environments," *IEEE Trans. Evolutionary Computation*, vol. 10, no. 5, pp. 459-472, 2006.

[18] A. B. Hashemi, and M. R. Meybodi, "Cellular PSO: A PSO for Dynamic Environments," *Advances in Computation and Intelligence*, pp. 422-433, 2009.

[19] A. B. Hashemi, and M. R. Meybodi, "A Multi-role Cellular PSO for Dynamic Environments," *Proc, Int'l Conf. CSI Computer*, pp. 412-417, 2009.

[20] C. Li, and S. Yang, "Fast Multi-Swarm Optimization for Dynamic Optimization Problems," *Proc, Int'l Conf. Natural Computation*, vol. 7, no. 3, pp. 624-628, 2008.

[21] S. Yang, and C. Li, "A Clustering Particle Swarm Optimizer for Locating and Tracking Multiple Optima in DynamicEnvironments," *IEEE Trans. Evolutionary Computation*, vol. 14, no. 6, pp. 959-974, 2010.

[22] V. Noroozi, A. B. Hashemi, and M. R. Meybodi," Cellular DE: A Cellular Based Differential Evolution for Dynamic Optimization Problems," *Adaptive and Natural Computing Algorithms*, pp. 340-349, 2011.

[23] Y. Jin, and J. Branke, "Evolutionary Optimization in Uncertain Environments-ASurvey," *IEEE Trans. Evolutionary Computation*, vol. 9, no. 3, pp. 303-317, 2005.

[24] S. Kumar, and D. K. Chaturvedi, "Tuning of Particle Swarm Optimization Parameter Using Fuzzy Logic," *Proc, Int'l Conf. Communication Systems and Network Technologies*, pp.174-179, 2011.

[25] P. Yadmellat, S. M. A. Salehizadeh, and M. B. Menhaj, "Fuzzy Parameter Particle Swarm Optimization," *Proc, Int'l Conf. Intelligent Networks and Intelligent Systems*, pp. 93-98, 2008.

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

[27] H. Xiaohui, and R. C. Eberhart, "Adaptive Particle Swarm Optimization: Detection and Response to Dynamic Systems," *Proc, Cong. Evolutionary Computation*, vol. 2, no. 1, pp. 1666-1670, 2002.

[28] M. Kamosi, A. B. Hashemi, and M. R. Meybodi, "A New Particle Swarm Optimization Algorithm for Dynamic Environments," *Int'l Conf. Swarm, Evolutionary, and Memetic Computing*, pp. 129-138, 2010.

Bound Techniques," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 37, no. 1, pp. 77-83, 2007.

[2] X. Chen, Y. Ong, M. Lim, and K. Tan, "A Multi-facet Survey on Memetic Computation," *IEEE Trans. Evolutionary Computation*, vol. 15, no. 5, pp. 591-607, 2011.

[3] Y. Ong, M. Lim, N. Zhu, and K. Wong, "Classification of Adaptive Memetic Algorithms: AComparative Study," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 36, no. 1, pp. 141-152, 2006.

[4] F. G. Guimaraes, F. Campelo, H. Igarashi, D. A. Lowther, and J.A. Ramirez, "Optimization of Cost Functions Using Evolutionary Algorithms with Local Learning and Local Search," *IEEE Trans. Magnetics*, vol. 43, no. 4, pp. 1641-1644, 2007.

[5] J. Branke, *Evolutionary Optimization in Dynamic Environments*, Kluwer Academic, 2001.

[6] H. Wang, S. Yang, W. H. Ip, and D. Wang, "A Particle Swarm Optimization based Memetic Algorithm for Dynamic Optimization Problems," *Natural Computing*, vol. 9, no. 3, pp. 703-725, 2010.

[7] H. Wang, D. Wang, and S. Yang, "A Memetic Algorithm with Adaptive Hill Climbing Strategy for Dynamic Optimization Problems," *Journal of Soft Computing*, vol. 13, no. 8, pp. 763-780, 2008.

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

[9] M. Akhtari, and M. R. Meybodi, "Memetic-CLA-PSO: A Hybrid Model for Optimization," *Proc, Int'l Conf. Modeling and Simulation*, pp. 20-25, 2011.

[10] N. Baktash, and M. R. Meybodi, "A New Hybrid Model of PSO and ABC Algorithms for Optimization in Dynamic Environment," *Int'l Journal of Computing Theory Engineering*, vol. 4, no. 3, pp. 362-364, 2012.

[11] R. I. Lung, and D. Dumitrescu, "A Collaborative Model for Tracking Optima in Dynamic Environments," *Proc, IEEE Cong. Evolutionary Computation*, pp. 564-567, 2007.

[12] R. I. Lung, and D. Dumitrescu, "Evolutionary Swarm Cooperative Optimization in Dynamic Environments," *Natural Computing*, vol. 9, no. 1, pp. 83-94, 2010.

[13] L. Liu, D. Wang, and J. Tang, "Composite particle optimization with hyper-reflection scheme in dynamic environments," *Applied Soft Computing*, vol. 11, no. 8, pp. 4626-4639, 2011.

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

[29] I. Rezazadeh, M. R. Meybodi, and A. Naebi, "Adaptive Particle Swarm Optimization Algorithm for Dynamic Environments," *Journal of Advances in Swarm Intelligence*, vol. 67, no. 28, pp. 120-129, 2011.

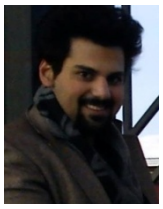
[30] D. Yazdani, B. Nasiri, R. Azizi, A. Sepas-Moghadam, and M. R. Meybodi, "Optimization in Dynamic Environment Utilizing A Novel Method based on Particle Swarm Optimization," *Int'l Journal of Artificial Intelligence*, vol. 11, no. 13, pp. 170-192, 2013.

[31] M. Kamosi, A. B. Hashemi, and M. R. Meybodi, "A Hibernating Multi-Swarm Optimization Algorithm for Dynamic Environments," *Proc, World Cong. Nature and Biologically Inspired Computing*, pp. 370-376, 2010.

[32] R. Mendes, and A. Mohais, "DynDE: a Differential Evolution for Dynamic Optimization Problems," *Proc. IEEE Cong. Evolutionary Computation*, pp. 2808-2815, 2005.

[33] D. Yazdani, B. Nasiri, A. Sepas-Moghadam, and M. R. Meybodi, "A Novel Multi-swarm Algorithm for Optimization in Dynamic Environments Based on Particle Swarm Optimization," *Journal of Applied Soft Computing*, vol. 13, no. 4, pp. 2144-2158, 2013.

[34] S. Nabizadeh, A. Rezvanian, and M. R. Meybodi, "Tracking Extrema in Dynamic Environment using Multi-Swarm Cellular PSO with Local Search," *Int'l Jornal of Electronics and Informatics* ,vol. 1, no. 1, pp. 29-37, 2012.



مرتضی علی‌زاده کارشناسی خود را در سال ۱۳۸۹ در دانشگاه علوم و فنون مازندران در رشته مهندسی کامپیوتر گرایش نرم‌افزار به پایان رسانده و از سال ۱۳۸۹ موفق به ادامه تحصیل در مقطع کارشناسی‌ارشد در دانشگاه آزاد اسلامی قزوین در گرایش هوش مصنوعی شده است. وی هم‌اکنون به عنوان دانشجوی دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه آزاد قزوین مشغول به تحقیق می‌باشد. زمینه‌های تحقیقاتی ایشان شامل: الگوریتم‌های تکاملی، سیستم‌های فازی و کاربرد آنها در مهندسی می‌باشد.

آدرس پست‌الکترونیکی ایشان عبارت است از:

m_alizadeh@qiau.ac.ir

محمدرضا میبدی در سال ۱۳۳۱ در اراک بدنیا آمد. وی تحصیلات دانشگاهی خود را در دانشگاه شهید بهشتی در رشته اقتصاد آغاز و تا سطح کارشناسی‌ارشد در آنجا به تحصیل پرداخت و در سال ۱۳۵۶ موفق به اخذ مدرک کارشناسی‌ارشد شد. وی در سال ۱۳۶۲ موفق به اخذ دکتری رشته علوم کامپیوتر از دانشگاه اوکلاهمای آمریکا شد و هم‌اکنون عضو هیئت علمی و استاد دانشگاه صنعتی امیرکبیر می‌باشد. مشاغل و سمت‌های اداری و مدیریتی دکتر محمدرضا میبدی به ترتیب زیر است: رئیس و عضو کمیته مهندسی کامپیوتر در وزارت علوم تحقیقات و فناوری از سال ۱۳۷۵ تا کنون، عضو کمیته داوری برای چهاردهمین جشنواره خوارزمی در وزارت علوم تحقیقات و فناوری از سال ۱۳۷۵ تا کنون، عضو کمیته علمی در مرکز تحقیقات ریاضیات و فیزیک نظری وزارت علوم تحقیقات و فناوری از سال ۱۳۷۷ تا ۱۳۷۹، عضو کمیته چند رشته‌ای وزارت علوم تحقیقات و فناوری از سال ۱۳۷۵ تا کنون، عضو کمیته پژوهشی وزارت علوم تحقیقات و فناوری از سال ۱۳۸۲ تا سال ۱۳۸۳، عضو کمیته عملی در مرکز تحقیقات علوم پایه وزارت علوم تحقیقات و فناوری از سال ۱۳۸۱ تا کنون، عضو کمیته انتخاب بهترین کتاب ایرانی وزارت فرهنگ و ارشاد اسلامی از سال ۱۳۷۵ تا کنون.

فعالیت‌های آموزشی و عنوان دروس ارائه شده توسط محمدرضا میبدی به قرار زیر است: نظریه زبان‌ها و ماشین‌ها، تحلیل و طراحی الگوریتم‌های موازی، ساختمان داده، تحلیل و طراحی الگوریتم، محاسبات موازی.

آدرس پست‌الکترونیکی ایشان عبارت است از:

mmeybodi@aut.ac.ir

علیرضا رضوانیان کارشناسی خود را در سال ۱۳۸۶ در دانشگاه بوعلی سینا همدان در رشته مهندسی کامپیوتر گرایش نرم‌افزار به پایان رسانده و در سال ۱۳۸۹ موفق به اخذ مدرک کارشناسی‌ارشد از دانشگاه آزاد اسلامی قزوین در گرایش هوش مصنوعی شده است. وی هم‌اکنون به عنوان دانشجوی دکتری در آزمایشگاه محاسبات نرم دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران) مشغول به تحقیق می‌باشد. زمینه‌های تحقیقاتی ایشان شامل: محاسبات نرم، الگوریتم‌های تکاملی، پردازش تصویر، شبکه‌های پیچیده و تحلیل شبکه‌های اجتماعی می‌باشد.

آدرس پست‌الکترونیکی ایشان عبارت است از:

a.rezvanian@aut.ac.ir

اطلاعات بررسی مقاله:

تاریخ ارسال: ۹۲/۱/۳۰

تاریخ اصلاح: ۹۲/۵/۱۲

تاریخ قبول شدن: ۹۲/۵/۲۴

نویسنده مرتبط: مرتضی علی‌زاده، دانشکده مهندسی برق، رایانه و فناوری اطلاعات، دانشگاه آزاد اسلامی واحد قزوین، قزوین، ایران.

[1]Hybridization Strategy
[2]Local Search
[3]Hill Climbing
[4]Baldwin
[5] Lamarck
[6]Muscato
[7] Memetic
[8]Evolutionary Algorithm
[9] Fitness
[10]Self-Organized Random Immigrants
[11]Cellular Automata
[12]Differential Evolution
[13]Compound PSO
[14]Charged PSO
[15]Cellular PSO
[16]Fast PSO
[17] Dynamic Environment
[18]Restriction
[19]Exploitation
[20]Exploration
[21]Neri
[22]Particle Swarm Optimization
[23]Itration
[24]Current Error
[25]Offline Error
[26]Chaotic