Journal of Applied And Basic Machine Intelligence Research (ABMIR)

پژوهش‌های نظری و کاربردی هوش ماشینی

سال دوم، شماره دوم، پاییز و زمستان ۱۴۰۳، صفحات ۱۶۱ - ۱۴۵

تنبیه: عملگری جدید برای کنترل فشار انتخاب و بهبود کارایی الگوریتم کپک مخاطی

آرزو رحیمی[۱]، محمد فرشی[۲]، سپهر ابراهیمی مود[۳]*

[۱] دانشجوی کارشناسی ارشد گروه علوم کامپیوتر، دانشکده علوم ریاضی، دانشگاه یزد، یزد، ایران

[۲] دانشیار گروه علوم کامپیوتر، دانشکده علوم ریاضی، دانشگاه یزد، یزد، ایران

[۳] استادیار گروه علوم کامپیوتر، دانشکده علوم ریاضی، دانشگاه یزد، یزد، ایران

مقاله پژوهشی

تاریخ دریافت:
۱۴۰۳/۱۲/۵
تاریخ پذیرش:
۱۴۰۴/۲/۱۶

کلیدواژه‌ها:
الگوریتم کپک مخاطی، فشار انتخاب، عملگر تنبیه، بهینه‌سازی

نویسنده‌مسئول:
s.ebrahimi@yazd.ac.ir

چکیده

در این مقاله، الگوریتم کپک مخاطی (SMA) که از رفتار بیولوژیکی کپک‌های مخاطی الهام گرفته‌شده است، به‌عنوان روشی فرابتکاری قدرتمند برای حل مسائل پیچیده بهینه‌سازی بررسی و ارزیابی‌شده است. یکی از چالش‌های اصلی این الگوریتم، همگرایی زودرس ناشی از عدم کنترل فشار انتخاب است. به‌منظور رفع این مشکل، عملگر تنبیه ارائه‌شده است تا فشار انتخاب را کنترل کند وتنوع جمعیت را حفظ کند. این عملگر برخلاف روش‌های دیگر که فشار انتخاب را حین فرآیند محاسباتی الگوریتم کنترل نمی‌کنند، به‌صورت پویا رفتار خود را براساس شرایط فعلی الگوریتم تنظیم می‌کند. تا در مواقعی که ذرات دچار همگرایی زودرس می‌شوند، یک نیروی دافعه به آن‌ها اعمال می‌کند که از بهینه‌محلی رهایی یابند و کاوش بهتری در فضای جستجو انجام دهند. آزمایش‌های گسترده‌ای بر روی ۲۳ تابع آزمایشی CEC2017، که شامل توابع مختلفی از قبیل تک‌حالته، چندحالته، ترکیبی و پیچیده است، انجام‌شده تا عملکرد عملگر پیشنهادی در شرایط پیچیده و متنوع ارزیابی شود. نتایج حاصل نشان می‌دهد که SMAی بهبودیافته توانسته است در مقایسه با نسخه اصلی روی توابع تست استاندارد کارایی الگوریتم را ۳۵/۵٪ بهبود دهد. نتایج شبیه‌سازی و آزمایش‌ها در این پژوهش نشان‌دهنده کارایی عملگر پیشنهادی در کنترل فشار انتخاب و درنتیجه کارایی بهتر الگوریتم کپک مخاطی بهبودیافته در حل مسائل بهینه‌سازی پیچیده است و می‌تواند در زمینه‌های گسترده‌ای از علوم و مهندسی به‌طور مؤثر و کارآمد استفاده شود.

doi : 10.22034/ABMIR.2025.22831.1104

E-ISSN: 2821-2037

۱- مقدمه

الگوریتم‌های تکاملی به روش‌های جستجوی تصادفی مبتنی بر جمعیت گفته می‌شود که از تکامل طبیعی الهام می‌گیرند. این الگوریتم‌ها با ایجاد جمعیتی اولیه از نقاط تصادفی در فضای جستجو شروع می‌شوند، که هر نقطه به‌عنوان یک عامل جستجو (نامزد حل) در نظر گرفته می‌شود [۱]. الگوریتم‌های تکاملی روش‌های بهینه‌سازی هستند که برای حل مسائل پیچیده بهینه‌سازی طراحی‌شده‌اند و در بین محققان دانشگاهی در حوزه‌های مختلفی ازجمله مهندسی، علوم، اقتصاد، صنعت، نظامی، پزشکی و هوش‌مصنوعی برای یافتن راه‌حل‌های تقریبی مناسب به کار می‌روند، به‌ویژه در مسائلی که جستجو در فضای بزرگ و ناشناخته با روش‌های قطعی بسیار زمان‌بر یا غیرممکن است [۲]. الگوریتم‌های تکاملی شامل چهار گام اصلی هستند: ۱. تولید جمعیت اولیه: در ابتدا، مجموعه‌ای از نقاط به‌صورت تصادفی تولید و ارزیابی می‌شود. ۲. انتخاب و تغییر: با انتخاب اعضای مناسب‌تر از جمعیت اولیه و اعمال تغییرات تصادفی، مجموعه‌ای جدید از نقاط تولید و ارزیابی می‌شود. ۳. جایگزینی: در این مرحله، برخی از اعضای جمعیت قبلی با اعضای جدید جایگزین می‌شوند. ٤. تکرار فرآیند: اگر شرط توقف برقرار نبود، فرآیند با بازگشت به مرحله دوم تکرار می‌شود [۳]. هر الگوریتم جستجو باید به کاوش و بهره‌وری از فضای جستجو بپردازد. کاوش، جستجوی کامل فضای‌شدنی مسئله، با هدف پیداکردن فضاهای جدید و راه‌حل‌های جدید است، در حالیکه بهره‌وری جستجوی پیرامون بهترین پاسخ پیدا شده، به‌منظور بهبود کیفیت آن و درواقع همگرایی الگوریتم است [٤]. هدف از کاوش این است که الگوریتم بتواند به‌دنبال راه‌حل‌های جدید و متنوع بگردد و از گرفتاری در بهینه‌های محلی جلوگیری کند. هدف از بهره‌وری این است که الگوریتم روی نواحی امیدوارکننده تمرکز کند و سعی کند بهترین راه‌حل ممکن را در همان نواحی پیدا کند. برای موفقیت در الگوریتم‌های بهینه‌سازی، تعادل بین کاوش و بهره‌وری بسیار مهم است [۵]. زمانی که توانایی کاوش الگوریتم بیشتر باشد، این الگوریتم قادراست فضای راه‌حل را بررسی کرده و مجموعه‌های متمایزتری از راه‌حل‌ها تولید کند. اما هنگامی که توانایی بهره‌وری بیشتری دارد، جستجو به‌صورت محلی‌تر انجام می‌شود تا کیفیت و دقت راه‌حل‌ها افزایش یابد [٦]. فشار انتخاب، تأثیر پاسخ‌های بهتر در راهبری عوامل جستجوی جمعیت و همچنین تعریف نسل بعد جمعیت است. اگر فشار انتخاب بالا باشد، جمعیت در تکرار بعد شبیه به اعضای با کیفیت بالا در نسل فعلی خواهد بود. این می‌تواند به سرعت همگرایی به بهترین جواب‌ها کمک کند، اما ممکن است تنوع جمعیت را کاهش دهد و خطر قرار گرفتن در نقاط محلی بهینه را افزایش دهد و باعث افزایش بهره‌وری می‌شود، همچنین اگر فشار انتخاب پایین باشد، اعضای ضعیف جمعیت فعلی هم در تولید جمعیت تکرار بعد مشارکت خواهند داشت. [۳].

الگوریتم بهینه‌سازی کپک مخاطی (SMA) یک روش فراابتکاری است که در سال ۲۰۲۰ و از رفتار کپک مخاطی در طبیعت الهام گرفته است [٦]. این الگوریتم با وجود کارایی بسیار خوب در حل مسائل بهینه‌سازی، تعادل مناسبی در قابلیت کاوش و بهره‌وری و همچنین کنترل فشار انتخاب ندارد [۷]. الگوریتم SMA در تنظیم فشار انتخاب ناتوان است و نتوانسته این مشکل را که منجر به همگرایی زودرس می‌شود، حل کند. دلیل این ضعف، تمایل بیش‌ازحد ذرات به دنبال کردن بهترین جواب‌های فعلی است که باعث کاهش تنوع جمعیت و گرفتار شدن در بهینه‌های محلی می‌شود. درواقع، SMA فاقد یک سازوکار قوی برای کنترل فشار انتخاب است و نمی‌تواند تعادل مناسبی بین کاوش و بهره‌وری برقرار کند. همین موضوع باعث می‌شود که عملکرد آن در مسائل پیچیده و چندحالته کاهش یابد و در برخی موارد به سرعت همگرا شود، بدون اینکه فضای جستجو را به‌طور مؤثر کاوش کند [۸]. در روش پیشنهادی یک عملگر جدید، برای حل مشکل همگرایی زودرس طراحی و به الگوریتم SMA اضافه‌شده است. این عملگربراساس وضعیت همگرایی جمعیت، فشار انتخاب را تنظیم می‌کند و در مواقعی که احتمال گرفتار شدن الگوریتم در بهینه‌های محلی زیاد است، یک نیروی دافعه به ذرات وارد می‌کند. این فرآیند باعث افزایش تنوع جمعیت شده و به ذرات کمک می‌کند مسیرهای جدید را در فضای جستجو بررسی کنند. از طرف دیگر، تعادل بین کاوش و بهره‌وری را بهبود می‌بخشد و دقت الگوریتم را ارتقا

می‌دهد. برخلاف الگوریتم SMA که کنترلی روی فشار انتخاب نیست، در این روش فشار انتخاب براساس شرایط تنظیم‌شده و در هر مرحله بهینه‌سازی مقدار آن تغییر می‌کند.

هدف این مقاله، معرفی یک عملگر جدید به‌منظور کنترل فشار انتخاب است. همچنین کارایی این الگوریتم با سایر الگوریتم‌ها مقایسه و ارزیابی‌شده و نتایج این مقایسه نشان‌دهنده کارایی بالای الگوریتم پیشنهادی در مقایسه با سایر الگوریتم‌ها است. این آزمایش‌ها گسترده بر روی مجموعه توابع آزمایشی CEC2017، که شامل توابع مختلفی از قبیل توابع تک‌حالته و چندحالته است، انجام‌شده است.

ساختار ادامه مقاله به شرح زیر است: بخش ۲ به ارائه کارهای مرتبط اختصاص دارد، در بخش ۳ مفهوم الگوریتم کپک مخاطی و مفاهیم و مدل‌سازی ریاضی این الگوریتم بیان‌شده است. بخش ٤ به معرفی الگوریتم پیشنهادی می‌پردازد. در بخش ۵ یک تجزیه‌وتحلیل کیفی از الگوریتم ارائه‌شده است و مقایسه جامعی از عملکرد آن بر روی ۲۳ تابع آزمایشی انجام‌شده است، همچنین، نتایج الگوریتم با سایر الگوریتم‌ها مقایسه شده است و درنهایت، بخش ٦ به نتیجه‌گیری کل پژوهش می‌پردازد.

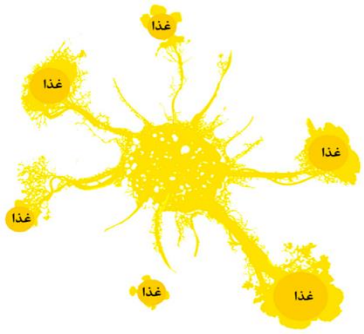
۲- کارهای مرتبط

الگوریتم‌های تکاملی و فراابتکاری به دلیل انعطاف‌پذیری بالا و توانایی حل مسائل پیچیده، نقش مهمی در بهینه‌سازی فرآیندهای صنعتی ایفا می‌کنند. این روش‌ها به‌طور گسترده در صنایعی مانند خودروسازی، هوافضا، انرژی، عمران، و تولید به کار گرفته‌شده‌اند. [۹]. در صنعت انرژی، این روش‌ها برای بهینه‌سازی توان واکنشی در شبکه‌های قدرت و کاهش تلفات انرژی در سیستم‌های توزیع به کار گرفته‌شده‌اند. همچنین، در صنایع تولید و برنامه‌ریزی، استفاده از الگوریتم‌های تکاملی در زمان‌بندی پروژه‌ها و بهینه‌سازی خطوط تولید منجر به کاهش هزینه‌ها و افزایش کارایی شده است [۱۰]. الگوریتم‌های فراابتکاری معمولاً از پدیده‌های طبیعی الهام گرفته‌اند تا با شبیه‌سازی قوانین فیزیکی یا فرایندهای زیستی، راه‌حل‌های بهینه‌ای برای مسائل پیچیده بهینه‌سازی ارائه دهند. همچنین در طیف گسترده‌ای از مسائل در حوزه‌های مختلف مهندسی، بهینه‌سازی مسائل چند هدفه، داده‌کاوی و یادگیری ماشین مورداستفاده هستند [۱۱]. بسطامی و دولتشاهی در سال ۲۰۲٤ با استفاده از الگوریتم جستجوی گرانشی، معماری‌های بهینه شبکه عصبی کانولوشنی را برای طبقه‌بندی تصاویر طراحی کردند و با کاهش هزینه محاسباتی و افزایش دقت، عملکرد بهتری نسبت به روش‌های موجود ارائه دادند [۱۲]. محمدی و همکاران در سال ۲۰۲٤ از الگوریتم‌های پیشرفته مانند الگوریتم بهینه‌سازی ازدحام ذرات برای خوشه‌بندی و الگوریتم بهینه‌سازی کرم شب‌تاب برای شناسایی مسیرهای بهینه استفاده کردند. این رویکردها در بهبود عملکرد سیستم‌های مبتنی بر اینترنت اشیا و کاهش تأخیرها و هزینه‌های امنیتی تأثیرگذار بودند [۱۳]. الگوریتم‌های فراابتکاری به دودسته اصلی تقسیم می‌شوند: روش‌های مبتنی بر ازدحام و الگوریتم‌های تکاملی. دسته اول، عمدتاً پدیده‌های فیزیکی را شبیه‌سازی می‌کنند و از قوانین یا روش‌های ریاضی برای بهینه‌سازی استفاده می‌کند. مثال‌هایی از این دسته الگوریتم عبارت‌اند از: الگوریتم جستجوی گرانشی [۱٤]، الگوریتم سینوس کسینوس [۱۵] و الگوریتم مبتنی برآموزش-یادگیری [۱٦]. دسته دوم که همان روش‌های الهام گرفته از طبیعت هستند نیز به دو گروه عمده تقسیم می‌شوند: روش‌های تکاملی و روش‌های هوش ازدحامی. روش‌های تکاملی از فرآیندهای تکاملی زیستی در طبیعت الهام گرفته‌اند. برخی از الگوریتم‌های معروف در این دسته عبارت‌اند از: الگوریتم ژنتیک [۱۷] و الگوریتم تکامل تفاضلی [۱۸]. روش‌های هوش ازدحامی بر اساس شبیه‌سازی رفتار گروهی و خودسازمان‌دهی سیستم‌های زیستی در طبیعت شکل‌گرفته است. این الگوریتم‌ها با الهام از هوش جمعی و رفتار هماهنگ موجودات زنده، به‌منظور دستیابی به اهداف مشخص طراحی‌شده‌اند. برخی از الگوریتم‌های معروف در این دسته عبارت‌اند از: الگوریتم بهینه‌سازی ازدحام ذرات [۱۹] و الگوریتم بهینه‌سازی گرگ خاکستری [۲۰]. الگوریتم بهینه‌سازی کپک مخاطی یک روش فراابتکاری است که دردسته روش‌های هوش ازدحامی قرارگرفته است که توسط لی و همکاران [٦] در سال ۲۰۲۰ ارائه‌شده و از رفتار کپک مخاطی در طبیعت الهام گرفته است. گورسس و همکاران در سال ۲۰۲۱ در پژوهش [۲۱] یک رویکرد برای بهینه‌سازی سراسری و انتخاب ویژگی معرفی کردند که یکی از

کاربردهای عملی این رویکرد، کاهش وزن خودروها برای کاهش مصرف سوخت و آلایندگی است. تانگ و همکاران در سال ۲۰۲۱ در پژوهش [۲۲] راهبرد یادگیری مبتنی بر آشوب و تضاد برای افزایش تنوع جمعیت را معرفی کردند که قابلیت آن را در مسائل بهینه‌سازی مهندسی واقعی نیز موردبررسی قرار داده است. این روش می‌تواند در زمینه‌هایی مانند طراحی سازه‌ها، بهینه‌سازی سیستم‌های مهندسی، مورداستفاده قرار گیرد. کریشنا و همکاران در سال ۲۰۲۱ در پژوهش [۲۳] رویکردی برای چالش‌های طراحی عددی و مهندسی معرفی کردند که در حل مسائل بهینه‌سازی در طراحی مهندسی است. این رویکرد می‌تواند در چالش‌هایی مانند بهینه‌سازی سیستم‌ها و ساختارهای مهندسی که دارای پیچیدگی‌های بالایی هستند، به‌ویژه در چالش‌های بهینه‌سازی مهندسی واقعی، استفاده شود. یسری و همکاران در سال ۲۰۲۱ در پژوهش [۲٤] رویکردی برای مقایسه انواع مختلف ماژول‌های فتوولتایی ارائه کردند که حل مسائل بهینه‌سازی پیچیده و چندبعدی در مدل‌سازی سیستم‌های فتوولتایی با شرایط سایه‌زنی جزئی، با کارایی بالا و مقاوم در برابر تغییرات محیطی و تابش‌های مختلف استفاده شود. نایک و همکاران در سال ۲۰۲۱ در پژوهش [۲۵] مفهوم بهینه‌ساز تعادل را در SMA معرفی کردند و به SMA توانایی پرش از بهینه محلی دادند. حسین و همکاران در سال ۲۰۲۱ در پژوهش [۲٦] رویکردی به‌منظور بهبود عملکرد SMA از طریق تقویت مرحله جستجوی محلی و جلوگیری از همگرایی زودهنگام معرفی کردند. این روش، در مسائلی همچون بهینه‌سازی ترکیبی و مشکلات طراحی مهندسی مانند طراحی فنر کششی، مخزن فشار آزمایش‌شده است. عبدالباسط و همکاران در سال ۲۰۲۰ در پژوهش [۲۷] رویکردی که بر تخصیص بهینه و تعیین اندازه بانک‌های خازنی و تولیدات پراکنده به‌طور هم‌زمان تأکید دارد، همچنین بازآرایی بهینه سیستم توزیع شعاعی را برای رفع مشکلاتی از قبیل افزایش تلفات خطوط و انحراف ولتاژ شامل می‌شود را ارائه دادند. اویس و همکاران در سال ۲۰۲۱ در پژوهش [۲۸] رویکردی درزمینه بهینه‌سازی و پیش‌پردازش داده برای افزایش کارایی الگوریتم‌های یادگیری ماشین و انتخاب ویژگی‌های مؤثر ارائه دادند. یکی از مشکلات اصلی الگوریتم‌های تکاملی همگرایی زودرس است، که به معنای گرایش جمعیت به یک راه‌حل شبه‌بهینه به‌جای راه‌حل بهینه واقعی است [۲۹]. همچنین مشکل دیگر این الگوریتم‌ها تعادل بین کاوش و بهره‌وری است که می‌توان با کنترل فشار انتخاب این مشکل را حل کرد ولی با وجود بهبودهایی که تاکنون انجام‌شده است هنوز فشار انتخاب قابل‌کنترل نیست [۷]. بنابراین در این مقاله برای رفع این مشکل، یک عملگر جدید تعریف و به الگوریتم اضافه‌شده است که ذرات را از تمرکز بیش‌ازحد روی بهترین موقعیت‌ها دور کرده و مانع از گرفتار شدن آن‌ها در بهینه‌های محلی می‌شود. این عملگر با افزایش تنوع جمعیت، تعادل بین کاوش و بهره‌برداری را ایجاد کرده و کارایی الگوریتم دریافتن پاسخ بهینه را ارتقا می‌دهد. این عملگر در مواقعی که ذرات دچار همگرایی زودرس می‌شوند، یک نیروی دافعه به آن‌ها اعمال می‌کند تا از بهینه‌محلی رهایی یابند و کاوش بهتری در فضای جستجو انجام دهند. با اضافه شدن عملگر تنبیه به الگوریتم پیشنهادی، فشار انتخاب به‌صورت پویا در زمان اجرا کنترل خواهد شد.

۳- الگوریتم SMA

الگوریتم کپک مخاطی رفتار جستجوی غذا توسط کپک مخاطی را شبیه‌سازی می‌کند. کپک مخاطی در مکان‌های تاریک و مرطوب مثل جنگل‌ها و زیر برگ‌ها زندگی می‌کند. مرحله اصلی تغذیه، مرحله فعال کپک مخاطی و همچنین مرحله اصلی این مقاله است. در این مرحله مواد آلی در کپک به دنبال غذا می‌گردند، آن را احاطه می‌کنند و برای هضم آن آنزیم‌هایی ترشح می‌کنند. همان‌طور که در شکل (۱) نشان داده‌شده است، در طی مرحله مهاجرت، قسمت جلویی به شکل یک فن گسترش می‌یابد، به دنبال آن یک شبکه وریدی به‌هم‌پیوسته ایجاد می‌شود که جریان سیتوپلاسم در آن برقرار است. این جریان امکان اتصال منابع غذایی را فراهم می‌کند و اگر غذای کافی در محیط وجود داشته باشد، کپک حتی می‌تواند تا بیش از ۹۰۰ سانتی‌متر مربع رشد کند [۳۰].



شکل (۱): مورفولوژی علوفه‌شناسی کپک مخاطی

می‌دانیم که وقتی یک ورید به منبع غذایی نزدیک می‌شود، نوسان‌ساز زیستی یک موج انتشاری تولید می‌کند که جریان سیتوپلاسمی در ورید را زیاد می‌کند، و هر چه سیتوپلاسم سریع‌تر جریان یابد، ورید ضخیم‌تر می‌شود. جریان سیتوپلاسم، که ماده شفاف و پرکننده فضای داخلی کپک است، ضخامت وریدها را تنظیم می‌کند با افزایش جریان، وریدها ضخیم‌تر و با کاهش جریان نازک‌تر می‌شوند. این فرآیند به کپک کمک می‌کند بهترین مسیرها را برای دسترسی به منابع غذایی ایجاد کند [۳۱]. سه همبستگی بین تغییرات مورفولوژیکی ساختار وریدی و حالت انقباض کپک مخاطی وجود دارد:

۱) انقباض متفاوت از حالت خارج: اگر سرعت انقباض در نقاط مختلف یکسان نباشد، رگه‌های ضخیم به‌صورت شعاعی از مرکز به بیرون تشکیل می‌شوند.

۲) انقباض ناپایدار: در حالت انقباض نامنظم، ساختار شبکه‌ای به‌هم می‌ریزد وکپک شکل نامتقارن به خود می‌گیرد.

۳) الگوی انقباض نامرتب: اگر انقباض الگوی خاصی نداشته باشد، ساختار وریدی کاملاً از بین می‌رود [۳۰].

این تغییرات، به کپک اجازه می‌دهند مسیرهای محکم و بهینه‌ای برای دسترسی به منابع غذایی ایجاد کند. کپک می‌تواند به‌صورت طبیعی بهترین منابع غذایی را شناسایی کند و از مسیری که بیشترین غلظت مواد غذایی در آن موجود است، استفاده کند [۳۲]. هنگامی که کیفیت منابع مختلف غذایی متفاوت است، کپک منبع غذایی با بالاترین غلظت را انتخاب می‌کند. علاوه براین، کپک باید در جستجوی غذا، تعادلی میان سرعت و دقت برقرار کند. اگرچه تصمیم‌گیری سریع می‌تواند از آسیب‌های زیست محیطی جلوگیری کند، اما ممکن است دقت را کاهش دهد و منجر به شناسایی نادرست منبع اصلی غذا شود. بنابراین، ایجاد تعادل برای بهینه‌سازی جستجوی منابع ضروری است [۳۳].

۳-۱ مدل‌سازی ریاضی

در این بخش، رفتار کپک مخاطی به‌عنوان یک الگوریتم جستجو مدل شده و به‌صورت ریاضی تعریف‌شده است.

نزدیک شدن به غذا

کپک مخاطی از طریق بوی غذا که در هوا پخش می‌شود به سمت منابع غذایی حرکت می‌کند. برای بیان رفتار آن فرمول‌های زیر پیشنهادشده است:

$$\overrightarrow{\mathbf{X(t+1)}} = \begin{cases} \overrightarrow{\mathbf{X_b(t)}} + \overrightarrow{\mathbf{vb}}.\left(\overrightarrow{\mathbf{W}}.\left(\overrightarrow{\mathbf{X_A(t)}} - \overrightarrow{\mathbf{X_b(t)}}\right)\right), & \mathbf{r < p} \\ \overrightarrow{\mathbf{vc}}.\overrightarrow{\mathbf{X(t)}}, & \mathbf{r \geq p} \end{cases} \quad (۱)$$

که در این معادله، $\overrightarrow{vb}$ یک پارامتر کنترلی در بازه $[-a,a]$ است که تعیین می‌کند کپک چقدر سریع به سمت مکان جدید حرکت کند. این مقدار، متناسب با شرایط و تغییرات محیطی، میزان اثر حرکت را تنظیم می‌کند، $\overrightarrow{vc}$ پارامتر خطی است که از یک به صفر کاهش می‌یابد و در طول زمان تغییر می‌کند تا سرعت حرکت کپک به‌مرور کاهش پیدا کند و در موقعیت ثابتی قرار بگیرد، p پارامتر تصمیم‌گیری است، r متغیر تصادفی است، $\overrightarrow{X(t)}$ موقعیت فعلی کپک در لحظه t که در حال جستجوی غذا است را نشان می‌دهد، $\overrightarrow{X_b}$ نشان‌دهنده مکانی است که، کپک در آن‌جا بیشترین بو را استشمام کرده است. درواقع این نقطه به‌عنوان هدف یا بهترین موقعیت کنونی محسوب می‌شود که کپک به آن‌جا نزدیک شود، $\overrightarrow{X}$ نشان‌دهنده موقعیت کپک است، $\overrightarrow{X_A}$ و $\overrightarrow{X_B}$ نشان‌دهنده این است که دو موقعیت تصادفی از میان جمعیت کپک انتخاب‌شده است، $\overrightarrow{W}$ وزنی که برای تفاوت بین دو موقعیت A و B ضرب می‌شود را بیان می‌کند. درواقع $\overrightarrow{W}$ شبیه به عاملی می‌شود که میزان تمایل کپک به سمت بهترین موقعیت را تقویت یا تضعیف می‌کند. اگر $\overrightarrow{W}$ زیاد باشد، کپک با شدت بیشتری به سمت مکان‌های پر غذا حرکت می‌کند و همچنین اگر کم باشد، تمایل کپک به سمت مکان کم غذا است و بیشتر تمایل دارد به مسیرهای پراکنده و جستجوی

نواحی جدید در نوسان باشد. حالت اول که $p > r$ است، کپک به سمت موقعیتی که بوی غذا بیشتر در آن‌جا حس می‌شود حرکت می‌کند و حالت دوم که $p \leq r$ است کپک به‌جای حرکت به سمت بوی غذا، با سرعت کمتری به موقعیت فعلی‌اش نزدیک می‌شود. فرمول p به‌شرح زیر است:

$$p = \tanh|S(i) - DF| \quad (۲)$$

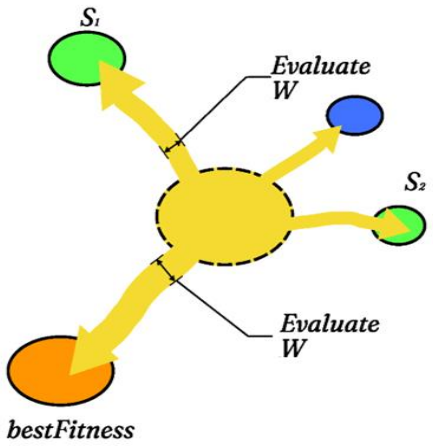
که این فرمول تعیین می‌کند که کپک به کدام یک از دوحالت معادله (۱) برود، به عبارتی وقتی کپک در موقعیت فعلی است نشان‌دهنده این است که کپک چقدر دور یا نزدیک به منبع است. نشان $S(i)$ دهنده کیفیت یا بوی غذا در موقعیت iام است، DF نشان‌دهنده بهترین مقدار کیفیت در تمام تکرارها است یا موقعیتی که کپک در آن‌جا به بهترین نتیجه یا نزدیک‌ترین حالت به غذا را دارد. فرمول $\overrightarrow{vb}$ به‌صورت زیر محاسبه می‌شود:

$$\overrightarrow{vb} = [-a, a] \quad (۳)$$

همچنین فرمول پارامتر a به‌شرح زیر است:

$$a = \operatorname{arctanh}\left(-\left(\frac{t}{max\_t}\right) + 1\right) \quad (٤)$$

a پارامتری است که میزان تغییر و حرکت کپک را تنظیم می‌کند. با افزایش تکرار، این پارامتر تغییر می‌کند و حرکت کپک تنظیم می‌شود. t نشان‌دهنده شماره تکرار فعلی است، max_t حداکثرتعداد تکرارها را نشان می‌دهد. شکل (۲) ارزیابی و محاسبه مقادیر کیفیت کپک را براساس بهترین برازندگی (بهترین مقدار تابع هدف)، وزن کپک و کیفیت هر ذره را به‌تصویر می‌کشد.



شکل (۲): ارزیابی مقادیر کیفیت

$\overrightarrow{W}$ به‌صورت زیر محاسبه می‌شود:

$$\overrightarrow{W(SmellIndex)} = \begin{cases} 1 + r.\log\left(\frac{bF - S(i)}{bF - wF} + 1\right), & condition \\ 1 - r.\log\left(\frac{bF - S(i)}{bF - wF} + 1\right), & others \end{cases} \quad (۵)$$

$$SmellIndex = sort(S) \quad (٦)$$

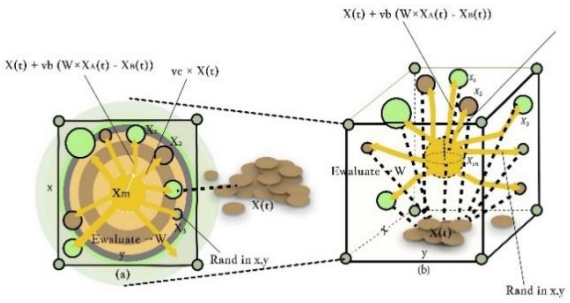
$\overrightarrow{W}$ که بیانگر وزن کپک است، $SmellIndex$ دنباله‌ای از مقادیر تناسبی است که به ترتیب صعودی مرتب شده‌اند، r یک مقدار تصادفی در بازه $[0,1]$ است که تنوع در حرکت ایجاد می‌کند، bF بهترین میزان کیفیت غذا در تکرار فعلی است و wF بدترین میزان کیفیت غذا در تکرار فعلی است.

در معادله (۵) در شرط هدف این است که وقتی کیفیت منبع غذایی $(S(i))$ در نیمه بالایی جمعیت قرار دارد، به کپک وزن بیشتری تعلق بگیرد. این نشان می‌دهد که منابع غذایی با کیفیت بالاتر جزو بهترین گزینه‌ها برای جستجو هستند و کپک بیشتر باید به سمت این منابع جذب شود. اگر کیفیت $S(i)$ خوبی داشته باشد، وزن کپک افزایش می‌یابد و امکان تمرکز بیشتر روی این منابع فراهم می‌شود. r یک عدد تصادفی در بازه $[0,1]$ تنوع الگوریتم را افزایش می‌دهد و به‌طور تصادفی بر وزن و کیفیت منابع غذایی تأثیر می‌گذارد. این مکانیسم تضمین می‌کند که تغییرات کیفیت منابع با اختلاف کم بین bF و $S(i)$ تأثیر قابل‌توجهی نداشته باشد، اما نسبت به منابع کم کیفیت، حساسیت بیشتری نشان دهد. به‌طورکلی، این شرط به کپک کمک می‌کند بهترین منابع غذایی را شناسایی کرده و تمرکز بیشتری روی آن‌ها داشته باشد، درحالی‌که از منابع کم کیفیت فاصله می‌گیرد و بهره‌وری مؤثرتری از منابع با کیفیت بالاتر انجام می‌دهد.

در غیر این‌صورت برای موقعیت‌هایی طراحی‌شده است که کیفیت منابع غذایی در نیمه پایین جمعیت قرار دارد. استفاده از ۱- در این شرط نشان می‌دهد که وقتی منابع غذایی کیفیت کمی دارند، وزن کپک کاهش می‌یابد و تمرکز کمتری روی این منابع کم کیفیت می‌کند و توانایی جستجوی خود را برای تمرکز روی منابع با

کیفیت حفظ می‌کند. به‌طور خلاصه این شرط به کپک کمک می‌کند منابع غذایی با کیفیت پایین را شناسایی کند.

شکل (۳) اثرات و پارامترهای معادله (۱) را نشان داده و تغییر موقعیت فرد جستجوگر را با توجه به معادله (۱) در فضای سه‌بعدی به تصویر می‌کشد.



شکل (۳): مکان‌های ممکن در فضای دوبعدی و سه‌بعدی

پیچیدن غذا

این بخش انقباض ساختار وریدی کپک را هنگام جستجوی غذا شبیه‌سازی می‌کند. هرچه غلظت غذایی که ورید با آن تماس پیداکرده، بیشتر باشد؛ موج تولیدشده توسط نوسان‌ساز زیستی، قوی‌تر است و جریان سیتوپلاسم افزایش می‌یابد. بازخورد مثبت و منفی بین عرض ورید کپک و غلظت غذایی به‌صورت ریاضی در معادله زیر بیان می‌شود:

$$\overrightarrow{X(t)} = \begin{cases} rand.(UB - LB) + LB, rand < p \\ \overrightarrow{X_b(t)} + \overrightarrow{vb}.\left(\vec{W}.\overrightarrow{X_A(t)} - \overrightarrow{X_B(t)}\right), & r < p \\ \overrightarrow{vc}.\overrightarrow{X(t)}, & r \geq p \end{cases} \quad (۷)$$

که در این معادله $\overrightarrow{\boldsymbol{X(t)}}$ بیانگر موقعیت جدید کپک بعد از به‌روزرسانی است، $\boldsymbol{LB}$ و $\boldsymbol{UB}$ نشان‌دهنده حداقل و حداکثر دامنه جستجوی کپک هستند، $\boldsymbol{rand}$ یک مقدار تصادفی در بازه $[\mathbf{0,1}]$ برای تنوع در جستجوی کپک است. $\boldsymbol{z}$ مقدار آستانه‌ای است که مشخص می‌کند آیا کپک از جستجوی تصادفی استفاده کند یا خیر. تعادل بین کاوش و بهره‌وری زمانی ایجاد می‌شود که $\boldsymbol{z = 0.03}$ باشد. پارامتر $\boldsymbol{r}$ عدم قطعیت حالت وریدی شبکه را شبیه‌سازی می‌کند.

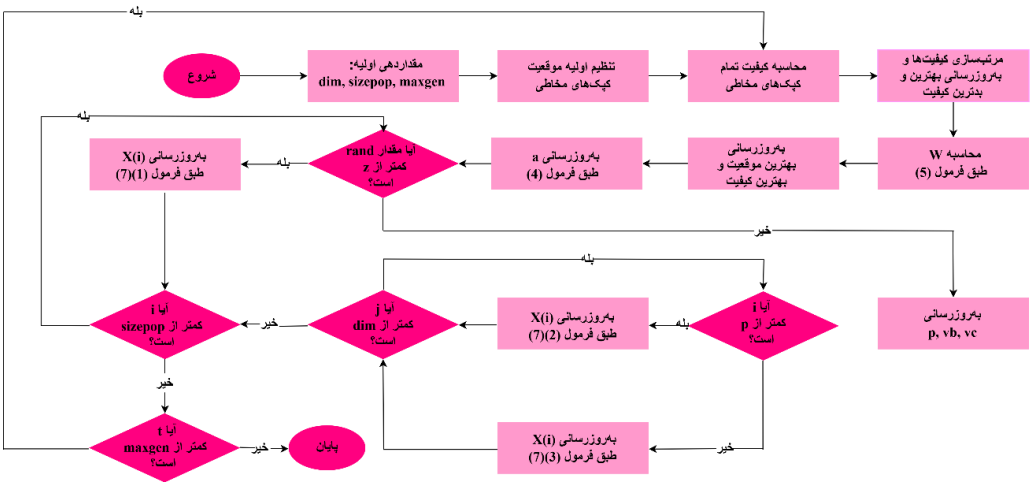
وقتی غلظت غذا بالا باشد، وزن آن منطقه افزایش می‌یابد و جذب آن می‌شود، اما در غلظت‌های پایین‌تر وزن کاهش می‌یابد و قابلیت کاوش مناطق دیگر بیشتر می‌شود. **شرط اول** که $\boldsymbol{rand < z}$ موقعیت ذره را به‌صورت تصادفی در محدوده $[\boldsymbol{LB, UB}]$باز تنظیم می‌کند تا نقاط جدید کشف و تنوع جمعیت حفظ شود و از گیرافتادن در بهینه‌های محلی جلوگیری شود. **شرط دوم** که $\boldsymbol{r < p}$باعث می‌شود، ذره به سمت بهترین موقعیت شناخته شده حرکت کند و با تأثیر تصادفی از سایر ذرات به‌روزرسانی می‌شود. **شرط سوم** که $\boldsymbol{r \geq p}$ اجازه می‌دهد کپک به‌جای حرکت به سمت بوی غذا، با سرعت کمتری به موقعیت فعلی‌اش نزدیک شود.

نوسان

پارامتر $\overrightarrow{\boldsymbol{W}}$ نوسان فرکانس کپک را در نزدیکی غلظت‌های مختلف شبیه‌سازی می‌کند، به‌طوری‌که در نزدیکی منابع غذایی با کیفیت بالا سریع‌تر و در غلظت‌های پایین‌تر کندتر حرکت می‌کند.$\overrightarrow{\boldsymbol{vb}}$ پارامتری تصادفی است که بین محدوده نوسان $[-\boldsymbol{a}, \boldsymbol{a}]$ می‌کند و با افزایش تعداد تکرارها به صفر نزدیک می‌شود، که تعیین می‌کند کپک تصمیم بگیرد به منبع غذایی نزدیک شود یا به کاوش ادامه دهد. $\overrightarrow{\boldsymbol{vc}}$پارامتری است که بین محدوده **[-1,1]** نوسان می‌کند و درنهایت به صفر میل می‌کند. تا کپک حتی پس از یافتن غذا همچنان مناطق جدید را جستجو کند، زیرا ممکن است منابع غذایی با کیفیت بهتری وجود داشته باشد.

حتی اگر منبع غذایی مناسب پیداکرده باشد، مقداری از ماده آلی را جدا می‌کند تا مناطق دیگر را جستجو کند به این دلیل که ممکن است منبع غذایی با کیفیت بهتری درجایی دیگر وجود داشته باشد و نمی‌خواهد تمام انرژی را صرف یک منبع کند. جستجوی غذا همیشه برای کپک راحت نیست، با چالش‌هایی مانند نور یا محیط خشک روبه‌رو است، اما این موانع به فرار از بهینه‌های محلی و یافتن منابع بهتر کمک می‌کنند. ولی پارامتر $\boldsymbol{z}$ که از پیش تعیین می‌شود، تعداد نمونه‌برداری الگوریتم را کنترل می‌کند. همچنین این مقدار هر چه بزرگ‌تر باشد، الگوریتم تمایل بیشتری به کاوش دارد. علاوه بر این کاوش تصادفی برای افزایش تنوع جمعیت امری ضروری است و می‌تواند به جمعیت کمک کند که از دام بهینه‌های محلی فرار کند. پژوهش با آزمایش‌های متعدد نشان می‌دهد که

مقدار $z = 0.03$ به جمعیت این امکان را می‌دهد که به تعادل بین کاوش و بهره‌وری برسد. فرایند بصری دقیق الگوریتم در شکل (٤) نشان داده‌شده است.



شکل (٤): فلوچارت الگوریتم SMA

٤- الگوریتم PSMA

در بخش ۳ الگوریتم SMA معرفی شد. همان‌طور که گفته شد، الگوریتم SMA یکی از الگوریتم‌های بهینه‌سازی است که براساس رفتار هوشمند کپک‌های مخاطی دریافتن منابع غذایی طراحی‌شده است. این الگوریتم به دلیل ساختار ساده و کارایی بالا در بسیاری از مسائل بهینه‌سازی کاربردهای گسترده‌ای دارد. بااین‌حال، همانند سایر الگوریتم‌های فراابتکاری با چالش‌هایی مواجه است که نیاز به بررسی دقیق و ارائه راهکارهای بهبوددهنده دارد. در ادامه به مشکلات این الگوریتم، دلایل آن‌ها و رویکرد پیشنهادی برای بهبود عملکرد الگوریتم پرداخته می‌شود. یکی از مشکلات اصلی الگوریتم SMA کاهش تنوع جمعیت در مسائل چندحالته و پیچیده است که می‌تواند باعث کاهش توانایی الگوریتم دریافتن راه‌حل‌های بهینه شود. در ابتدا ذرات، به دلیل تنوع موقعیت‌های اولیه و کاوش در فضای جستجو عملکرد مناسبی دارند. اما با پیشرفت الگوریتم و نزدیک شدن به جواب بهینه محلی، ذرات به سمت بهترین موقعیت شناخته شده متمایل می‌شوند و تمرکز بیش‌ازحد بر این موقعیت باعث کاهش تنوع جمعیت می‌گردد. این مسئله منجر به گرفتار شدن الگوریتم در بهینه‌های محلی می‌شود، چراکه اگر فضای جستجو چندین بهینه محلی داشته باشد، والگوریتم خیلی زود به یکی از بهینه‌ها نزدیک شود و همه عوامل به سمت آن جذب شوند، امکان کشف کردن بقیه جواب‌های بهتر از بین می‌رود. این مشکل به دلیل نداشتن دافعه مناسب در الگوریتم رخ می‌دهد [۲۹]. یکی دیگر از چالش‌های این الگوریتم، عدم توجه به موقعیت‌های بد یا نامطلوب است. در الگوریتم SMA تنها بهترین موقعیت به‌عنوان مرجع برای بهبود سایر ذرات در نظر گرفته می‌شود و موقعیت‌های نامطلوب نقشی در بهبود الگوریتم ندارند. درحالی‌که در طبیعت، موجودات زنده نه‌تنها به سمت منابع مطلوب جذب می‌شوند، بلکه از موقعیت‌های خطرناک یا نامطلوب دوری می‌کنند. به‌عنوان‌مثال، حیوانات در هنگام حرکت در محیط، علاوه بر جستجوی غذا، از شکارچیان یا مناطق خطرناک فاصله می‌گیرند. این رفتار طبیعی می‌تواند به‌عنوان یک الگو برای طراحی الگوریتم‌های بهینه‌سازی استفاده شود [٦]. این مسئله می‌تواند تنوع در حرکات عوامل جستجو را کاهش دهد. یکی از راهکارهای مؤثر برای رفع این مشکلات، و افزایش تنوع جمعیت و جلوگیری از گرفتار شدن در بهینه‌های محلی اضافه کردن نیروی دافعه به

الگوریتم است. این رویکرد براساس فاصله هر ذره از بدترین موقعیت عمل می‌کند. به‌عبارت‌دیگر، علاوه بر محاسبه بهترین موقعیت، بدترین موقعیت در هر تکرار الگوریتم شناسایی‌شده و نیرویی برای دور کردن سایر ذرات از این موقعیت اعمال می‌شود. این نیروی دافعه به حفظ تنوع جمعیت کمک می‌کند و احتمال گرفتار شدن الگوریتم در بهینه‌های محلی را کاهش می‌دهد. همچنین به الگوریتم اجازه می‌دهد که تعادل بهتری بین کاوش و بهره‌وری ایجاد کند. در عمل، این مکانیزم باعث می‌شود که برخی از ذرات که در موقعیت‌های ضعیف‌تری قرار دارند، نیرویی دریافت کنند که آن‌ها را به سمت مناطق جدیدتری از فضای جستجو هدایت کند. این امر نه‌تنها احتمال کشف بهینه‌های بهتر را افزایش می‌دهد، بلکه از تمرکز بیش‌ازحد جمعیت بر یک منطقه خاص جلوگیری می‌کند. باوجوداینکه بهبودهای زیادی برای الگوریتم SMA وجود دارد، فشار انتخاب در این الگوریتم و در فرآیند جستجو به‌خوبی تنظیم‌نشده است [۷]. در طبیعت به‌منظور کاهش تأثیر منفی ذراتی که در موقعیت‌های بدتر قرار دارند، عملگر تنبیه طراحی‌شده است. عملگر تنبیه به این صورت عمل می‌کند که در هر تکرار الگوریتم، علاوه بر محاسبه بهترین ذره، بدترین ذره نیز شناسایی می‌شود. بدترین ذره نشان‌دهنده موقعیتی است که کمترین کیفیت را در بین تمام ذرات جمعیت دارد. برای هر ذره در جمعیت، فاصله آن با بدترین ذره محاسبه می‌شود. سپس یک نیروی دافعه براساس این فاصله تعریف می‌شود. این نیروی دافعه، ذرات را از موقعیت بدترین ذره دور می‌کند و مانع از تجمع آن‌ها در نواحی ضعیف فضای جستجو می‌شود. به‌عبارت‌دیگر:

جذب به سمت بهترین موقعیت: ذرات به سمت بهترین ذره شناخته شده در جمعیت حرکت می‌کنند. دافعه از بدترین موقعیت: هم‌زمان نیرویی از سمت بدترین ذره اعمال می‌شود که ذرات را از آن دور می‌کند. این دونیرو (جاذبه و دافعه) با یکدیگر ترکیب‌شده و تعادلی ایجاد می‌کنند که باعث می‌شود ذرات هم‌زمان فضای جستجو را کاوش کنند. یکی از مزایای مهم عملگر دافعه، کنترل فشار انتخاب است. در الگوریتم‌های بهینه‌سازی فشار انتخاب به معنی میزان تمایل ذرات به حرکت به سمت بهترین جواب‌ها است. فشار انتخاب بیش‌ازحد، منجر به کاهش تنوع جمعیت و گرفتاری الگوریتم در بهینه‌های محلی می‌شود. عملگر دافعه با اعمال یک نیروی مخالف، این فشار را متعادل می‌کند. وقتی ذرات به سمت بهترین موقعیت جذب می‌شوند، معمولاً تعدادی از آن‌ها در موقعیت‌های ضعیف‌تری قرار دارند. این ذرات که عملکرد مناسبی ندارند، ممکن است به‌مرورزمان به نواحی ضعیف‌تر فضای جستجو کشیده شوند. عملگر دافعه باایجاد نیرویی که این ذرات را از نواحی ضعیف دور می‌کند، به آن‌ها کمک می‌کند که دوباره در فضای جستجو پراکنده شوند و به کاوش بپردازند. این مکانیزم به حفظ تنوع جمعیت کمک کرده و از همگرایی زودهنگام جلوگیری می‌کند. نیروی دافعه معمولاً براساس فاصله بین هر ذره و بدترین ذره تعریف می‌شود. این عملگر زمانی که ذرات بیش‌ازحد به یک نقطه متمرکزشده و در یک بهینه محلی گرفتار می‌شوند، فعال می‌شود. با اعمال نیروی دافعه، فاصله بین ذرات افزایش‌یافته و از همگرایی زودرس جلوگیری می‌شود. این فرایند به حفظ تنوع جمعیت کمک کرده و امکان کاوش مؤثرتر در فضای جستجو را فراهم می‌کند، درنتیجه احتمال یافتن پاسخ‌های بهتر بیشتر می‌شود. این نیرو می‌تواند به‌صورت یک تابع کاهش‌یافته از فاصله باشد. فرمول این عملگر به‌صورت زیر است:

$$\vec{\mathbf{X}} = \overrightarrow{\mathbf{vc}}.\overrightarrow{\mathbf{X(t)}} - \mathbf{C}(\overrightarrow{\mathbf{X(t)}} - \mathbf{wp(j)}) \qquad (۸)$$

این معادله تعمیم‌یافته معادله (۷) است. در الگوریتم SMA، موقعیت جدید ذره بر اساس بهترین موقعیت‌ها و بردارهای تفاوت به‌روزرسانی می‌شود. برای بهبود جستجوی سراسری و جلوگیری از گرفتاری در بهینه‌های محلی، یک مؤلفه دافعه بر پایه فاصله از بدترین ذره به معادله اضافه‌شده است. در این معادله، $\overrightarrow{\boldsymbol{X(t)}}$ موقعیت فعلی ذره، $\boldsymbol{wp(j)}$ مختصات بدترین موقعیت شناخته شده و $\boldsymbol{C}$ یک ثابت مثبت است که به‌عنوان ضریب تنبیه عمل می‌کند و شدت نیروی دافعه را تعیین می‌کند. همچنین $\overrightarrow{\boldsymbol{vc}}$ پارامتری است که بین محدوده **[-1,1]** نوسان می‌کند و درنهایت به صفر میل می‌کند.

بخش اول معادله، $\overrightarrow{\boldsymbol{vc}}.\overrightarrow{\boldsymbol{X(t)}}$ باعث حفظ ذره کنونی می‌شود در حالیکه بخش دوم، $\boldsymbol{C}(\overrightarrow{\boldsymbol{X(t)}} - \boldsymbol{wp(j)})$ به‌عنوان نیروی دافعه عمل می‌کند که جهت آن برخلاف فاصله میان ذره و بدترین موقعیت است. هنگامی‌که یک ذره به موقعیت بدترین ذره نزدیک‌تر می‌شود، فاصله آن با کاهش‌یافته و درنتیجه نیروی دافعه تشدید

می‌شود. که این فرآیند سبب پراکندگی بیشتر ذرات در فضای جستجو شده و به خروج ذرات از نواحی مینیمم محلی کمک می‌کند. این رویکرد نه‌تنها از تمرکز بیش‌ازحد ذرات در نواحی ضعیف جلوگیری می‌کند، بلکه تعادل بین مراحل کاوش و بهره‌وری را نیز بهبود می‌بخشد. شبه کد الگوریتم PSMA در الگوریتم (۱) نشان داده‌شده است.

الگوریتم(۱): شبه کد الگوریتم PSMA

ورودی‌ها: Max_t, size_pop, dim

خروجی‌ها: bestFitness

۱- تولید جمعیت اولیه به‌صورت تصادفی

۲- مقداردهی موقعیت‌های اولیه کپک به‌صورت تصادفی

۳- تازمانی‌که معیار توقف برقرار نشده انجام بده:

٤- محاسبه تابع هدف تمام کپک‌های مخاطی

٥- به‌روزرسانی بهترین مقدار تابع هدف و موقعیت متناظر با آن

٦- اگر $\boldsymbol{i < \frac{N}{2}}$ آن‌گاه:

۷- محاسبه $\boldsymbol{W}$ طبق معادله (٥,١)

۸- در غیر این‌صورت

۹- محاسبه $\boldsymbol{W}$ طبق معادله (٥,٢)

۱۰- پایان اگر

۱۱- اگر $\boldsymbol{rand < z}$ آن‌گاه:

۱۲- به‌روزرسانی $\boldsymbol{X(i)}$ طبق معادله (۷)

۱۳- در غیراین‌صورت

۱٤- به‌روزرسانی $\boldsymbol{p, vb, vc}$

۱٥- پایان اگر

۱٦- اگر $\boldsymbol{r < p}$ آن‌گاه:

۱۷- به‌روزرسانی طبق $\boldsymbol{X(i)}$ معادله (۱)

۱۸- در غیر این‌صورت

۱۹- به‌روزرسانی طبق $\boldsymbol{X(i)}$ معادله (۸)

۲۰- پایان اگر

۲۱- $\boldsymbol{t = t + 1}$

۲۲- پایان حلقه

۲۳- چاپ بهترین جواب

۲٤- پایان

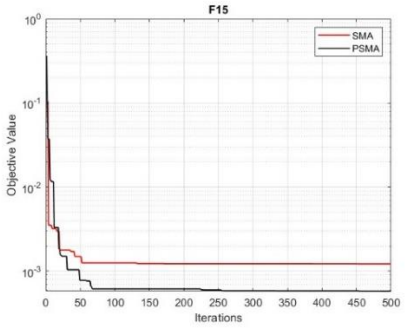
۵- نتایج تجربی و تجزیه‌وتحلیل

در این بخش به‌منظور ارزیابی کارایی الگوریتم پیشنهادی، الگوریتم PSMA با الگوریتم‌های EO [۳۴]، GWO [۳۵]، GSA [۱۴]، HHO [۳۶]، SMA [۶]، QCMSMA [۳۷] و CSSMA [۳۸] روی مجموعه‌ای جامع از ۲۳ تابع معیار CEC2017 مقایسه شده‌اند. جدول (۱) جزئیات مربوط به پارامترهای این الگوریتم‌ها را ارائه می‌دهد که پارامترها و متغیرهای این الگوریتم‌ها با توجه به مقاله مرجع آن‌ها تنظیم‌شده‌اند. برای الگوریتم پیشنهادی مقدار پارامتر عملگر تنبیه بر اساس آزمایش‌ها و نتایج‌ها متعدد-C=1e 195، تنظیم‌شده است. توابع هدف مورداستفاده شامل مجموعه‌ای از توابع تک‌حالته و چندحالته هستند که به‌عنوان معیار استاندارد برای ارزیابی عملکرد الگوریتم‌های بهینه‌سازی به کار می‌روند. توابع تک‌حالته برای آزمایش توانایی الگوریتم‌ها در همگرایی سریع و دقیق به سمت بهینه مطلق طراحی‌شده‌اند. ساده‌بودن ساختار این توابع آن‌ها را به گزینه‌ای مناسب برای بررسی عملکرد الگوریتم‌ها در شرایط کم‌چالش تبدیل می‌کند. در مقابل، توابع چندمدی شامل چندین نقطه بهینه محلی هستند. این توابع فضای جستجوی پیچیده‌ای دارند و از الگوریتم‌ها انتظار می‌رود که بتوانند از نقاط بهینه محلی عبور کرده و به نقطه بهینه مطلق دست یابند. استفاده از این توابع برای سنجش توانایی الگوریتم‌ها در کاوش فضای جستجو، اجتناب از گرفتاری در بهینه محلی و حل مسائل پیچیده مناسب است. نتایج شبیه‌سازی با ۵۰۰ تکرار و ۳۰ عامل جستجو به‌دست‌آمده است [۳۹]. نتایج مربوط به میانگین ۵۰ اجرای مستقل این الگوریتم‌ها روی توابع آزمایشی در جدول شماره (۲) آورده شده است. نتایج موجود در جدول (۲) نشان می‌دهد که الگوریتم PSMA در این آزمایش‌ها عملکرد متفاوتی روی توابع تست داشته است. این توابع به سه دسته تقسیم می‌شوند: توابع F1 تا F7 تک‌حالته هستند وتنها یک نقطه بهینه دارند، توابع F8 تا F13 چندحالته هستند و دارای چندین نقطه بهینه‌محلی هستند و توابع F14 تا F23 جزو توابع پیچیده هستند [۳۹]. در ارزیابی انجام‌شده، PSMA در توابع تک‌حالته عملکرد دقیقی داشته و موفق به یافتن مقدار بهینه با خطای کم یا مقدار صفر شده است، که نشان‌دهنده توانایی بالای الگوریتم در بهینه‌سازی توابع ساده و همچنین سرعت همگرایی بالای آن است. در توابع چندحالته، نتایج نشان می‌دهد که PSMA در توابعی مانند F9، F8 و F11 نسبت به الگوریتم SMA، بهبودیافته و از همگرایی زودرس جلوگیری کند. در دسته توابع پیچیده، عملکرد PSMA متغیر بوده است. این الگوریتم در برخی توابع مانند F13 و F15 عملکرد خوبی داشته است و به نتایج قابل قبولی دست‌یافته است، اما در سایر توابع به‌ویژه توابعی که بسیار پیچیده هستند، کارایی مشابه با سایر الگوریتم‌ها دارد. به‌طورکلی می‌توان نتیجه گرفت که PSMA در توابع ساده و برخی توابع چندحالته عملکرد مطلوبی داشته و نسبت به SMA بهبودیافته است و از همگرایی زودرس جلوگیری می‌کند [۳۹]. همچنین منحنی‌های همگرایی F15 و F13 و F7 در شکل‌های (۵) و (۶) و (۷) را می‌توان به‌صورت بصری مشاهده کرد که PSMA سریع‌ترین روند همگرایی را در مقایسه با SMA داشته است. همچنین مشاهده می‌شود که در شکل (۵) الگوریتم SMA که با رنگ قرمز نشان داده‌شده است، در تکرار ۴۵ در دام بهینه‌محلی گرفتارشده است که نشان‌دهنده بهره‌وری بیشتر الگوریتم است ولی نمودار PSMA که با رنگ مشکی نشان داده‌شده است همچنان به جستجو ادامه داده است که نشان دهنده کاوش بیشتر الگوریتم است. همان‌طور که در شکل‌های (۵)، (۶) مشاهده می‌شود، زمانی‌که الگوریتم در بهینه محلی به دام می‌افتد، نمودار همگرایی آن‌ها به یک خط افقی تبدیل می‌شود تحت این شرایط الگوریتم PSMAفرآیند کاوش را اعمال می‌کند. در ابتدای کار انتظار می‌رود قابلیت کاوش زیاد باشد و فشار انتخاب کم که تنوع جمعیت زیاد باشد و الگوریتم تا حد امکان مناطق را جستجو کند و با پیشرفت الگوریتم انتظار می‌رود در پایان کار قابلیت بهره‌وری افزایش یابد و فشار انتخاب افزایش یابد تا الگوریتم روی نواحی امیدوارکننده تمرکز کند و کیفیت جواب را بهبود دهد. این الگوریتم می‌تواند مناطق جدیدتر را در فضای جستجو با تغییراتی کشف کند که از به دام افتادن آن در بهینه محلی جلوگیری می‌کند. نتایج نشان می‌دهد که اعمال نفوذ در الگوریتم SMA به‌طور قابل‌توجهی روند کاوش را افزایش می‌دهد. این بهبود به الگوریتم PSMA اجازه می‌دهد تا از دام‌های بهینه محلی فرار کند و جستجوی کارآمدتری

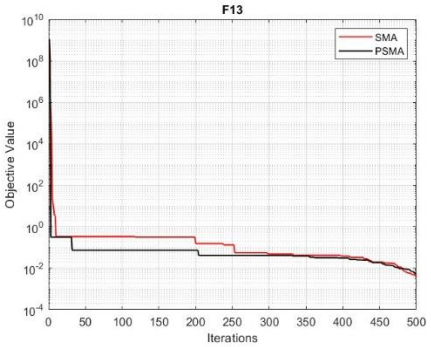
را در فضای جستجو انجام دهد. در الگوریتم پیشنهادی، اضافه شدن عملگر تنبیه، با وجود محاسبات کم و افزایش اندک زمان اجرا، کارایی الگوریتم دریافتن بهینه محلی را بسیار بهبود داده است. این افزایش زمان به دلیل اضافه شدن محاسبات مربوط به عملگر تنبیه است که منجر به کنترل فشار انتخاب‌شده است. بااین‌حال، در مقایسه با برخی الگوریتم‌ها مانند EO،GWO، CSSMA، QCMSMA و GSA همچنان زمان اجرای کمتری دارد. بنابراین PSMA تعادلی میان دقت و سرعت برقرار کرده است و در مقایسه با روش‌های کندتر، گزینه‌ای مناسب برای بهینه‌سازی مسائل پیچیده محسوب می‌شود. میانگین زمان اجرای الگوریتم‌های مقایسه‌ای در جدول (۳) آورده شده است. به‌منظور ارزیابی دقیق و کارایی الگوریتم پیشنهادی، نتایج جدول (۲) با آزمون فریدمن تحلیل‌شده است. این روش یک آزمون ناپارامتری است که الگوریتم‌ها را برای هر مجموعه داده متفاوت رتبه‌بندی می‌کند. فرضیه صفر در این آزمون بیان می‌کند که بین کارایی الگوریتم‌های مختلف تفاوت معناداری وجود ندارد. اگر فرضیه صفر رد شود، به این معنی است که اختلاف کارایی الگوریتم‌ها معنادار است [٤٠]. نتایج آزمون فریدمن در جدول (٤) ارائه‌شده است. همان‌طور که مشاهده می‌شود، الگوریتم PSMA رتبه اول را کسب کرده است و فرضیه صفر رد شده است که این امر اعتبار برتری الگوریتم PSMA را تقویت می‌کند. همچنین به‌منظور ارزیابی دقیق و کارایی الگوریتم پیشنهادی، نتایج جدول (۱) با آزمون ویلکاکسون تحلیل‌شده است. این روش یک آزمون ناپارامتری است که برای مقایسه دو مجموعه داده متفاوت به‌کار می‌رود. نتایج آزمون ویلکاکسون در جدول (۵) ارائه‌شده است. همان‌طور که مشاهده می‌شود، الگوریتم PSMA در مقایسه با سایر الگوریتم‌ها کارایی بهتری داشته و توانسته آن‌ها را مغلوب کند.

جدول (۱): مقادیر پارامترهای الگوریتم‌های مقایسه‌ای

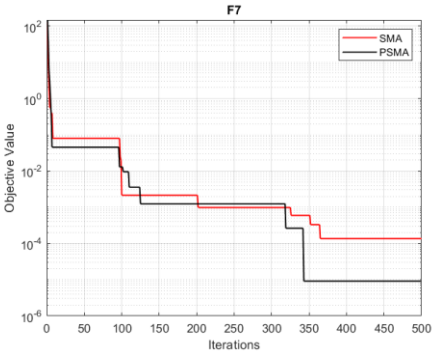
Algorithms	Parameters setting
PSMA	$z = 0.03, c = 1e-195$
SMA	$z = 0.03$
EO	$a_1 = 2, a_2 = 1$
GSA	$a = 20, G0 = 100$
GWO	$a = [2,0]$
HHO	$\beta = 1.5$
QCMSMA	$C = 10$
CSSMA	$z = 0.03, s = 0.01$ $, \beta = 1.5$



شکل (۵): مقایسه کارایی الگوریتم پیشنهادی با الگوریتم SMA روی تابع F15



شکل (٦): مقایسه کارایی الگوریتم پیشنهادی با الگوریتم SMA روی تابع F13



شکل (۷): مقایسه کارایی الگوریتم پیشنهادی با الگوریتم SMA روی تابع F7

جدول (۲): نتایج مقایسه الگوریتم‌ها بر روی توابع تست

Function	PSMA	SMA	GSA	GWO	HHO	EO	QCMSMA	CSSMA
F1	0.00	0.00	1.55e-17	2.59e-28	3.64e-106	3.28e-41	0.00	0.00
F2	2.99e-192	3.39e-186	2.29e-08	1.09e-16	1.16e-48	9.20e-23	0.00	2.00
F3	0.00	0.00	324.21	1.19e-08	1.72e-88	4.47e-09	0.00	0.00
F4	1e-193	1.55e-186	2.46e-09	5.18e-07	4.49e-50	5.16e-10	2.46e-130	6.36
F5	0.038383	28.0606	26.4083	27.1446	0.046607	25.30543	6.27	3.63e-04
F6	3.77e-03	38e-02.1	0.00	2.49e-01	7.89e-06	7.28e-06	8.69e-01	1.35e-04
F7	3.98e-05	1.64e-04	2.29e-02	1.85e-03	2.30e-05	1.34e-03	8.14e05	6.24e-05
F8	-12568.95	-12568.95	-2.40e+03	-5759.16	-12569.48	-9098.07	-1.26e+04	-1.26e+04
F9	0.00	0.00	23.87	5.68e-14	0.00	1.89e-15	0.00	0.00
F10	8.88e-16	8.88e-16	3.07e-09	1.28e-13	8.88e-16	8.34e-15	4.44e-16	4.44e-16
F11	0.00	0.00	2.61	0.00	0.00	1.8e-03	0.00	0.00
F12	7.65e-04	3.61e-03	1.86e-19	5.88e-02	1.69e-05	7.46e-07	1.95e-07	3.31e-05
F13	8.96e-04	4.99e-03	3.08e-18	3.95e-01	1.38e-05	1.20e-02	5.00e+04	5.56e-05
F14	9.98e-01	9.98e-01	2.00	9.98e-01	9.98e-01	1.06	9.98e01	9.98e-01
F15	4.08e-04	1.22e-03	3.50e-04	4.03e-04	3.83e-04	3.07e-03	3.42e-04	5.16e-04
F16	-1.03	-1.03	-1.03	-1.03	-1.03	-1.03	-1.03	-1.03
F17	3.97e-01	3.97e-01	3.9e-01	3.97e-01	3.97e-01	3.97e-01	3.98e-01	3.98e-01
F18	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00
F19	-3.86	-3.86	-3.86	-3.85	-3.86	-3.86	-3.86	-3.86
F20	-3.20	-3.19	-3.32	-3.08	-3.04	-3.27	-3.31	-3.30
F21	-10.15	-10.15	-2.68	-10.15	-5.05	-9.56	-1.02e+01	-1.02e+01
F22	-10.40	-10.4029	-10.4029	-10.4017	-5.0845	-8.896016	-1.04e+01	-1.04e+01
F23	-10.53	-10.53	-10.53	-10.53	-5.12	-10.26	-1.05e+01	-1.04e+01

جدول (۳): میانگین زمان اجرای الگوریتم‌های مقایسه‌ای

Algorithms	Time
PSMA	0.8316
SMA	0.2312
GSA	3.259
GWO	137.64
HHO	0.1930
EO	128.79
QCMSMA	123.98
CSSMA	98.456

جدول (٤): نتایج آزمون فریدمن برای مقایسه الگوریتم‌ها

Methods	Ranks
PSMA	76.78261
CSSMA	83.47826
QCMSMA	85.10870
SMA	85.58690
HHO	93.43478
EO	96.19565
GWO	101.32609
GSA	118.08696
$\boldsymbol{P\ value(\alpha = 0.05)}$	$H0 = rejected$

جدول (٥): نتایج آزمون ویلکاکسون برای مقایسه الگوریتم‌ها

Algorithms	Statistic	P-Value
PSMA vs QCMSMA	47	7.29e-01
PSMA vs GSA	57	1.26e-01
PSMA vs HHO	47	4.60e-01
PSMA vs SMA	6	2.84e-02
PSMA vs GWO	15	6.13e-03
PSMA vs CSSMA	44	5.93e-01
PSMA vs EO	32	1.12e-02

٦- نتیجه‌گیری

این مقاله به تنظیم فشار انتخاب در الگوریتم SMA به‌عنوان یک چالش بهینه‌سازی، با هدف بهینه‌سازی الگوریتم SMA می‌پردازد. برای دستیابی به این هدف از یک عملگر جدید به‌نام تنبیه برای بهبود الگوریتم SMA استفاده می‌شود. عملگر تنبیه با الهام از طبیعت، برای بهبود عملکرد الگوریتم کپک مخاطی تعریف‌شده است. الگوریتم PSMA عملکرد رضایت‌بخشی را در مقایسه با سایر الگوریتم‌ها در توابع تست CEC2017 نشان می‌دهد.

References

[1] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems, " in *Robots and Biological Systems: Towards a New Bionics*, H. Bolles and C. Giralt, Eds. Berlin, Germany: Springer, 1993, pp. 703–712.

[2] F. S. Gharehchopogh, et al., "Slime mould algorithm: A comprehensive survey of its variants and applications," *Arch. Comput. Methods Eng.*, vol. 30, no. 4, pp. 2683-2723, Jan 2023.

[3] H. Nezamabadi-pour, *Genetic Algorithm: Basic Concepts and Advanced Topics*. Kerman, Iran: Shahid Bahonar University of Kerman, 2010.

[4] T. T. Hills, et al., "Exploration versus exploitation in space, mind, and society," *Trends Cogn. Sci.*, vol. 19, no. 1, pp. 46-54, Jan. 2015.

[5] M. Črepinšek, S.-H. Liu, M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 1-33, Sep. 2013.

[6] S. Li, et al., "Slime mould algorithm: A new method for stochastic optimization," *Future Gener. Comput. Syst.*, vol. 111, pp. 300-323, Nov. 2020.

[7] D. Dhawale, V. K. Kamboj, P. Anand, "An effective solution to numerical and multi-disciplinary design optimization problems using

chaotic slime mold algorithm," *Eng. Comput.*, 2022.
[8] Y. Liu, et al., "Boosting slime mould algorithm for parameter identification of photovoltaic models," *Energy*, vol. 234, article 121164, Jul. 2021.
[9] H.-P. Schwefel, *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, 1977.
[10] D. Quagliarella, et al., *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, John Wiley & Sons, 1997.
[11] S. Desale, et al., "Heuristic and meta-heuristic algorithms and their relevance to the real world: a survey," *Int. J. Comput. Eng. Res. Trends*, vol. 351, no. 5, pp. 2349-7084, 2015.
[12] S. Bastami and M. Doulattshahi, "Compact neural architecture search for image classification using gravitational search algorithm," *J. Theor. Appl. Mach. Intell.*, vol. 2, no. 1, pp. 77–91, 2025.
[13] M. Mohammadi, et al., "Security-aware resource allocation in fog computing using a meta-heuristic algorithm," *Cluster Comput.*, vol. 28, no. 2, article 104, 2025.
[14] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, "GSA: a gravitational search algorithm," *Inf. Sci.*, vol. 179, no. 13, pp. 2232-2248, Jul. 2009.
[15] S. Mirjalili, "SCA: a sine cosine algorithm for solving optimization problems," *Knowl.-Based Syst.*, vol. 96, pp. 120-133, Nov. 2016.
[16] R. V. Rao, V. J. Savsani, D. Vakharia, "Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems," *Inf. Sci.*, vol. 183, no. 1, pp. 1-15, Jan. 2012.
[17] L. B. Booker, D. E. Goldberg, J. H. Holland, "Classifier systems and genetic algorithms," *Artif. Intell.*, vol. 40, no. 1-3, pp. 235-282, 1989.
[18] R. Storn, K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, vol. 11, pp. 341-359, 1997.
[19] J. Kennedy, R. Eberhart, "Particle swarm optimization," in *Proc. ICNN'95 - Int. Conf. Neural Networks*, IEEE, 1995, pp. 1942-1948.
[20] L. Lin, M. Gen, "Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation," *Soft Comput.*, vol. 13, pp. 157-168, 2009.
[21] D. Gürses, et al., "Comparison of the arithmetic optimization algorithm, the slime mold optimization algorithm, the marine predators algorithm, the salp swarm algorithm for real-world engineering applications," *Mater. Test.*, vol. 63, no. 5, pp. 448-452, 2021.
[22] A.-D. Tang, et al., "A modified slime mould algorithm for global optimization," *Comput. Intell. Neurosci.*, vol. 2021, article 2298215, 2021.
[23] Bala Krishna, S. Saxena, V. K. Kamboj, "hSMA-PS: a novel memetic approach for numerical and engineering design challenges," *Eng. Comput.*, vol. 38, no. 4, pp. 3513-3547, 2022.
[24] D. Yousri, et al., "A reliable approach for modeling the photovoltaic system under partial shading conditions using three diode model and hybrid marine predators-slime mould algorithm," *Energy Convers. Manag.*, vol. 243, article 114269, 2021.
[25] M. K. Naik, R. Panda, A. Abraham, "Adaptive opposition slime mould algorithm," *Soft Comput.*, vol. 25, no. 22, pp. 14297-14313, 2021.
[26] E. H. Houssein, et al., "Hybrid slime mould algorithm with adaptive guided differential evolution algorithm for combinatorial and global optimization problems," *Expert Syst. Appl.*, vol. 174, article 114689, 2021.
[27] S. R. Biswal, et al., "Optimal allocation/sizing of DGs/capacitors in reconfigured radial distribution system using quasi-reflected slime mould algorithm," *IEEE Access*, vol. 9, pp. 125658-125677, 2021.
[28] A. Ewees, et al., "Improved Slime Mould Algorithm based on Firefly Algorithm for feature selection: A case study on QSAR model," *Eng. Comput.*, 2021.Slowik, H. Kwasnicka, "Evolutionary algorithms and their applications to engineering problems," *Neural Comput. Appl.*, vol. 32, pp. 12363-12379, 2020.
[29] M. Becker, "On the efficiency of nature-inspired algorithms for generation of fault-tolerant graphs," in *2015 IEEE Int. Conf. Syst., Man, Cybern.*, IEEE, 2015, pp. 1059-1064.
[30] D. Kessler, "Plasmodial structure and motility," in *Cell Biology of Physarum and Didymium*, vol. 1, pp. 145–208, 1982.
[31] V. Šešum-Čavić, E. Kühn, D. Kanev, "Bio-inspired search algorithms for unstructured P2P overlay networks," *Swarm Evol. Comput.*, vol. 29, pp. 73-93, 2016.
[32] T. Latty, M. Beekman, "Speed–accuracy trade-offs during foraging decisions in the acellular slime mould Physarum polycephalum," *Proc. R. Soc. B*, vol. 278, no. 1705, pp. 539-545, 2011.
[33] Faramarzi, et al., "Equilibrium optimizer: A novel optimization algorithm," *Knowl.-Based Syst.*, vol. 191, article 105190, 2020.

[34] S. Mirjalili, S. M. Mirjalili, A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46-61, 2014.

[35] A. Heidari, et al., "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849-872, 2019.

[36] Z. Duan, X. Qian, W. Song, "Multi-Strategy Enhanced Slime Mould Algorithm for Optimization Problems," *IEEE Access*, 2025.

[37] T.-L. Wang, et al., "CSSMA: A novel algorithm of slime mold optimizer for global optimization problems," 2023.

[38] G. Wu, R. Mallipeddi, P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization," *Technical Report*, 2017.

[39] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1-30, Jan. 2006.

Journal of Applied And Basic Machine Intelligence Research (ABMIR)

Applied and Basic Machine Intelligence Research

Vol. 2, No. 2, Autumn & Winter 2024, pp. 145 - 161

Punishment: A new operator to control selection pressure and improve the efficiency of the slime mold algorithm

Arezoo Rahimi[1], Mohammad Farshi[2], Sepehr Ebrahimi Mood[3*]

[1]PhD Student, Department of Computer Science, Yazd University, Yazd, Iran

[2]Associate Professor, Department of Computer Science, Yazd University, Yazd, Iran

[3]Assistant Professor, Department of Computer Science, Yazd University, Yazd, Iran

Article Information

Original Research Paper

Received:
2025 February 23

Accepted:
2025 May 6

Keywords:
Slime mould algorithm, Selection pressure, Punishment operator, Optimization

Corresponding Author*:
s.ebrahimi@yazd.ac.ir

Abstract

In this paper, the Slime Mould Algorithm (SMA), inspired by the biological behavior of slime moulds, is examined and evaluated as a powerful metaheuristic method for solving complex optimization problems. One of the main challenges of this algorithm is premature convergence caused by the lack of control over selection pressure. To address this issue, a penalizing operator is proposed to regulate selection pressure and maintain population diversity. Unlike other methods that do not manage selection pressure during the computational process of the algorithm and make decisions regardless of the algorithm's current state, the proposed operator dynamically adjusts its behavior based on the algorithm's current conditions. When particles experience premature convergence, a repulsive force is applied to help them escape local optima and achieve better exploration of the search space. Extensive experiments were conducted on 23 benchmark functions from the CEC2017 suite, including various types such as unimodal, multimodal, hybrid, and complex functions, in order to evaluate the performance of the proposed operator under diverse and challenging conditions. The results show that the improved SMA achieved a 35.5% performance enhancement compared to the original version on standard test functions. The simulation results and experimental findings in this study demonstrate the effectiveness of the proposed operator in controlling selection pressure, leading to improved performance of the enhanced Slime Mould Algorithm in solving complex optimization problems. This makes it a practical and efficient solution for a wide range of applications in science and engineering.

: 10.22034/ABMIR.2025.22831.1104

E-ISSN: 2821-2037