دوره پانزدهم، شماره پاییز و زمستان ۱۴۰۱

مجله فناوری اطلاعات در طراحی مهندسی

Information Technology in Engineering Design

http://ited.sinaweb.net

بهبود فرایند جستجوی الگوریتم بهینه‌سازی وال با استفاده از نظریهٔ تکامل داروین و سیستم استنتاج فازی

محسن پرهیزگار [۱] سیدمحمدحسین معطر* [۲]

(۱) گروه مهندسی کامپیوتر، واحد فردوس، دانشگاه آزاد اسلامی، فردوس، ایران

(۲) گروه مهندسی کامپیوتر، واحد مشهد، دانشگاه آزاد اسلامی، مشهد، ایران*

(تاریخ دریافت :۱۴۰۱/۰۷/۲۳ تاریخ پذیرش: ۱۴۰۱/۱۲/۱۸)

چکیده

الگوریتم بهینه‌سازی وال (WOA) را به دلیل توانایی اکتشاف و بهره‌برداری، می‌توان یک بهینه‌ساز سراسری در نظر گرفت. با این حال، همچنان این الگوریتم از نظر دقت و سرعت همگرایی ضعیف است و در برخی مسائل جوابهایی در اطراف بهینه سراسری تولید می کند. در این مقاله، با استفاده از اصل تکاملی داروین و ترکیب مکانیزم تولید مثل با الگوریتم WOA، اکتشاف و بهره‌برداری الگوریتم را بهبود داده‌ایم. هدف از این ترکیب بهبود دقت همگرایی الگوریتم است. در این روش ترکیبی از عملگر ادغام و جهش به ترتیب به‌منظور بهبود بهره‌برداری و اکتشاف استفاده شده است. عملگر جهش با فراهم کردن شرایط تصادفی بیشتر، باعث بهبود اکتشاف الگوریتم می‌شود. عملگر ادغام با ترکیب بهترین موقعیت با دیگر عناصر جستجو باعث بهبود بهره‌برداری الگوریتم می‌شود. همچنین از یک سیستم استنتاج فازی (FIS) به‌منظور ایجاد تعادل بین فاز اکتشاف و بهره‌برداری استفاده کرده‌ایم. استفاده از FIS در روش پیشنهادی باعث می‌شود فازهای اکتشاف و بهره‌برداری در هر بار اجرا متناسب با نیاز تنظیم شوند. مقایسه عملکرد روش پیشنهادی بر روی ۲۳ تابع محک، نشان داد روش پیشنهادی عملکرد بهتری نسبت به الگوریتم‌های مورد مقایسه دارد. همچنین آزمایشات نشان می‌دهد که با استفاده از رویکردهای پیشنهادی الگوریتم زودتر همگرا می‌شود.

کلمات کلیدی: نظریه تکامل داروین، الگوریتم بهینه‌سازی وال، الگوریتم ژنتیک گل‌میخ، سیستم استنتاج فازی

* عهده‌دار مکاتبات:

محمدحسین معطر

نشانی: گروه مهندسی کامپیوتر، واحد مشهد، دانشگاه آزاد اسلامی، مشهد، ایران

پست الکترونیکی: *moattar@mshdiau.ac.ir*

۱- مقدمه

با توجه به ظهور مسائل بهینه‌سازی پیچیده در سال‌های اخیر و ناتوانی روش‌های سنتی بهینه‌سازی در حل این مسائل، اخیرا شاهد ظهور تکنیک‌های بهینه‌سازی متفاوتی هستیم که از جمله‌ی آنها می‌توان به الگوریتم‌های فرااکتشافی اشاره کرد. الگوریتم‌های فرااکتشافی به منظور حل مسائل پیچیده ارائه شده‌اند. روش‌های بهینه‌سازی الهام گرفته از طبیعت، از جمله الگوریتم‌های بهینه‌سازی فرااکتشافی[۱] هستند که با الهام از پدیده‌های طبیعی برای حل مسائل بهینه‌سازی ارائه شده‌اند. از جمله محبوب‌ترین این تکنیک‌ها می‌توان به الگوریتم ژنتیک[۲] (GA)[۱]، الگوریتم بهینه‌سازی ازدحام ذرات[۳] (PSO) [۲]، الگوریتم کلونی مورچگان[۴] (ACO)[۳]، الگوریتم کلونی زنبور عسل مصنوعی[۵] (ABC)[۴]، الگوریتم کرم شب‌تاب[۶] (FA)[۵]، الگوریتم جستجوی گرانشی[۷] (GSA)[۶]، بهینه‌سازی مبتنی بر جغرافیای زیستی[۸] (BBO)[۷]، جستجوی فاخته[۹] (CS)[۸]، الگوریتم خفاش[۱۰] (BA)[۹]، بهینه‌ساز شیرمورچه[۱۱] (ALO) [۱۰]، بهینه‌ساز گرگ خاکستری[۱۲] (GWO)[۱۱]، بهینه‌ساز چند جهانی[۱۳] (MVO)[۱۲]، الگوریتم بهینه‌سازی سینوس کسینوس[۱۴] (SCA)[۱۳]، الگوریتم بهینه‌سازی پروانه-شعله[۱۵] (MFO)[۱۴] و الگوریتم بهینه‌سازی وال[۱۶] (WOA)[۱۵] اشاره کرد.

فرایند جستجو در الگوریتم‌های بهینه‌سازی اکتشافی را می‌توان به فازهای اکتشاف[۱۷] و بهره‌برداری[۱۸] تقسیم کرد[۱۶-۱۷]. در فاز اکتشاف الگوریتم نیاز به عملگرهای تصادفی دارد تا بتواند فضای جستجو را به‌طور سراسری پوشش دهد. این کار از طریق تغییرات ناگهانی راه‌حل‌های کاندید حاصل می‌شود. در فاز بهره‌برداری، جستجوی محلی در اطراف ناحیه امیدبخش حاصل از فاز اکتشاف صورت می‌گیرد. یک الگوریتم بهینه‌سازی در صورتی می‌تواند به‌عنوان یک بهینه‌ساز سراسری در نظر گرفته شود که شامل عملگرهایی برای پوشش دادن این دو فاز باشد. تعیین مقدار سهم این دو فاز از جستجو در هر مرحله از حل مسئله یکی از نکات بسیار مهم است. عدم تعادل فاز اکتشاف و بهره‌برداری در برخی الگوریتم‌های فرااکتشافی باعث

---

[۱] meta-heuristic
[۲] Genetic Algorithm (GA)
[۳] Particle Swarm Optimization (PSO)
[۴] Anti Colony Optimization (ACO)
[۵] Artificial Bee Colony (ABC)
[۶] Firefly Algorithm (FA)
[۷] Gravitational Search Algorithm (GSA)
[۸] Biogeography-Based Optimization algorithm (BBO)
[۹] Cuckoo Serach (CS)
[۱۰] Bat Algorithm (BA)
[۱۱] Ant Lion Optimization (ALO)
[۱۲] Grey Wolf Optimization (GWO)
[۱۳] Multi Verse Optimization (MVO)
[۱۴] Sine Cosine Optimization (SCO)
[۱۵] Moth Flame Optimization (MFO)
[۱۶] Whale Optimization Algorithm (WOA)
[۱۷] exploration
[۱۸] exploitation

می‌شود این الگوریتم‌ها عملکرد مناسبی به‌عنوان یک بهینه‌سازی سراسری نداشته باشند و توانایی این الگوریتم‌ها در رسیدن به بهینه سراسری کاهش می‌یابد. از سوی دیگر در برخی الگوریتم‌های فرااکتشافی که تعادل مناسبی بین فازهای اکتشاف و بهره‌برداری و جود دارد، می‌توان با بهبود متعادل این دو فاز، فرایند جستجوی الگوریتم را بهبود بخشید.

به‌طور کلی پژوهش‌های انجام شده در این حوزه را می‌توان به سه گروه اصلی تقسیم کرد: الف) ارائه الگوریتم‌های جدید ب) بهبود روش‌های موجود و یا ج) ترکیب با الگوریتم‌های دیگر [۱۳]. الگوریتمهای جدید در یکی از چهار گروه ۱) الگوریتمهای تکاملی (مانند الگوریتم ژنتیک و استراتژی تکاملی)، ۲) تکنیکهای هوش ازدحامی[۱] (مانند الگوریتم ازدحام ذرات و بهینه سازی کلونی مورچگان)، ۳) تکنیکهای مبتنی بر فیزیک (مانند جستجوی گرانشی و حفره سیاه) و ۴) الگوریتمهای مبتنی بر انسان (مانند الگوریتم قهرمانی لیگ و الگوریتم انفجار معدن) طبقه بندی می شوند. از این بین الگوریتم بهینه‌سازی وال، به عنوان الگوریتم پایه این مقاله، در گروه دوم قرار می گیرد.

در مقابل برای بهبود روشهای موجود رهکارهای بسیاری وجود دارد. از محبوب‌ترین روش‌ها به منظور بهبود روش‌های بهینه‌سازی می‌توان به الف) بهبود برخی ساختارهای جستجوی الگوریتم‌ها، ب) استفاده از نگاشت‌های آشوبی، ج) استفاده از عملگرهای تکاملی و جستجوهای محلی اشاره کرد [۱۸-۲۱]. بهبودهایی که از دیدگاه سوم استفاده می کنند، به دلیل اینکه با اضافه کردن مجموعه ای از الگوریتمها سعی در بهبود جستجو از جمله جستجوی محلی و سراسری دارند، سربار کمتری به سیستم تحمیل می کنند و در برخی موارد بسیار موثر هستند. از سایر تکنیکهای از جمله الگوریتم‌های بهبود یافته با استفاده از عملگرهای تکاملی می‌توان به ترکیب الگوریتم بهینه‌سازی ازدحام ذرات با تکامل داروین [۲۲]، ترکیب الگوریتم چندجهانی با مکانیزم تولیدمثل الگوریتم ژنتیک [۲۴]، ترکیب الگوریتم تکامل تفاضلی با عملگر ادغام [۲۳]، ترکیب الگوریتم کلونی زنبورعسل مصنوعی با عملگر ادغام الگوریتم ژنتیک [۲۴-۲۵] اشاره کرد.

در این پژوهش با استفاده از عملگرهای تکاملی و یک سیستم استنتاج فازی[۲] (FIS)، فرایند جستجوی الگوریتم WOA بهبود داده می‌شود. در این روش با استفاده از نظریه تکامل داروین، عناصر جستجویی که موقعیت بهتری دارند در طول فرایند بهینه‌سازی باقی می‌مانند و عناصر جستجوی کم‌اهمیت‌تر از بین می‌روند. استفاده از عملگرهای ژنتیک باعث بهبود فاز اکتشاف و بهره‌برداری الگوریتم WOA می‌شود. تنظیم احتمال عملگرهای ژنتیک توسط یک FIS انجام می‌شود. این کار باعث تنظیم پویای احتمال عملگرهای ژنتیک در طول فرآیند بهینه‌سازی شده و تعادل مناسبی را بین فازی‌های اکتشاف و بهره‌برداری فراهم می‌کند. در روش پیشنهادی از دو عملگر ادغام و جهش استفاده می‌شود. استفاده از عملگر ادغام در الگوریتم WOA باعث بهبود فاز بهره‌برداری الگوریتم می‌شود. از سوی دیگر به منظور بهبود فاز اکتشاف الگوریتم از عملگر جهش استفاده می‌شود. استفاده از این دو عملگر باعث بهبود همزمان فازهای اکتشاف و بهره‌برداری الگوریتم شده و عملکرد الگوریتم را رسیدن به بهینه سراسری بهبود می‌بخشد. بهترین عامل جستجو برای ادغام یا جفت‌گیری با دیگر عوامل جستجو در هر نسل با استفاده

[۱] swarm intelligence

[۲] Fuzzy Inference System (FIS)

از یک احتمال ادغام انتخاب می‌شـود. این کار باعث می‌شـود عوامل جسـتجو تحت تأثیر بهترین موقعیت قرار گیرند و باعث بهبود فاز بهره‌برداری الگوریتم می‌شود. در مرحله بعد با استفاده از احتمال جهش، عملگر جهش روی عوامل جستجو صورت می‌گیرد. این کار باعث بهبود فاز اکتشـاف الگوریتم می‌شـود، زیرا با فراهم کردن شـرایط تصادفی بیشتر برای الگوریتم، باعث می‌شـود الگوریتم توانایی بیشتری در پوشش دادن فضای جستجو داشته باشد. تنظیم احتمال جهش و احتمال ادغام به صورت غیرخطی و با استفاده از یک FIS انجام می‌شود. FIS براساس شماره تکرار و بهترین موقعیت جاری پارامترهای احتمال جهش و احتمال ادغام را تنظیم می‌کند. در ابتدای فرآیند جستجو، الگوریتم WOA، هیچ اطلاعی در مورد راه‌حل بهینه ندارد، به همین دلیل راه‌حل‌های متنوع به‌منظور پوشش نقاط مختلف فضای جستجو بسیار ضروری است، بنابراین احتمال جهش زیاد می‌شود. با گذشـت زمان و با نزدیک شـدن به بهترین راه‌حل، جسـتجوی تصـادفی باید کاهش یابد، این در حالی اسـت که جستجوی محلی باید بیشتر شود، که این کار از طریق بیشتر شدن احتمال ادغام انجام می‌شود.

ادامه مقاله به صورت زیر سازماندهی شده است. مفاهیم پایه شامل الگوریتم WOA در بخش دوم و سوم ارائه شده است. در بخش (۴) روش پیشـنهادی ارائه شـده اسـت. بررسـی عملکرد الگوریتم روش پیشـنهادی روی توابع محک به ترتیب در بخش‌های (۵) و (۶) مورد آزمایش قرار گرفته است. در انتها نیز، نتیجه گیری آورده شده است.

۲- الگوریتم بهینه سازی وال (WOA)

با الهام از رفتار شـکار وال‌های کوهان‌دار الگوریتم فرااکتشـافی جدیدی به نام الگوریتم بهینه‌سـازی وال (WOA) ارائه شده اسـت [۱۵]. الگوریتم WOA از سـه عملگر محاصـره شـکار ، رفتار جسـتجوی غذای حباب-شـبکه و جستجو برای شکار وال‌های کوهان‌دار برای شبیه‌سازی استفاده می‌کند.

۲-۱- محاصره شکار

به منظور محاصـره شـکار باید موقعیت شـکار مشـخص شود، اما این موقعیت در فضای جستجو مشخص نیست، بنابراین در WOA بهترین موقعیت کاندید موجود به عنوان هدف شـکار در نظر گرفته می‌شود و سایر عوامل جستجو موقعیت خود را براساس آن به‌روزرسانی می‌کنند. این رفتار توسط روابط (۱) و (۲) مشخص می‌شود.

$$\vec{D} = \left|\vec{C}.\vec{X}^*(t) - \vec{X}(t)\right| \quad (۱)$$

$$\vec{X}(t+1) = \vec{X}^*(t) - \vec{A}.\vec{D} \quad (۲)$$

که در این روابط t نشـان‌دهنده تکرار فعلی، $\vec{X}^*$بردار موقعیت بهترین راه‌حل به‌دست‌آمده تاکنون، $\vec{X}$ بردار موقعیت و $\vec{A}$ و $\vec{C}$ بردارهای ضرایب هستند که مقادیر آن‌ها با استفاده از روابط (۳) و (۴) محاسبه می‌شوند:

$$\vec{A} = 2\vec{a}\cdot\vec{r} - \vec{a} \quad (۳)$$

$$\vec{C} = 2 \cdot \vec{r} \qquad (۴)$$

که در این روابط $\vec{a}$به طور خطی از ۲ به ۰ کاهش می‌یابد و $\vec{r}$ یک بردار تصادفی در [۰،۱] است.

۲-۲- جستجوی غذای حباب-شبکه

والهای کوهان‌دار با استراتژی حباب-شبکه به شکار حمله می‌کنند. مدل‌سازی ریاضی این رفتار در WOA توسط دو رویکرد مکانیزم انقباض محاصره و به‌روزرسانی مارپیچ طراحی شده است.

مکانیزم انقباض محاصره توسط کاهش مقدار $\vec{a}$ در رابطه (۳) حاصل می‌شود. این کاهش مقدار $\vec{a}$ باعث کاهش تغییرات محدوده $\vec{A}$ می‌شود.

در روش به‌روزرسانی مارپیچ، ابتدا فاصله بین موقعیت وال و موقعیت شکار محاسبه می‌شود. سپس یک رابطه مارپیچ بین موقعیت وال و شکار برای تقلید حرکت مارپیچ شکل والهای کوهان‌دار به‌صورت رابطه (۵) ساخته می‌شود:

$$\vec{X}(t+1) = \vec{D}'.e^{bl}.cos(2\pi l) + \vec{X}^*(t) \qquad (۵)$$

که در این رابطه l یک عدد تصادفی در [0,1]، b یک ثابت برای تعریف شکل مارپیچی لگاریتمی و $\vec{D}'$ فاصله iامین وال از بهترین راه‌حل به‌دست‌آمده تاکنون است و به‌صورت $\vec{D}' = \left|\vec{X}^*(t) - \vec{X}(t)\right|$ محاسبه می شود.

والهای کوهان‌دار در اطراف شکار به صورت همزمان در داخل یک انقباض دایره و در امتداد یک مسیر مارپیچ شکل شنا می‌کنند. برای مدل‌سازی این رفتارهای همزمان، به منظور به‌روزرسانی موقعیت والها در طول فرایند بهینه‌سازی، فرض می‌شود هر یک از رفتارهای انقباض محاصره و مدل مارپیچ با یک احتمال ۵۰٪ انتخاب شوند.

$$\vec{X}(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A}.\vec{D}, & if\ p < 0.5 \\ \vec{D}'.e^{bl}.cos(2\pi l) + \vec{X}^*(t), & if\ p \geq 0.5 \end{cases} \qquad (۶)$$

که در این رابطه p یک عدد تصادفی در [۰،۱] است.

۲-۳- جستجو برای شکار

والهای کوهان‌دار جستجوی تصادفی را براساس موقعیت یکدیگر انجام می‌دهند. بنابراین در WOA مقدار $\vec{A}$ به صورت مقادیری تصادفی بزرگ‌تر از ۱ یا کوچک‌تر از ۱- به منظور اجبار عامل جستجو به حرکت دور شدن از وال مرجع تعیین می‌گردد. از سوی دیگر، موقعیت یک عامل جستجو براساس یک انتخاب تصادفی عامل جستجو به جای بهترین عامل جستجو به‌دست‌آمده تاکنون به‌روزرسانی می‌شود. این مکانیزم‌ها به WOA اجازه می‌دهد که یک جستجوی سراسری را انجام دهد. مدل ریاضی به صورت روابط (۷) و (۸) است.

$$\vec{D} = \left|\vec{C}.\overrightarrow{X_{rand}} - \vec{X}\right| \quad (۷)$$

$$\vec{X}(t+1) = \overrightarrow{X_{rand}} - \vec{A}.\vec{D} \quad (۸)$$

که در این روابط $\overrightarrow{X_{rand}}$ یک بردار موقعیت تصادفی انتخاب شده از جمعیت موجود است. شبه کد WOA در شکل ۱ نشان داده شده است.

بر روی الگوریتم بهینه سازی نهنگ، بهبودهای متعددی در سالهای اخیر انجام شده است. از این جمله می توان به [۲۶] اشاره نمود که یک پارامتر انتخاب منحصر به فرد برای متعادل کردن فاز جستجوی جهانی و محلی الگوریتم معرفی کرده است. همچنین در مرحله اکتشاف، حرکت تصادفی مجاز است تا بار محاسباتی الگوریتم را کاهش دهد و وزن اینرسی در مرحله بهره برداری برای جستجوی جامع در نزدیکی بهترین راه حل معرفی شده است. در [۲۷] یک مکانیزم ادغام و سه استراتژی جستجوی موثر به نام‌های مهاجرت، انتخاب ترجیحی، و طعمه محاصره‌شده غنی‌شده برای بهبود الگوریتم بهینه سازی نهنگ پیشنهاد شده است. در [۲۸]، یک الگوریتم بهینه‌سازی نهنگ هیبریدی (NHWOA) برای افزایش سرعت هم‌گرایی و پوشش جستجو در حل مسائل بهینه‌سازی سراسری پیشنهاد شده‌است. در این الگوریتم، از تکنیک نیچینگ برای ارتقای تنوع جمعیت و مهار همگرایی زودرس در جستجوی بهترین راه حل سراسری استفاده شده است. نویسندگان [۲۹]، اصلاحات مختلفی در معادلات به روز رسانی موقعیت الگوریتم اولیه نهنگ ارائه کرده اند و از طرفی ساختار الگوریتم را نیز اصلاح نموده اند. همچنین در مقاله [۳۰]، از تکنیک جستجوی ممنوعه برای کاهش فضای جستجو و تسریع الگوریتم استفاده شده است.در مقاله [۳۱] از مفاهیم نظریه بازیها و نقطه تعادل نش به منظور بهبود عملکرد الگوریتم بهینه سازی وال استفاده شده است. الگوریتم بهینه سازی وال از جمله الگوریتمهای پرکاربرد در حوزه بهینه سازی است. از جمله آخرین کاربردهای الگوریتم وال در بهینه سازی می توان به کاربرد آن در انتخاب ویژگی [۳۲]، بهینه سازی جریان کاری در صنعت [۳۳]، بهینه سازی پارامترهای ماشین پیشگویی بردار پشتیبان [۳۴] و بهینه سازی پارامترهای سیستم استنتاج فازی [۳۵] اشاره نمود.

۳- سیستم استنتاج فازی

در تئوری مجموعه فازی ابزاری قوی جهت مواجه با عدم قطعیت ناشی از ابهام است. اگرچه سیستم‌های فازی پدیده‌های غیرقطعی و نامشخص را توصیف می‌کنند با این حال خود تئوری فازی یک تئوری دقیق است [۳۶]. منطق فازی برای اولین بار توسط لطفی زاده ارائه شد [۳۷]. منطق فازی در مقابل منطق ارسطویی قرار دارد [۳۸]. سیستم استنتاج فازی (FIS) درواقع سیستمی است که تجربیات بشر را با توابع عضویت و قوانین فازی پیاده‌سازی می‌کند و یک روش عمومی برای ترکیب دانش، فن‌آوری هوشمند، کنترل و تصمیم‌گیری است، از مهم‌ترین الگوریتم‌های استنتاج فازی می‌توان به الگوریتم استنتاج ممدانی، تاکاگی سوگنو و سوکماتو اشاره کرد که در میان این الگوریتم‌ها، الگوریتم استنتاج فازی ممدانی و سوگنو بیشترین کاربردها را دارند. به‌طورکلی هر FIS از سه جزء اصلی فازی سازی، موتور استنتاج فازی و غیرفازی سازی تشکیل شده است. تبدیل متغیرهای صریح به متغیرهای زبانی را فازی سازی می‌گویند. موتور استنتاج با استفاده از الگوریتم‌های استنتاج، قوانین را

ارزیابی و اسـتنتاج می‌کند و پس از تجمیع قوانین خروجی توسط واحد غیرفازی ساز به مقدار صریح یا عددی تبدیل می‌شود [۳۹].

Initialize the whales population X_i (i = 1, 2, ..., n)

Calculate the fitness of each search agent

X^*=the best search agent

***while** (t < maximum number of iterations)*

***for** each search agent*

Update a, A, C, l, and p

***if1** (p<0.5)*

***if2** (|A|<1)*

Update the position of the current search agent by the Eq. (1)

***else if2** (|A|≥1)*

Select a random search agent (X_{rand})

Update the position of the current search agent by the Eq. (8)

end if2

***else if1** (p≥0.5)*

Update the position of the current search by the Eq. (5)

end if1

end for

Check if any search agent goes beyond the search space and amend it

Calculate the fitness of each search agent

Update X^* if there is a better solution

t=t+1

end while

return X^*

شکل ۱. شبه کد WOA [۱۸]

استفاده از FIS ها روز به روز افزایش پیدا کرده است. یکی از کاربردهای FIS ها استفاده از آن‌ها در روش‌های بهینه‌سازی است. در اغلب الگوریتم‌های بهینه‌سازی پارامترهای خطی وجود دارند که کنترل میزان مشارکت فازهای اکتشاف و بهره‌برداری

الگوریتم را بر عهده دارند. استفاده از FIS به جای استفاده از پارامتر خطی گزینه‌ای مناسب در جهت بهبود فرایند جستجوی الگوریتم به حساب می‌آید.

۴- روش پیشنهادی

الگوریتم WOA مانند سایر الگوریتم‌های فرااکتشافی دارای دو فاز اکتشاف و بهره‌برداری است، که در این پژوهش با بهبود هر دو فاز اکتشاف و بهره‌برداری با استفاده از عملگرهای ژنتیک، عملکرد این الگوریتم را بهبود داده‌ایم. همچنین با استفاده از نظریه تکامل داروین، عناصر جستجویی که موقعیت بهتری دارند طول عمر بیشتری خواهند داشت و عناصر جستجوی کم‌اهمیت‌تر زودتر می‌میرند. همچنین از یک FIS به منظور تنظیم پویای میزان مشارکت فاز اکتشاف و بهره‌برداری استفاده کرده‌ایم. در ادامه به معرفی بخش‌های مختلف روش پیشنهادی خواهیم پرداخت.

۴-۱- بهبود فاز بهره‌برداری با استفاده از عملگر ادغام

الگوریتم ژنتیک گل‌میخ (Stud GA) یکی از انواع الگوریتم‌های ژنتیک است که در آن گل‌میخ (ژنوم بهینه) برای ادغام در هر تولیدمثل فراهم می‌شود [۴۰]. به‌عبارت‌دیگر، در این الگوریتم گل‌میخ برای جفت‌گیری با دیگر فرزندان جدید استفاده می‌شود. در الگوریتم Stud GA ابتدا یک جمعیت اولیه به صورت تصادفی ایجاد شده و سپس گل‌میخ برای جفت‌گیری انتخاب می‌شود. پس از مشخص شدن گل‌میخ عمل ادغام انجام می‌شود. این فرایند تا فراهم شدن معیار توقف الگوریتم این حلقه تکرار می‌شود. به منظور بهبود فاز بهره‌برداری الگوریتم WOA، از عملگر ادغام الگوریتم Stud GA استفاده شده است. این عملگر ادغام، ویژگی تکامل طبیعی را به‌منظور جفت‌گیری گل‌میخ با دیگر عامل‌های جستجو فراهم می‌کند این کار باعث می‌شود که تولیدمثل‌های جدید، راه‌حل‌های بهتری برای هر عامل جستجو ایجاد کند.

ایده اصلی این عملگر ادغام، توسعه گل‌میخ برای جفت‌گیری با دیگر عامل‌های جستجو است که به‌منظور تولید فرزندان جدید در مکانی که یک راه‌حل خیلی خوب وجود ندارد انجام می‌شود. در این روش، پس از پیدا کردن گل‌میخ در گام اول و با توجه به احتمال ادغام، عمل ادغام به منظور تولید فرزندان جدید در گام دوم انجام می‌شود. در گام سوم، کیفیت فرزند تولید شده جدید ارزیابی می‌شود و در صورتی که فرزند تولید شده جدید بهتر بود، پذیرفته می‌شود. این عملگر باعث بهبود فاز بهره‌برداری الگوریتم WOA می‌شود.

۴-۲- بهبود فاز اکتشاف با استفاده از عملگر جهش

بهبود فاز اکتشاف الگوریتم نیازمند تغییرات تصادفی بیشتر در راه‌حل‌های کاندید است، که این شرایط با استفاده از عملگرهای تصادفی حاصل می‌شود. در روش پیشنهادی عملگر جهش برای بالا بردن تنوع جامعه‌ی جواب و افزایش قدرت جستجو در الگوریتم WOA استفاده شده است. با استفاده از این عملگر و با توجه به احتمال جهش، موقعیت برخی از راه‌حل‌های کاندید به صورت تصادفی تغییر می‌کنند. این عملگر یک پوشش سراسری در کل فضای جستجو را تضمین می‌کند و باعث بهبود فاز اکتشاف الگوریتم WOA می‌شود.

۴-۳- تعادل پویا فاز اکتشاف و بهره‌برداری با استفاده از سیستم استنتاج فازی

به منظور برقرار کردن یک تعادل پویا بین میزان مشارکت فازهای اکتشاف و بهره‌برداری الگوریتم WOA، از یک FIS استفاده شده است. در این روش مقادیر احتمال ادغام (Pc)، احتمال جهش (Pm) و پارامتر خطی a توسط این FIS تعیین می‌گردد. از آنجا که مقادیر Pm، Pc در بازه [۰،۱] و مقدار a در بازه [۰،۲] تغییر می کنند و از طرفی هر متغیر دارای ۳ متغیر زبانی پایین، متوسط و بالا طراحی شده‌است، مرکز ثقل توابع فازی مثلثی در نقاط ۰,۲۵، ۰,۵ و ۰,۷۵ قرار گرفته است. همین کار برای متغیرهای فازی "بهترین راه‌حل" و "تکرار" نیز، البته بعد از هنجارسازی به بازه [۰،۱]، انجام شده است. طول توابع فازی مذکور نیز برابر و به میزان ۰,۵ در نظر گرفته شده است. در ابتدای فرایند بهینه‌سازی تنظیم این پارامترها به‌گونه‌ای انجام می‌شود که تأکید بر فاز اکتشاف الگوریتم صورت گیرد.

در ادامه فرایند بهینه‌سازی و به‌منظور رسیدن به بهترین راه‌حل، FIS تنظیم پارامترها را در جهت تأکید بر جستجوی محلی انجام می‌دهد. این کار باعث تعادل پویا بین جستجوی سراسری و جستجوی محلی شده و در نتیجه عملکرد الگوریتم بهبود می‌یابد. براساس این مکانیزم، قوانین فازی FIS به صورت جدول ۱ تعیین شده‌اند. قوانین جدول ۱ با توجه به سایر کارهای مشابهی که در زمینه استفاده از منطق فازی در تعیین پارامترهای الگوریتمهای بهینه سازی معرفی شده اند طراحی شده اند و البته هریک منطق خاص خود را دارند. مثلا هر چه تعداد تکرارها (نسلها) بیشتر می‌شود، همانطور که در بخش ۲-۱ و توضیح رابطه ۳ و ۴ بیان شد، a کاهش می‌یابد. یا هر چه تعداد نسلها بیشتر می‌شود معمولا برای بهبود جستجوی محلی احتمال ادغام (Pc) بیشتر و احتمال جهش (Pm) کمتر می‌شود.

جدول ۱: قوانین فازی FIS

شماره قانون	ورودی		خروجی		
	بهترین راه‌حل	تکرار	احتمال جهش (Pm)	احتمال ادغام (Pc)	ضریب خطی a در رابطه ۳
۱	پایین	پایین	بالا	پایین	بالا
۲	متوسط	پایین	متوسط	متوسط	متوسط
۳	بالا	پایین	بالا	پایین	بالا
۴	پایین	متوسط	بالا	پایین	بالا
۵	متوسط	متوسط	متوسط	متوسط	متوسط
۶	بالا	متوسط	بالا	پایین	بالا
۷	پایین	بالا	پایین	بالا	پایین
۸	متوسط	بالا	پایین	بالا	پایین
۹	بالا	بالا	متوسط	متوسط	متوسط

۵- فلوچارت الگوریتم پیشنهادی

در الگوریتم پیشنهادی پس از مقداردهی اولیه پارامترها و جمعیت، مقادیر پارامترهای a، Pm و Pm توسط FIS تعیین می‌گردد. سپس الگوریتم استاندارد WOA اجرا می‌شود. در مرحله بعد یک عدد تصادفی ایجاد شده و در صورتی که این مقدار تصادفی کوچک‌تر یا مساوی احتمال ادغام باشد، عملگر ادغام اجرا می‌شود. در مرحله بعد عدد تصادفی دیگری ایجاد شده و در صورتی که این مقدار تصادفی کوچک‌تر یا مساوی احتمال جهش باشد، عملگر ادغام اجرا می‌شود. این فرایند تا فراهم شدن معیار پایان که رسیدن به حداکثر تعداد تکرارهای تعریف شده است تکرار می‌شود. شبه کد الگوریتم پیشنهادی در شکل ۲ نشان داده شده است.

Initialize the whales' population X_i *(i = 1, 2, ..., n).*

Calculate the fitness of each search agent.

X^**=the best search agent.*

while *(t < maximum number of iterations)*

Update Pm, Pc and a by FIS.

for *each search agent*

Update a, A, C, l, and p.

if1 *(p<0.5)*

if2 *(|A|<1)*

Update the position of the current search agent by the Eq. (1).

else if2 *(|A|≥1)*

Select a random search agent (X_{rand}*).*

Update the position of the current search agent by the Eq. (8).

end if2

else if1 *(p≥0.5)*

Update the position of the current search by the Eq. (5).

end if1

if3 *Rand()<=Pm*

Choose the best stud for mating (X_{stud}*).*

Generate new search agent by crossover (X_{new}*).*

If4 *fitness(*X_{new}*)>fitness(*X_{stud}*)*

accept X_{new}*.*

else if4

discard X_{new}*.*

end if4

end if3

if5 *Rand()<=Pc*

Choose random search _agent (X_{rand}*).*

$X_{rand}=X_{rand}*C.$

end if5

Calculate the fitness of each search agent.

end for

Check if any search agent goes beyond the search space and amend it.

Calculate the fitness of each search agent.

Update X^* if there is a better solution.

t=t+1.

end while

return X^*.

شکل ۲. شبه کد الگوریتم پیشنهادی

۶- نتایج و بحث

با استفاده از بیست و سه تابع محک که شامل توابع محک تک مد، چند مد و چند مد با ابعاد ثابت است، روش پیشنهادی مورد ارزیابی قرار گرفته‌است. مشخصات این توابع در جدول‌های ۲، ۳ و ۴ آورده شده است. در این جدول M، D، R و fopt به ترتیب نشان‌دهنده ابعاد تابع، ماهیت تابع، محدوده فضای جستجوی تابع و مقدار بهینه تابع هستند. همچنین اندازه جمعیت و حداکثر تولیدمثل به ترتیب ۵۰ و ۱۰۰ در نظر گرفته شده است.

جدول ۲: توابع محک تک مد (Unimodal benchmark functions)

Name	Function	D	R	f_{opt}
Sphere	$f_1(x)=\sum_{i=1}^{n} x_i^2$	۱۰	[-۱۰۰، ۱۰۰]	۰
Schwefel 2.22	$f_2(x)=\sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	۱۰	[-۱۰ ، ۱۰]	۰
Schwefel 1.2	$f_3(x)=\sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j\right)^2$	۱۰	[-۱۰۰، ۱۰۰]	۰
Schwefel 2.21	$f_4(x)=max_i\{\lvert x_i\rvert, 1\le i\le n\}$	۱۰	[-۱۰۰، ۱۰۰]	۰
Rosenbrock	$f_5(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	۱۰	[-۳۰، ۳۰]	۰
Step	$f_6(x)=\sum_{i=1}^{n}([x_i+0.5])^2$	۱۰	[-۱۰۰، ۱۰۰]	۰
Quartic	$f_7(x)=\sum_{i=1}^{n} ix_i^4+random[0,1)$	۱۰	[-۱,۲۸، ۱,۲۸]	۰

جدول ۳: توابع محک چند مد (Multimodal benchmark functions)

Name	Function	D	R	f_{opt}
Schwefel 2.26	$f_8(x)=\sum_{i=1}^{n}-x_i sin\left(\sqrt{\lvert x_i\rvert}\right)$	۱۰	[−۵۰۰، ۵۰۰]	−۴۱۸,۹۸۲۸×۵
Rastrigin	$f_9(x)=\sum_{i=1}^{n}[x_i^2-10cos(2\pi x_i)+10]$	۱۰	[−۵,۱۲، ۵,۱۲]	۰
Ackley	$f_{10}(x)=-20\,exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-exp\left(\frac{1}{n}\sum_{i=1}^{n}cos(2\pi x_i)\right)+20+e$	۱۰	[−۳۲، ۳۲]	۰
Griewank	$f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	۱۰	[−۶۰۰، ۶۰۰]	۰
Generalized Penalized	$f_{12}(x)=\frac{\pi}{n}\{10\,sin(\pi y_1)+\sum_{i=1}^{n-1}(y_1-1)^2[1+sin^2(\pi y_{1+1})]+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4}$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\0 & -a<x_i<a\\k(-x_i-a)^m & x_i<-a\end{cases}$	۱۰	[−۵۰، ۵۰]	۰
Penalty	$f_{13}(x)=0.1\{sin^2(3\pi x_i)+\sum_{i=1}^{n}(x_i-1)^2[1+sin^2(3\pi x_i+1)]+(x_i-1)^2[1+sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	۱۰	[−۵۰، ۵۰]	۰

جدول ۴: توابع محک چند مد با ابعاد ثابت (Fixed-dimension multimodal benchmark functions)

Name	Function	D	R	f_{opt}
De Joung (shekel's Foxholes)	$f_{14}(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	۲	[−۶۵، ۶۵]	۱
Kowalik	$f_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_i(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right]^2$	۴	[−۵، ۵]	۰,۰۰۰۳۰
Camel Back -6 Hump	$f_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	۲	[−۵، ۵]	−۱,۰۳۱۶
Branin	$f_{17}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_i-6\right)^2+10\left(1-\frac{1}{8\pi}\right)cos\,x_1+10$	۲	[−۵، ۵]	۰,۳۹۸

Goldstein and Price	$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	۲	[-۲، ۲]	۳
Hartman 3-Dimentional	$f_{19}(x) = -\sum_{i=1}^{4} c_i \, exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	۳	[۱، ۳]	-۳,۸۶
Hartman 6-Dimentional	$f_{20}(x) = -\sum_{i=1}^{4} c_i \, exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	۶	[۰، ۱]	-۳,۳۲
Shekel 1	$f_{21}(x) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	۴	[۰، ۱۰]	-۱۰,۱۵۳۲
Shekel 2	$f_{22}(x) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	۴	[۰، ۱۰]	-۱۰,۴۰۲۸
Shekel 3	$f_{23}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	۴	[۰، ۱۰]	-۱۰,۵۳۶۳

ابتدا به‌منظور اعتبارسنجی روش پیشنهادی، الگوریتم WOA استاندارد را با انواع ترکیب‌های تکاملی روش WOA مقایسه کرده‌ایم. برای این منظور الگوریتم استاندارد WOA را با الگوریتم WOA داروینی (DWOA)، الگوریتم WOA داروینی و سیستم استنتاج فازی (DFWOA)، الگوریتم WOA داروینی و FIS و عملگر ادغام (DFCWOA) و الگوریتم WOA داروینی و FIS و عملگر ادغام و عملگر جهش[۱] (DFCMWOA) که همان روش پیشنهادی است مورد مقایسه قرار دادیم. جدول ۵ میانگین و انحراف معیار مقادیر حاصل از الگوریتمها با ۵۰ بار اجرا را نشان می‌دهند.

جدول ۵: مقایسه میانگین نرمال شده خروجی الگوریتم پیشنهادی DFCMWOA با نسخه‌های دیگری از WOA

(مقادیر داخل پرانتز انحراف معیار است)

تابع	WOA	DWOA	DFWOA	DFCWOA	DFCMWOA
F01	۱,۳۲۸۰۲۹ (±۰,۰۵۲۴)	۱۰۰,۰۰۰۰۰۰ (±۰)	۱,۵۸۵۳۳۶ (±۰,۵۱۱)	۱,۵۹۲۰۷ (±۰,۳۲۴۵)	**۱,۰۰۰۰۰۰ (±۰)**
F02	۱۶,۱۵۹۶۳ (±۰,۲۶۶۵)	۱۰۰,۰۰۰۰۰۰ (±۰)	۸,۱۷۵۵۱۴ (±۰,۱۳۲۱)	۴,۳۳۶۶۷۴ (±۰,۱۵۴۸)	**۱,۰۰۰۰۰۰ (±۰)**
F03	۱۰۰,۰۰۰۰۰۰ (±۰)	۲۰,۹۸۶۶۷ (±۱,۰۵۴۸)	۱۶,۴۸۰۵ (±۰,۹۵۱۲)	۳۸,۶۳۷۹۸ (±۱,۲۱۷۸)	**۱,۰۰۰۰۰۰ (±۰)**
F04	۱۰۰,۰۰۰۰ (±۰)	۳۱,۷۹۳۳۶ (±۰,۵۵۸۹)	۳۴,۹۳۲۸۸ (±۱,۴۶۹)	۶۳,۰۱۴۵۱۰ (±۲,۵۱۵)	**۱,۰۰۰۰۰۰ (±۰)**
F05	۹,۷۲۷۵۹۷ (±۰,۲۵۸)	۱۰۰,۰۰۰۰ (±۰)	۱,۰۰۹۴۰۷ (±۰,۰۵۱۴)	۸۵,۹۱۲۶۹ (±۴,۲۳۲)	**۱,۰۰۰۰۰۰ (±۰)**
F06	**۱,۰۰۰۰۰۰ (±۰)**	۸۲,۸۵۲۰۴ (±۲,۰۰۴۴)	۴۴,۲۳۶۶۵ (۱,۰۲۸۸۴)	۱۰۰,۰۰۰۰ (±۰)	۱۵,۹۹۹۰۵ (±۳,۱۲)
F07	۱۰۰,۰۰۰۰ (±۰)	۴۳,۳۴۵۷۶ (±۱,۰۹۴۴)	۵۸,۱۷۱۱۲ (±۲,۰۸۴۳)	۷۶,۹۳۰۸۱۱ (±۲,۱۵۸)	**۱,۰۰۰۰۰۰ (±۰)**

[۱] Darwinian Fuzzy Crossover Mutation WOA

F08	۷۹,۳۶۵۸۶۷ (±۲,۹۴۲)	۵۰,۸۹۳۸۹۴ (±۳,۱۵۵)	۳۶,۴۷۹۹۴ (±۰,۹۸۴)	۱۰۰,۰۰۰۰ (±۰)	**۱,۰۰۰۰۰۰ (±۰)**
F09	۱۰۰,۰۰۰۰ (±۰)	۵۸,۴۲۳۴۱۷ (±۱,۰۹۷)	۶۳,۵۰۰۸۷۷ (±۱,۵۸۱)	۷۴,۹۲۷۶۵۴ (±۶,۳۲۱)	**۱,۰۰۰۰۰۰ (±۰)**
F10	۱,۰۰۰۰۲۳ (±۰)	۱,۰۰۰۰۸۳ (±۰)	۱,۰۰۰۰۰۷ (±۰)	۱۰۰,۰۰۰۰ (±۰)	**۱,۰۰۰۰۰۰ (±۰)**
F11	۸۴,۶۱۲۴۷ (±۲,۰۸۴۲)	۷۴,۸۵۱۳۷ (±۳,۰۵۰۸)	۸۲,۵۱۴۱۰۴ (±۲,۵۱۵)	۱۰۰,۰۰۰۰ (±۰)	**۱,۰۰۰۰۰۰ (±۰)**
F12	۸,۳۸۲۲۶۶ (±۰,۴۹۲)	۱۶,۳۰۹۲۸۴ (±۱,۲۰۱)	۱۲,۰۱۶۴۶۰ (±۱,۰۰۵)	۱۰۰,۰۰۰۰ (±۰)	**۱,۰۰۰۰۰۰ (±۰)**
F13	۱,۱۳۱۸۷۸ (±۰,۲۵۴۴)	۱,۳۹۰۵۱۷ (±۰,۰۸۸۴)	۲۳,۷۵۴۳۵۹ (±۱,۱۵)	۱۰۰,۰۰۰۰ (±۰)	**۱,۰۰۰۰۰۰ (±۰)**
F14	۱۰۰,۰۰۰۰ (±۰)	۳۳,۹۱۹۵۹۱ (±۱,۰۹۳)	۱۷,۳۵۲۵۱ (±۲,۰۴۳۵)	**۱,۰۰۰۰۰۰ (±۰)**	۸۲,۸۸۰۴۷۰ (±۲,۸۱۲)
F15	۱۰۰,۰۰۰۰ (±۰)	۷,۶۰۲۷۷۶ (±۰,۴۳۱۲)	۴,۱۲۶۸۵۸ (±۰,۵۵۸)	**۱,۰۰۰۰۰۰ (±۰)**	۱,۴۴۶۳۱۰ (±۰,۲۱۱)
F16	۸۷,۵۶۳۱۱۸ (±۳,۶۷۳)	۱,۰۰۰۱۷۹ (±۰,۰۰۱۲)	**۱,۰۰۰۰۰۰ (±۰)**	۱۰۰,۰۰۰۰ (±۰)	۱۳,۰۹۱۳۱۱ (±۰,۰۹۵)
F17	۱۳,۰۲۴۸۷۱ (±۰,۹۶۳)	۱۳,۳۰۳۳۸ (±۱,۰۸۰۷)	۴۰,۲۹۱۴۲۴ (±۲,۴۷۸)	۱۰۰,۰۰۰۰ (±۰)	**۱,۰۰۰۰۰۰ (±۰)**
F18	۶۱,۴۶۳۱۳ (±۲,۰۱۷۸)	**۱,۰۰۰۰۰۰ (±۰)**	۱۰۰,۰۰۰۰ (±۰)	۵۶,۷۷۷۰۱۶ (±۲,۰۹۹)	۵۶,۸۳۴۴۰۳ (±۱,۹۴۲)
F19	۱۰۰,۰۰۰۰ (±۰)	۱,۰۰۰۰۳ (±۰)	**۱,۰۰۰۰۰۰ (±۰)**	۱,۲۲۷۷ (±۰,۰۰۵)	۱۲,۹۱۵۶ (±۰,۰۴۴۴)
F20	۱۰۰,۰۰۰۰ (±۰)	**۱,۰۰۰۰۰۰ (±۰)**	۳۶,۹۳۱۶ (±۲,۳۶۱)	۱,۷۱۲۷ (±۰,۰۵۷۶)	۱۷,۳۲۹۶ (±۰,۲۱۱)
F21	۱۰۰,۰۰۰۰ (±۰)	۷۷,۹۰۸۲۹ (±۳,۸۴۸)	۸۹,۶۹۰۶ (±۲,۰۸۴)	۴۷,۸۲۶۹ (±۱,۰۸۸)	**۱,۰۰۰۰۰۰ (±۰)**
F22	۱۰۰,۰۰۰۰ (±۰)	۵۵,۲۸۲۳ (±۱,۰۹۹)	۷۷,۱۰۱۹ (±۲,۰۱۶۵)	۵۲,۲۳۰۳۴ (±۱,۸۸۴)	**۱,۰۰۰۰۰۰ (±۰)**
F23	۱۰۰,۰۰۰۰ (±۰)	۳۵,۷۲۲۴ (±۲,۳۷۳)	۵۷,۰۰۶۳۸ (±۳,۲۴۹)	۵۵,۲۲۴۷ (±۲,۰۱۰۲)	**۱,۰۰۰۰۰۰ (±۰)**

جدول ۶ مقایسه میانگین نرمال شده خروجی الگوریتم پیشنهادی DFCMWOA با الگوریتم های بهینه سازی دیگر

(مقادیر داخل پرانتز انحراف معیار است)

Function	ALO	GWO	MFO	MVO	PSO	SCA	WOA	DFCMWOA
F01	۱,۳۹۵۰(±۰,۰۰۴)	**۱,۰۰۰۰ (±۰)**	۱۰۰,۰۰۰(±۰)	۴,۷۱۱(±۰,۴۶)	۱,۰۰۲(±۰,۱۲۰)	۲۴,۳۲(±۲,۵۴)	**۱,۰۰۰۰ (±۰)**	**۱,۰۰۰۰۰(±۰)**
F02	۱۰۰,۰۰ (±۰)	**۱,۰۰۰۰۰(±۰)**	۵,۷۰۴(±۰,۴۲)	۲,۷۸۰(±۰,۱۲)	۱,۱۵۳۳(±۰,۵۱۴)	۱,۵۲۵(±۰,۰۹)	**۱,۰۰۰۰(±۰)**	**۱,۰۰۰۰۰(±۰)**
F03	۱۸,۷۰۶(±۰,۷۵)	**۱,۰۰۰۰ (±۰)**	۳۰,۱۵۳(±۲,۳)	۱,۰۷۸(±۰,۰۸)	۱,۰۱۴۵(±۰,۰۶۸)	۶,۱۶۳(±۰,۲۵)	۱۰۰,۰۰ (±۰)	**۱,۰۰۰۰ (±۰)**
F04	۲۷,۰۴۵(±۱,۰۳)	۱,۰۰۲(±۰,۰۳)	۳۷,۲۴۹(±۲,۳)	۳,۰۷۶(±۰,۶۱)	۱,۳۷۹۲(±۰,۰۹۹)	۱۷,۸۳(±۰,۶۴)	۱۰۰,۰۰ (±۰)	**۱,۰۰۰۰ (±۰)**

F05	۱۰۰,۰۰۰(±۰)	**۱,۰۰۰۰۰۰ (±۰)**	۷۲,۱۳۶۸(±۳,۲۱)	۲۷,۹۵۲۹(±۲,۵۹)	۱,۷۳۳۷(±۰,۴۸)	۱۷,۰۱۷۸(±۳,۸۴)	۳,۵۱۶۲(±۰,۳۲)	۱,۱۰۳(±۰,۰۵)
F06	۵,۷۳۸۰۹(±۰,۲۶)	۱,۳۵۰۱۹(±۰,۰۷)	۱۰۰,۰۰۰(±۰)	۵,۰۷۹۶۸(±۰,۳۶)	**۱,۰۰۰۰۰ (±۰)**	۳۴,۷۰۴۲(±۱,۰۵)	۲,۲۰۹۷(±۰,۴۳)	۲,۴۱۲۳(±۰,۳)
F07	۱۰۰,۰۰۰(±۰)	۴,۹۵۸۲۶(±۰,۲۳)	۵۶,۹۵۹۲(±۲,۲۳)	۱۴,۵۵۵(±۱,۰۸)	۵۱,۱۹۲۸(±۲,۱۵)	۲۴,۲۲۳۸۹(±۱,۴)	۲۰,۰۰۵۸(±۱,۰۹)	**۱,۰۰۰۰ (±۰)**
F08	۷۵,۳۹۲۱(±۴,۹۴)	۶۴,۹۸۵۷(±۲,۶)	۲۸,۱۱۹۹(±۱,۰۳)	۵۲,۷۶۶۴(±۲,۸۴)	۱۰۰,۰۰۰(±۰)	۹۹,۱۱۱۵(±۶,۸۴)	۳۶,۵۹۰۸(±۳,۲۶)	**۱,۰۰۰۰ (±۰)**
F09	۹۱,۲۹۸۳۵۷(±۶,۳۴)	۲۰,۵۳۹۶(±۲,۲۵)	۱۰۰,۰۰۰(±۰)	۹۷,۹۲۲۲(±۵,۰۸)	۶۴,۷۲۴۱(±۱,۲۳)	۵۷,۸۱۶۳(±۳,۴۴)	۲۶,۷۹۸۱(±۲,۰۸)	**۱,۰۰۰۰ (±۰)**
F10	۱۰۰,۰۰۰(±۰)	**۱,۰۰۰۰۰ (±۰)**	۵۵,۱۸۸۳(±۴,۲۳)	۲۱,۸۹۵۱(±۱,۲۶)	۱,۳۸۵۲۲۷(±۰,۰۹)	۳۹,۴۰۵۳(±۲,۶۷)	**۱,۰۰۰۰۰۰ (±۰)**	**۱,۰۰۰۰۰ (±۰)**
F11	۱۳,۵۸۴۹(±۱,۰۲)	۴,۶۹۶۶۷(±۰,۹۸)	۵۵,۶۶۹۰(±۵,۰۴)	۳۹,۲۰۷۹(±۲,۱۲)	۱۰۰,۰۰۰(±۰)	۳۹,۴۹۸۵(±۱,۹۶)	۹,۳۸۴۷(±۱,۰۹)	**۱,۰۰۰۰ (±۰)**
F12	۱۰۰,۰۰۰(±۰)	۱,۲۲۵۲۲(±۰,۰۲)	۱۵,۳۲۱۲(±۱,۹۴)	۸,۹۴۸۱(±۱,۴۶)	**۱,۰۰۰۰۰۰ (±۰)**	۱۲,۰۲۶۴(±۱,۰۲)	۴,۷۳۵۸۴(±۰,۰۹)	۱,۳۸۱ (۰,۰۷)
F13	۷۷,۱۰۰۳۰۸(±۶,۱۲)	۱,۷۸۹۰۶(±۰,۳۲)	۱۰۰,۰۰۰(±۰)	۲,۹۰۵۲۸۷(±۰,۳)	**۱,۰۰۰۰۰۰ (±۰)**	۵۷,۴۰۵۸(±۴,۹۲)	۱۰,۵۴۸۳(±۱,۰۳)	۸,۵۸۱۲(±۱,۳)
F14	۱۰۰,۰۰۰(±۰)	۶۴,۳۶۹۳(±۴,۱۸)	**۱,۰۰۰۰ (±۰)**	۲,۸۵۵۰۱(±۰,۵۶)	۲۸,۲۰۵۰۱۲(±۲,۱۳)	۱۶,۷۰۱۰۵(±۲,۱)	۵۷,۵۰۲۵(±۶,۱۱)	۱۶,۶۴۱۶۹(±۲)
F15	۱۰۰,۰۰۰(±۰)	۳۱,۰۱۶۶(±۱,۳۲)	۹,۳۹۵۱۹(±۱,۶۲)	۸۲,۱۷۱۹(±۶,۵۴)	۶,۸۸۰۹۲۱(±۰,۶۱)	۱۱,۴۷۳۳(±۱,۹۵)	۱۹,۳۵۴۴(±۱۰۶۴)	**۱,۰۰۰۰۰ (±۰)**
F16	**۱,۰۰۰۰۰۰ (±۰)**	۱,۲۱۰۴۹(±۰,۴۸)	**۱,۰۰۰۰ (±۰)**	۴,۸۱۰۲۹(±۰,۴۶)	**۱,۰۰۰۰۰۰ (±۰)**	۱۰۰,۰۰۰(±۰)	۱,۰۶۷۰۲(±۰,۰۹)	۱,۰۰۴ (±۰,۰۲)
F17	**۱,۰۰۰۰۰۰ (±۰)**	۲۸,۴۴۳۴(±۰,۲۳)	**۱,۰۰۰۰ (±۰)**	۱,۱۵۵۶۷(±۰,۰۷)	**۱,۰۰۰۰۰۰ (±۰)**	۱۰۰,۰۰۰(±۰)	۵,۸۰۳۹۶(±۱,۰۷)	۱,۰۸۱۶ (±۰,۱)
F18	**۱,۰۰۰۰۰۰ (±۰)**	۱,۰۴۵۵۶(±۰,۰۲)	**۱,۰۰۰۰۰ (±۰)**	۱,۰۰۴۹۴(±۰,۰۱)	**۱,۰۰۰۰۰۰ (±۰)**	۱,۱۰۳۰۵(±۰,۱)	۱۰۰,۰۰۰(±۰)	۵۰,۴۴۱ (±۲,۷)
F19	۱,۴۹۹۲۸۶(±۰,۰۵)	۷,۲۷۶۸۶(±۰,۴۳)	**۱,۰۰۰۰۰ (±۰)**	۱,۰۵۷۱(±۰,۱۶)	**۱,۰۰۰۰۰۰ (±۰)**	۴۳,۹۶۳۷(±۲,۹۱)	۱۰۰,۰۰۰(±۰)	۴۳,۹۸۵(±۱,۰)
F20	۸,۵۲۱۷۰۵(±۱,۰۲)	۱۵,۷۶۱۰(±۱,۶۴)	۱۳,۳۹۰۵۷(±۱,۵)	۸,۱۷۵۹۶(±۰,۱۹)	**۱,۰۰۰۰۰۰ (±۰)**	۱۰۰,۰۰۰(±۰)	۲۴,۴۰۴(±۳,۳۶)	۵,۸۹۶۴(±۱,۱)
F21	۶۴,۵۷۸۸۹۴(±۴,۹۲)	۲۱,۱۵۳۲(±۱,۶۱)	۳۱,۱۸۸۳(±۱,۷۳)	۵۲,۵۰۰۳(±۴,۶۵)	۵۵,۷۴۲۸۳۴(±۴,۹)	۱۰۰,۰۰۰(±۰)	۳۶,۶۴۷۲(±۲,۹۳)	**۱,۰۰۰۰۰ (±۰)**
F22	۵۵,۱۶۰۴۱۷(±۴,۹۶)	**۱,۰۰۰۰۰۰ (±۰)**	۲۴,۰۳۸۳۴(±۳,۱)	۴۲,۲۶۴۹(±۳,۱۹)	۲۶,۰۱۰۷۷۴(±۳,۲)	۱۰۰,۰۰۰(±۰)	۴۸,۳۴۱۹(±۲,۰۸)	۲,۷۹۸ (±۱,۴)
F23	۶۸,۰۵۳۳(±۵,۳۶)	**۱,۰۰۰۰۰۰ (±۰)**	۱۶,۱۷۸۰(±۱,۴۳)	۴۲,۱۷۷(±۲,۰۹)	۳۵,۰۳۳۰۳(±۲,۸۶)	۱۰۰,۰۰۰(±۰)	۶۰,۰۳۵(±۵,۳۲)	۷,۲۲۳(±۰,۲۶)

در این جدول‌ها F01…F23 نشـــان‌دهنده توابع محک جدول‌های ۲، ۳ و ۴ هســـتند. همچنین بهترین مقادیر به‌دســـت‌آمده برجسـته شـده‌اند. مقایسـه نتایج نشان می‌دهد الگوریتم پیشنهادی روی ۱۱ تابع محک از ۲۳ تابع محک عملکرد بهتری نسبت سایر الگوریتمهای مورد بررسی داشته است. این بررسی نشان می‌دهد روش پیشنهادی بهترین روش ترکیبی در بین روش‌های ترکیبی مورد بررسـی بوده است. زیرا بهبود هر دو فاز اکتشاف و بهره‌برداری الگوریتم توسط عملگرهای جهش و ادغام باعث افزایش قدرت جسـتجوی الگوریتم می‌شـود. همچنین تعادل بهتر با اسـتفاده از کنترل پویای میزان مشارکت این دو فاز باعث تعادل بهتر این دو فاز الگوریتم WOA شده است.

عملکرد روش پیشنهادی روی بیست و سه تابع محک جدول‌های ۲، ۳ و ۴ با هفت الگوریتم بهینه‌سازی مورد مقایسه قرار گرفته شده است. این الگوریتم‌ها شامل الگوریتم WOA استاندارد، الگوریتم PSO [۲] و پنج الگوریتم از جدیدترین الگوریتم‌های بهینه‌سازی شامل: ALO [۱۰]، GWO [۱۱]، MFO [۱۴]، MVO [۱۲]، SCA [۱۳] است. جدول ۶ میانگین و انحراف معیار نتایج حاصل از ۵۰ بار اجرای الگوریتم‌ها را نشان می‌دهند. همچنین بهترین مقادیر به‌دست‌آمده برجسته شده‌اند. مقایسه نتایج نشان می‌دهند روش پیشنهادی در رسیدن به راه‌حل‌های بهینه موفق‌تر بوده است به طوری که به ترتیب روی نه تابع از بیست و سه تابع محک مورد بررسی توانسته است به بهترین مقادیر بهینه دست یابد.

به منظور بررسـی سـرعت همگرایی، فرایند بهینه‌سـازی برای بیسـت‌وسـه تابع محک با اسـتفاده از الگوریتم پیشنهادی و الگوریتم‌های بهینه‌سازی مورد بررسی در شکل‌های ۳-الف تا ۳-ق نشان داده شده است.

فرایند بهینه‌سازی برای تابع Sphere (F01) در شکل ۳-الف نشان داده شده است. این شکل نشان می‌دهد روش پیشنهادی برای این تابع نه تنها به جواب بهینه بهتری دست یافته است بلکه سرعت همگرایی بیشتری نسبت به سایر الگوریتم‌های مورد بررسی دارد. الگوریتم WOA دومین الگوریتم بهتر از نظر دقت و سرعت همگرایی بوده است.

نتایج آزمایش‌ها برای Schwefel 2.22 (F02) در شکل ۳-ب نشان داده شده است. برای این تابع با وجود اینکه همه الگوریتم‌های موردبررسی سرعت همگرایی مشابهی دارند اما این الگوریتم DFCMWOA است که توانسته است به مقادیر بهینه بهتری دست یابد. برای این تابع الگوریتم WOA دومین الگوریتم بهتر در رسیدن به بهترین راه‌حل بهینه بوده است.

شکل ۳-پ و شکل ۳-ت به ترتیب فرایند بهینه‌سازی را برای تابع Schwefel 1.2 (F03) و تابع Schwefel 2.21 (F04) نشان می‌دهند. شکل ۳-پ و شکل ۳-ت نشان می‌دهند الگوریتم DFCMWOA نه تنها نسبت به الگوریتم‌های دیگر به راه‌حل بهینه بهتری دست یافته است بلکه سرعت همگرایی بالاتری نسبت به دیگر الگوریتم‌های مورد بررسی داشته است.

مقادیر بهینه‌سازی به‌دست‌آمده برای سه تابع Rosenbrock (F05)، Step (F06) و Quartic (F07) در شکل‌های ۳-ج، ۳-چ و ۳-ح نشان داده شده است. برای این سه تابع نیز الگوریتم پیشنهادی دارای سرعت همگرایی بالاتری نسبت به الگوریتم‌های مورد بررسی است.

برای تابع Rosenbrock (F05)، الگوریتم WOA الگوریتم برتر در رسیدن به راه‌حل بهینه بهتر بوده است. برای تابع Step (F06)، الگوریتم DWOA بهترین عملکرد را در مقایسه با الگوریتم‌های دیگر در رسیدن به جواب بهینه بهتر داشته است. برای دو تابع Rosenbrock (F05) و Step (F06) روش پیشنهادی بدترین الگوریتم نبوده است و به ترتیب دومین و سومین الگوریتم بهتر در رسیدن به راه‌حل بهینه بهتر بوده است. الگوریتمDFCMWOA برای تابع Quartic (F07) بهترین عملکرد را در مقایسه با دیگر الگوریتم‌های موردبررسی داشته است.

فرایند بهینه‌سازی برای چهار تابع Schwefel 2.26 (F08)، Rastrigin (F09)، Ackley (F10) و Griewank (F11) در شکل‌های ۳-ح تا ۳-ذ نشان داده شده است. شکل‌های ۳-ح تا ۳-ذ نشان می‌دهند روش پیشنهادی نه تنها سرعت همگرایی بالاتری نسبت به الگوریتم‌های دیگر مورد بررسی دارد بلکه توانسته به مقادیر بهینه بهتری نیز دست یابد.

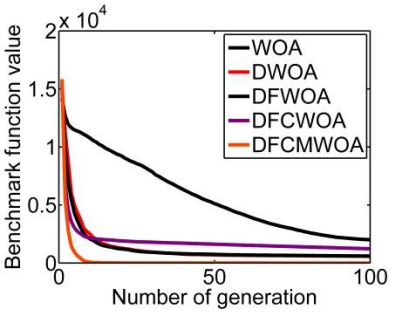
شکل‌های ۳-ر تا ۳-ض فرایند بهینه‌سازی برای شش تابع Generalized Penalized (F12)، Penalty (F13)، De Joung (shek-el's Foxholes) (F14)، Kowalik (F15)، Camel Back -6 Hump (F16) و Branin (F17) را نشان می‌دهد. سرعت همگرایی الگوریتم‌های مورد بررسی برای این توابع تقریباً مشابه هم بوده است.

برای تابع Kowalik (F15)، الگوریتم DFCMWOA الگوریتم برتر در رسیدن به راه‌حل بهینه بوده است این در حالی است که برای سایر توابع روش پیشنهادی بدترین الگوریتم نبوده است.

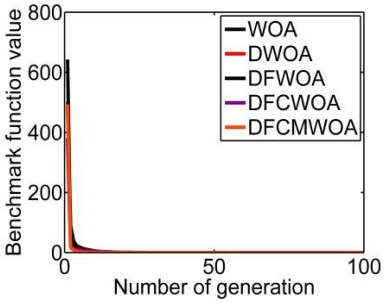
مقادیر بهینه به‌دست‌آمده برای توابع Goldstein and Price (F18)، Hartman 3-Dimentional (F19) و Hartman 6-Dimentional (F20) در شکل‌های ۳-ط تا ۳-ع نشان داده شده است. برای این توابع سرعت همگرایی تقریباً مشابه هم هستند. برای این توابع الگوریتم‌های WOA و DWOA عملکرد بهتری نسبت به دیگر الگوریتم‌های مورد بررسی داشته است.

فرایند بهینه‌سازی برای سه تابع Shekel 1 (F21)، Shekel 2 (F22) و Shekel 2 (F23) در شکل‌های ۳-غ تا ۳-ق نشان داده شده است. شکل‌های ۳-غ تا ۳-ق نشان می‌دهند روش پیشنهادی سرعت همگرایی بالاتری نسبت به الگوریتم‌های دیگر مورد بررسی دارد.

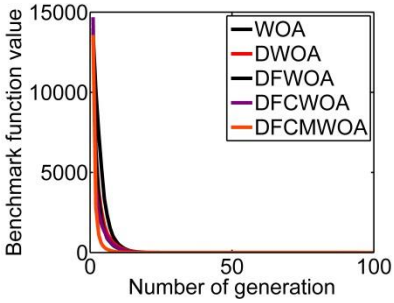
با در نظر گرفتن جدول‌های ۵ و ۶ و شکل ۳ می‌توان نتیجه گرفت که روش پیشنهادی روی اکثر توابع محک نه تنها توانسته است به جواب بهتری دست یابد، بلکه سرعت همگرایی بالاتری نیز دارد.



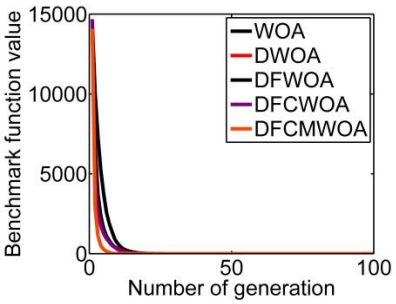
پ) عملکرد روی تابع F03



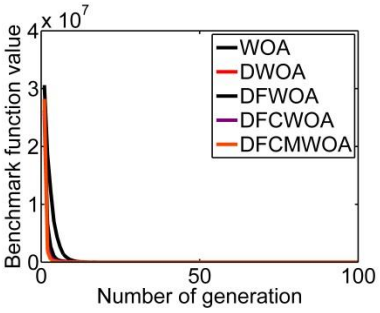
ب) عملکرد روی تابع F02



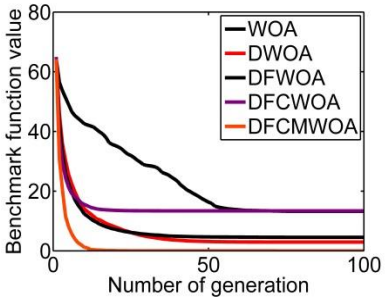
الف) عملکرد روی تابع F01



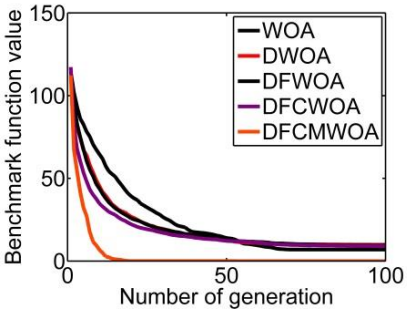
چ) عملکرد روی تابع .F06



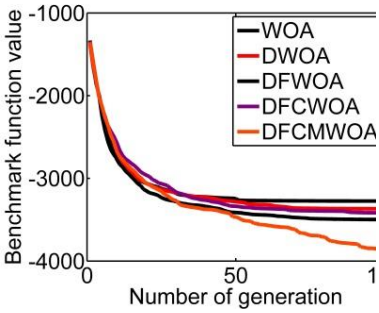
ج) عملکرد روی تابع F05



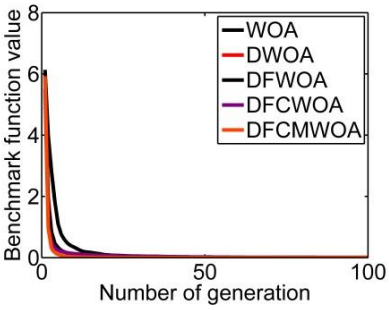
ت) عملکرد روی تابع F04



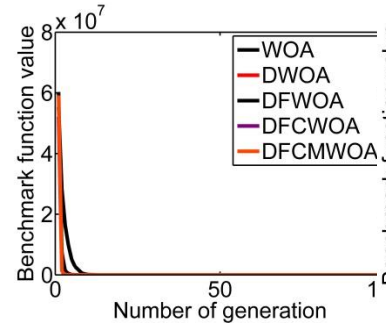
د) عملکرد روی تابع F09



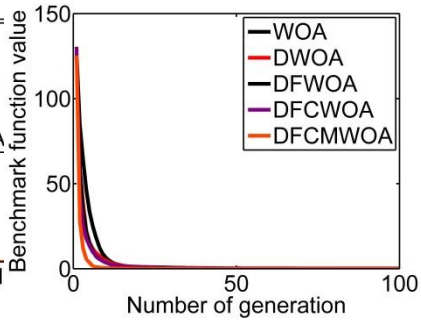
خ) عملکرد روی تابع F08



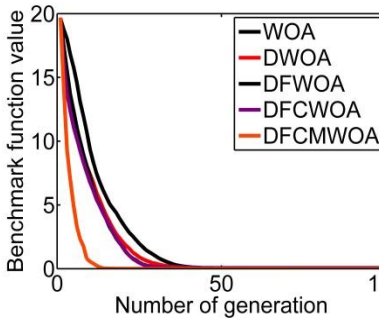
ح) عملکرد روی تابع F07



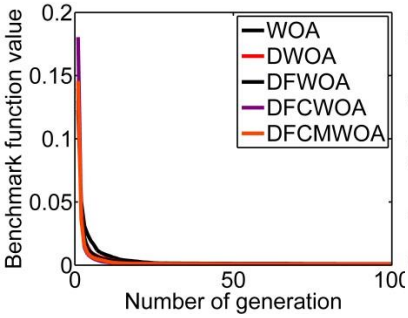
ز) عملکرد روی تابع F12



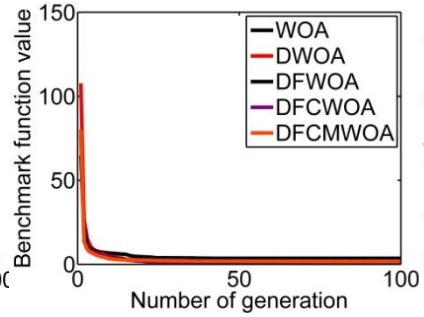
ر) عملکرد روی تابع F11



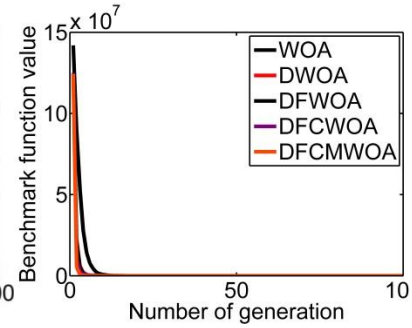
ذ) عملکرد روی تابع F10



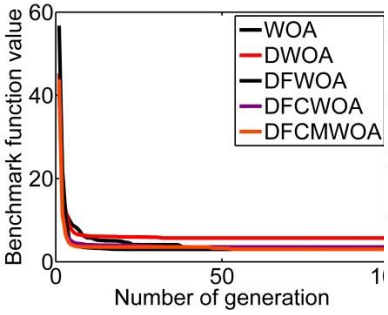
ش) عملکرد روی تابع F15



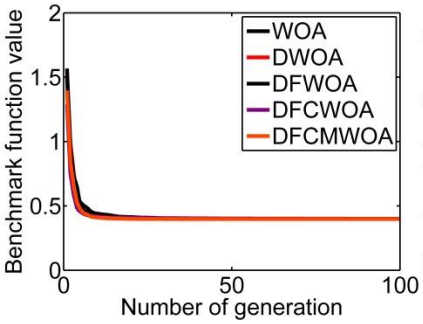
س) عملکرد روی تابع F14



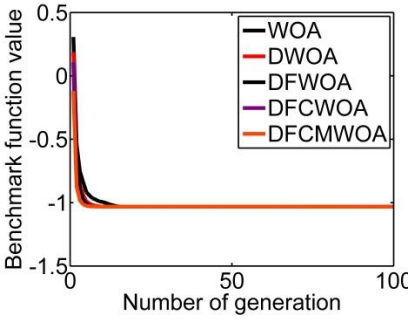
ژ) عملکرد روی تابع F13



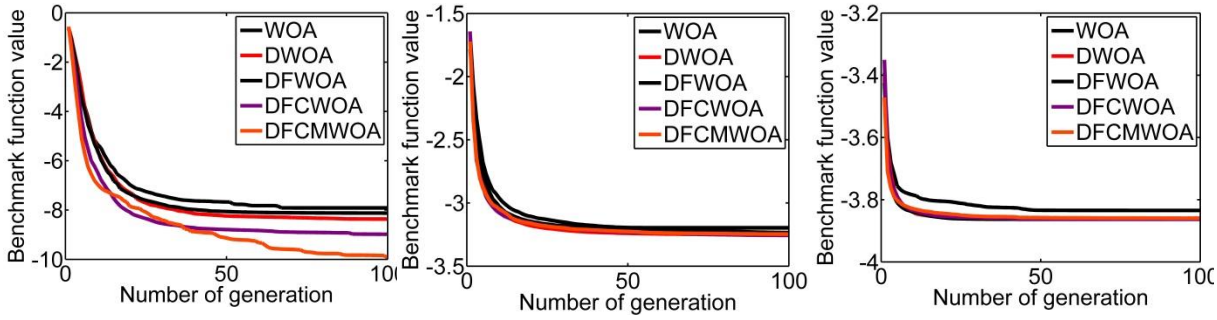
ط) عملکرد روی تابع F18



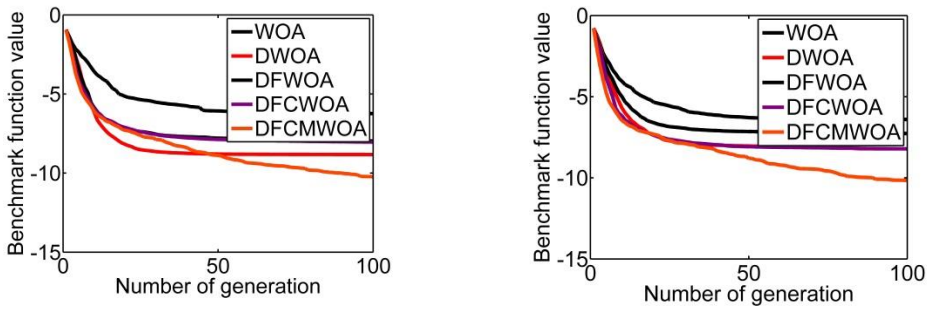
ض) عملکرد روی تابع F17



ص) عملکرد روی تابع F16



ظ) عملکرد روی تابع F19 ع) عملکرد روی تابع F20 غ) عملکرد روی تابع F21



ف) عملکرد روی تابع F22 ق) عملکرد روی تابع F23

شکل ۳: مقایسه عملکرد روش پیشنهادی در مقابل سایر روشها بر روی توابع **F01-F23** به ترتیب از شکل (الف) – (ق)

جدول ۷ متوسط زمان اجرای الگوریتم پیشنهادی در مقایسه با سایر روشها در بهینه‌سازی توابع محک را نشان می‌دهد. این جدول نشان می‌دهد، که علی‌رغم رسیدن به هدف افزایش سرعت همگرایی مطابق آنچه در شکل ۳ نشان داده شد و در مورد آن بحث گردید، زمان اجرای روش پیشنهادی در مقایسه با برخی از روشهای ساده‌تر بیشتر است. این موضوع البته تا حدودی قابل پیش‌بینی بود زیرا روش پیشنهادی در هر تکرار (نسل) از عملگرهای بیشتری استفاده می‌کند. برای مثال عملگرهای جهش و ادغام خود سربار اضافه بر الگوریتم تحمیل می کنند. با این حال این افزایش زمان بهایی است که در ازای افزایش سرعت همگرایی و همگرایی دقیقتر به بهینه سراسری پرداخت شده است. البته بازبینی در پیاده سازی روش و تعداد انجام این عملگرها می تواند این نقیصه را نیز کاهش دهد.

جدول ۷ مقایسه متوسط زمان الگوریتم پیشنهادی با الگوریتم های بهینه سازی دیگر بر روی توابع محک بر حسب ثانیه

تابع	ALO	GWO	MFO	MVO	PSO	SCA	WOA	DFCMWOA
F01	۱,۵۹۳۰۷	۰,۴۷۳۰۲	۰,۴۰۳۷۷	۰,۶۵۹۳۰	**۰,۲۱۷۶۶**	۰,۳۷۱۵۴	۰,۳۲۸۵۸	۱,۳۶۲۲۷
F02	۱,۶۹۵۱۷	۰,۴۳۷۲۵	۰,۴۱۱۴۹	۰,۵۳۷۶۷	**۰,۲۳۹۰۰**	۰,۳۹۶۳۴	۰,۳۴۹۱۱	۱,۳۰۶۶۳
F03	۱,۷۰۳۳۸	۰,۹۱۰۸۹	۰,۸۹۵۶۸	۰,۹۰۱۹۳۷	**۰,۷۵۷۰۶**	۰,۸۶۲۵۶	۰,۸۴۳۸۳	۱,۴۵۵۷۳

F04	۱,۶۲۵۰۰	۰,۳۸۲۹۲	۰,۴۱۵۸۸	۰,۵۰۸۳۶	**۰,۱۹۹۰۵**	۰,۳۴۲۶۸	۰,۳۰۱۲۱	۱,۲۷۷۲۷
F05	۱,۶۳۶۷۵	۰,۴۷۰۳۲	۰,۴۵۳۱۵	۰,۵۹۰۴۹	**۰,۲۸۹۱۱**	۰,۴۴۴۷۱	۰,۴۱۷۹۶	۱,۳۱۰۰۱
F06	۱,۶۲۲۲۳	۰,۳۶۵۰۸	۰,۳۶۴۳۷	۰,۴۹۵۲۴	**۰,۱۹۴۷۳**	۰,۳۳۴۷۰	۰,۳۰۹۳۷	۱,۲۷۰۳۱
F07	۱,۶۳۲۵۰	۰,۵۰۷۰۳	۰,۴۷۷۱۲	۰,۶۱۶۰۲	**۰,۳۲۲۷۶**	۰,۴۵۸۴۲	۰,۴۴۰۵۸	۱,۲۸۷۶۱
F08	۱,۶۲۴۳۶	۰,۴۶۸۰۵	۰,۴۴۸۰۲	۰,۴۹۹۱۸	**۰,۲۸۷۹۴**	۰,۴۳۱۸۷	۰,۴۰۳۷۸	۱,۶۹۴۶۰
F09	۱,۶۲۱۴۲	۰,۳۹۲۵۴	۰,۳۸۵۰۴	۰,۵۴۳۰۴	**۰,۲۱۸۸۰**	۰,۳۶۷۷۳	۰,۳۲۴۴۸	۱,۲۴۰۹۴
F10	۱,۶۲۱۳۳	۰,۴۴۴۴۱	۰,۴۵۶۴۸	۰,۶۰۶۰۶	**۰,۲۷۹۲۵**	۰,۴۳۷۷۴	۰,۳۹۶۲۳	۱,۲۷۲۲۰
F11	۱,۶۵۸۰۱	۰,۵۱۳۴۰	۰,۵۲۷۱۹	۰,۶۵۲۳۹	**۰,۳۶۱۳۳**	۰,۵۱۴۰۲	۰,۴۵۸۴۳	۱,۲۹۸۱۷
F12	۱,۸۰۷۷۸	۰,۳۹۹۳۹	۰,۶۶۲۴۴	۰,۱۶۲۸۰	**۰,۱۰۹۱۱۸**	۰,۱۲۲۱۱	۰,۱۳۱۰۷	۱,۶۲۲۷۸
F13	۲,۲۰۱۳۱	۰,۲۷۳۹۰	۰,۳۰۹۵۰	۰,۱۵۲۰۶	**۰,۱۱۲۰۴۶**	۰,۱۴۲۲۵	۰,۱۲۲۰۳	۲,۲۳۸۶۲
F14	۱,۱۱۲۸۰	۰,۴۲۰۸۳	۰,۴۳۱۶۰	۰,۴۴۷۵۳	**۰,۳۸۸۴۹**	۰,۳۹۸۲۳	۰,۳۹۴۲۰	۳,۳۱۴۲۹
F15	۱,۰۴۸۴۰	۰,۶۱۰۹۵	۰,۴۳۲۰۱	۰,۶۳۱۱۸	**۰,۴۰۱۵۲**	۰,۶۶۶۷۸	۰,۶۸۰۲۴	۱,۷۲۴۸۴
F16	۰,۶۱۴۰۵	۰,۲۶۲۳۷	۰,۳۵۶۳۴	۰,۴۵۰۵۶	**۰,۱۶۷۸۷**	۰,۳۲۶۳۹	۰,۳۳۳۳۴	۳,۱۰۶۴۵
F17	۰,۹۱۰۰۵	۰,۳۱۵۹۶	۰,۳۶۲۵۹	۰,۴۱۵۴۴	**۰,۲۷۰۳**	۰,۲۲۴۲۶	۰,۲۲۱۲۷	۳,۴۴۵۸۷
F18	۰,۵۲۰۶۴	۰,۲۳۰۳۲	۰,۲۷۰۲۵	۰,۳۳۳۷۲	**۰,۱۷۴۱**	۰,۲۳۷۳۳	۰,۲۲۲۹۱	۲,۵۸۳۲۴
F19	۰,۹۹۴۳۳	۰,۵۱۷۱۸	۰,۵۲۴۱۲	۰,۹۵۲۴۴	**۰,۴۰۵۸۵**	۰,۴۹۰۶۱	۰,۴۷۶۴۳	۲,۴۱۳۳۰
F20	۱,۷۲۱۲۱	۰,۴۳۱۶۵	۰,۴۶۸۴۱	۰,۵۳۹۱۸	**۰,۲۱۳۱**	۰,۴۳۹۴۸	۰,۵۴۸۴	۱,۴۵۱۶۵
F21	۱,۱۵۱۶۵	۰,۱۱۶۵۴	۰,۱۹۱۱۸	۰,۱۰۲۰۳۴	**۰,۱۰۵۴۴**	۰,۹۸۵۱۶	۰,۸۴۵۱	۱,۲۸۶۴
F22	۲,۲۱۳۱۶	۰,۳۶۵۴۶	۰,۳۵۷۲۱	۰,۲۳۲۶۸	**۰,۰۹۸۷**	۰,۱۵۱۶۶	۰,۱۱۱۵	۲,۲۱۳۵۱
F23	۱,۶۱۳۲۱	۰,۸۲۳۲	۰,۷۶۵۰۶	۰,۸۹۰۲۹	**۰,۴۵۱۶۵**	۰,۶۵۴۶	۰,۵۱۵۱	۱,۲۴۶۱۲
میانگین	۱,۴۷۵	۰,۴۴	۰,۴۵	۰,۵۱	**۰,۲۷**	۰,۴۲۶	۰,۳۹۸	۱,۸

۷- نتیجه‌گیری

همان‌طور که اشـاره شـد فرایند جسـتجو در الگوریتم‌های فرااکتشـافی را می‌توان به دو فاز اکتشاف و بهره‌برداری تقسیم کرد. اجرای متعادل این دو فاز در الگوریتم‌های فرااکتشـافی باعث بهبود عملکرد الگوریتم می‌شـود. در این پژوهش به منظور بهبود جسـتجوی الگوریتم WOA، فاز اکتشـاف و بهره‌برداری این الگوریتم را به‌طور ایسـتا و پویا بهبود دادیم. به منظور بهبود ایستا

فاز اکتشـــاف و بهره‌برداری این الگوریتم، یک روش ترکیبی جدید با اســـتفاده از اصـــل تکاملی داروین و عملگرهای ادغام و جهش الگوریتم ژنتیـک ارائه کردیم. در این روش ترکیبی از عملگر ادغام و جهش به ترتیب به منظور بهبود فاز بهره‌برداری و اکتشـاف الگوریتم اسـتفاده شـده اسـت. عملگر جهش با فراهم کردن شـرایط تصـادفی بیشتر برای الگوریتم، باعث بهبود فاز اکتشـــاف الگوریتم می‌شـــود. عملگر ادغام با فراهم کردن ادغام بهترین موقعیت با دیگر عناصـــر جســـتجو باعث بهبود فاز بهره‌برداری الگوریتم شـده اسـت. همچنین به‌منظور بهبود پویای این فازها از یک FIS اسـتفاده شده است. استفاده از FIS در روش پیشنهادی باعث می‌شود فازهای اکتشاف و بهره‌برداری در هر بار اجرای الگوریتم و به صورت پویا تنظیم شوند که این کار باعث تعادل بهتر این دو فاز شده و در نتیجه باعث بهبود فرایند جستجوی الگوریتم می‌شود.

به منظور ارزیابی روش پیشـنهادی از بیسـت و سـه تابع محک اسـتفاده شـده اسـت. در ابتدا عملکرد روش پیشنهادی با ترکیب‌های تکاملی دیگر مورد مقایسه قرار گرفت که نتایج به‌دست‌آمده نشان داد روش پیشنهادی بهترین روش ترکیبی است. سـپس روش پیشـنهادی با هفت الگوریتم بهینه‌سـازی مورد مقایسـه قرار گرفت. نتایج به‌دست‌آمده نشان داد روش پیشنهادی عملکرد بهتری در رسیدن به راه‌حل بهینه بهتر دارد. همچنین بررسی فرایند بهینه‌سازی در الگوریتم‌های مورد بررسی نشان داد روش پیشنهادی سرعت همگرایی بالاتری نیز دارد. بنابراین می‌توان نتیجه گرفت که روش پیشنهادی عملکرد بهتری نسبت به الگوریتم WOA استاندارد و جدیدترین روش‌های بهینه‌سازی مورد بررسی دارد.

از آنجا که علی‌رغم افزایش سـرعت همگرایی الگوریتم به بهینه سـراسـری و رسـیدن به آن در تعداد نسلهای کمتر، زمان اجرای الگوریتم به طور متوسـط از سـایر روشها بالاتر است، یکی از بهبودهایی که می توان در این الگوریتم ایجاد کرد، تغییر سـاختار و بهینه‌سـازی اسـتفاده از عملگرهای آن است. یکی دیگر از کارهای آینده نیز طبیعتا استفاده از رویکرد پیشنهادی در یک مساله دنیای واقعی خواهد بود.

مراجع

[1] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*: U Michigan Press, 1975.

[2] K. James and E. Russell, "Particle swarm optimization," in *Proceedings of 1995 IEEE International Conference on Neural Networks*, 1995, pp. 1942-1948.

[3] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, pp. 28-39, 2006.

[4] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of global optimization*, vol. 39, pp. 459-471, 2007.

[5] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, pp. 78-84, 2010.

[6] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information sciences*, vol. 179, pp. 2232-2248, 2009.

[7] D. Simon, "Biogeography-based optimization," *IEEE transactions on evolutionary computation,* vol. 12, pp. 702-713, 2008.

[8] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, 2009, pp. 210-214.

[9] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature inspired cooperative strategies for optimization (NICSO 2010)*, ed: Springer, 2010, pp. 65-74.

[10] S. Mirjalili, "The ant lion optimizer," *Advances in Engineering Software,* vol. 83, pp. 80-98, 2015.

[11] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software,* vol. 69, pp. 46-61, 2014.

[12] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-verse optimizer: a nature-inspired algorithm for global optimization," *Neural Computing and Applications,* vol. 27, pp. 495-513, 2016.

[13] S. Mirjalili, "SCA: a sine cosine algorithm for solving optimization problems," *Knowledge-Based Systems,* vol. 96, pp. 120-133, 2016.

[14] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowledge-Based Systems,* vol. 89, pp. 228-249, 2015.

[15] S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm," *Advances in Engineering Software,* vol. 95, pp. 51-67, 2016.

[16] O. Olorunda and A. P. Engelbrecht, "Measuring exploration/exploitation in particle swarms using swarm diversity," in *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, 2008, pp. 1128-1134.

[17] E. Alba and B. Dorronsoro, "The exploration/exploitation tradeoff in dynamic cellular genetic algorithms," *IEEE Transactions on Evolutionary Computation,* vol. 9, pp. 126-142, 2005.

[18] N. Mittal, U. Singh, and B. S. Sohi, "Modified grey wolf optimizer for global engineering optimization," *Applied Computational Intelligence and Soft Computing,* vol. 2016, p. 8, 2016.

[19] L. Wang, D.-Z. Zheng, and Q. Lin, "Survey on chaotic optimization methods," *Computing Technology and Automation,* vol. 20, pp. 1-5, 2001.

[20] N. Higashi and H. Iba, "Particle swarm optimization with Gaussian mutation," in *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, 2003, pp. 72-79.

[21] G.-G. Wang, L. Guo, A. H. Gandomi, A. H. Alavi, and H. Duan, "Simulated annealing-based krill herd algorithm for global optimization," in *Abstract and Applied Analysis*, 2013.

[22] J. Tillett, T. Rao, F. Sahin, and R. Rao, "Darwinian particle swarm optimization," 2005.

[23] S.-M. Guo and C.-C. Yang, "Enhancing differential evolution utilizing eigenvector-based crossover operator," *IEEE Transactions on Evolutionary Computation,* vol. 19, pp. 31-49, 2015.

[24] X. Yan, Y. Zhu, H. Chen, and H. Zhang, "A novel hybrid artificial bee colony algorithm with crossover operator for numerical optimization," *Natural Computing,* vol. 14, pp. 169-184, 2015.

[25] S. Kumar, V. K. Sharma, and R. Kumari, "A novel hybrid crossover based artificial bee colony algorithm for optimization problem," *arXiv preprint arXiv:1407.5574,* 2014.

[26] S. Chakraborty, A. K. Saha, R. Chakraborty, M Saha, An enhanced whale optimization algorithm for large scale optimization problems, Knowledge-Based Systems, vol. 233, 2021,107543, https://doi.org/10.1016/j.knosys.2021.107543.

[27] M. H. Nadimi-Shahraki, H. Zamani, Seyedali Mirjalili, Enhanced whale optimization algorithm for medical feature selection: A COVID-19 case study, Computers in Biology and Medicine, vol. 148, 2022, 105858, ISSN 0010-4825, https://doi.org/10.1016/j.compbiomed.2022.105858.

[28] X. Lin, X. Yu, W. Li, A heuristic whale optimization algorithm with niching strategy for global multi-dimensional engineering optimization, Computers & Industrial Engineering, vol. 171, 2022, 108361, https://doi.org/10.1016/j.cie.2022.108361.

[29] K. Reddy, A. K. Saha, A modified Whale Optimization Algorithm for exploitation capability and stability enhancement, Heliyon, 2022, e11027, https://doi.org/10.1016/j.heliyon.2022.e11027.

[30] K. Kamal A. Ghany, A.M. AbdelAziz, T. H. A. Soliman, A. Sewisy, A hybrid modified step Whale Optimization Algorithm with Tabu Search for data clustering, Journal of King Saud University - Computer and Information Sciences, vol. 34, Issue 3, 2022, Pages 832-839, https://doi.org/10.1016/j.jksuci.2020.01.015.

[31] W-H. Tan, J. Mohamad-Saleh, A hybrid whale optimization algorithm based on equilibrium concept, Alexandria Engineering Journal, Volume 68, 2023, Pages 763-786, https://doi.org/10.1016/j.aej.2022.12.019.

[32] A. Bhattacharya, B. Saha, S. Chattopadhyay, R. Sarkar, Deep feature selection using adaptive β-Hill Climbing aided whale optimization algorithm for lung and colon cancer detection, Biomedical Signal Processing and Control, Volume 83, 2023, 104692,
https://doi.org/10.1016/j.bspc.2023.104692.

[33] F. Zhao, Z. Xu, H. Bao, T. Xu, N. Zhu, A cooperative whale optimization algorithm for energy-efficient scheduling of the distributed blocking flow-shop with sequence-dependent setup time, Computers & Industrial Engineering, 2023, 109082, https://doi.org/10.1016/j.cie.2023.109082.

[34] H. Xian, J. Che, Unified whale optimization algorithm based multi-kernel SVR ensemble learning for wind speed forecasting, Applied Soft Computing, Volume 130, November 2022, 109690.

[35] Z. Fan, J. Gou, Predicting body fat using a novel fuzzy-weighted approach optimized by the whale optimization algorithm, Expert Systems with Applications, Volume 217, 2023, 119558, https://doi.org/10.1016/j.eswa.2023.119558.

[36] L.-X. Wang, A course in fuzzy systems: Prentice-Hall press, USA, 1999.

[37] L. A. Zadeh, "Fuzzy sets," Information and control, vol. 8, pp. 338-353, 1965.

[38] E. Cox, M. O'Hagan, R. Taber, and M. O'Hagen, The Fuzzy Systems Handkbook with Cdrom: Academic Press, Inc., 1998.

[39] S. Sumathi, P. Surekha, and P. Surekha, Computational intelligence paradigms: theory and applications using MATLAB vol. 1: CRC Press Boca Raton, FL, USA: 2010.

[40] W. Khatib and P. J. Fleming, "The stud GA: a mini revolution?" in International *Conference on Parallel Problem Solving from Nature*, 1998, pp. 683-691.