3rd International Conference on Soft Computing
• 20 & 21 Nov. 2019
• Faculty of Technology and Engineering- East of Guilan
Intelligent Systems Scientific Society of Iran
University of Guilan
سومین کنفرانس بین المللی
• ۲۹ و ۳۰ آبان ۱۳۹۸
• دانشکده فنی و مهندسی شرق گیلان

الگوریتم جستجوی پیمایش معکوس بهبود یافته با استفاده از جهش‌های جدید تطبیقی

اسماعیل میرکازهی ریگی [1*]، امین راحتی [2]

[1] دانشجوی کارشناسی ارشد علوم کامپیوتر، دانشگاه سیستان و بلوچستان؛ esmaeil.mirkazehi@pgs.usb.ac.ir
[2] استادیار گروه علوم کامپیوتر، دانشگاه سیستان و بلوچستان؛ a.rahati@usb.ac.ir

چکیده

الگوریتم جستجوی پیمایش معکوس (BSA) یکی از الگوریتم‌های تکاملی نوین است که به صورت موفق برای حل انواع مسائل بهینه‌سازی دنیای واقعی کاربرد دارد. با این حال، الگوریتم مذکور از سرعت همگرایی کند رنج می‌برد. لذا، در این مقاله یک نسخه بهبود یافته از BSA به نام IBSA معرفی می‌شود. الگوریتم IBSA دارای دو جهش با ویژگی‌های متفاوت است که با هدف اکتشاف بیشتر در مراحل اولیه فرآیند تکامل و بهره‌برداری بیشتر در مراحل بعدی، در هر نسل به صورت تطبیقی، یکی از دو جهش را استفاده می‌کند. همچنین دارای روش جدید تطبیقی پویا جهت تنظیم اندازه گام جستجوی مسیر می باشد که به توازن بین اکتشاف و بهره‌برداری کمک می‌کند. جهت بررسی کارایی الگوریتم IBSA، از توابع محک CEC 2019 استفاده شده است. نتایج آزمایشات برتری روش پیشنهادی را برحسب معیارهای دقت، پایداری و سرعت همگرایی در مقایسه با الگوریتم‌های فرااابتکاری دیگر نشان می‌دهد.

کلمات کلیدی

الگوریتم‌های تکاملی؛ جستجوی پیمایش معکوس؛ CEC 2019

۱- مقدمه

درسال‌های اخیر، الگوریتم‌های تکاملی[1] (EA) نظر بسیاری از محققان را برای حل انواع مسائل بهینه‌سازی عددی به‌خود جلب کرده‌اند. قابل ذکر است که اکثر EAها دارای عملگرهای پایه‌ای شامل جهش، برش و انتخاب هستند. با این حال، تفاوت اصلی آن‌ها در نحوه تولید فرزندان می‌باشد [۱]. از معروف‌ترین و پرکاربردترین EAها می توان به الگوریتم ژنتیک[2] (GA) [۲]، الگوریتم تکامل تفاضلی[3] (DE) [۳]، الگوریتم استراتژی تکامل انطباق ماتریس کوواریانس[4] (CMA-ES) [۴]، برنامه‌نویسی تکاملی[5] (EP) [۵] و الگوریتم جستجوی پیمایش معکوس[6] (BSA) [۶] اشاره نمود. BSA، یک الگوریتم مبتنی بر جمعیت با تکنیک جستجوی تصادفی است که اولین بار توسط سیویکیگلو[7] در سال ۲۰۱۳ معرفی شده است. دو تفاوت اصلی BSA نسبت به دیگر EAها، استفاده از اطلاعات مفید تاریخی در معادلات جهش و استفاده از برش غیریکنواخت برای تولید جمیعت آزمایشی نهایی است.

علی رغم اینکه الگوریتم BSA یک الگوریتم نسبتا جدید است اما به دلیل پایداری، ساختار نسبتا ساده، تعداد مولفه‌های کنترلی کم و عدم حساسیت به مقدار آن‌ها برای حل انواع مسائل بهینه‌سازی چندهدفه [۷]، مقید [۸-۱۰]، گسسته [۱۱] و مسائل پویا[۱۰] بکار رفته است. همچنین برای انواع مسائل بهینه‌سازی دنیای واقعی از قبیل توزیع سهام اقتصادی [۱۲]، توزیع اقتصادی غیرپوشا [۱۳]، برنامه کنترل کننده بهینه بهنگام برای سیستم مدیریت انرژی خانگی [۱۴]، پخش بار بهینه سیستم‌های جریان مستقیم-ولتاژ بالا[8] دو پایانه‌ای [۱۵] و واحد پردازش گرافیک [۱۶] به‌صورت موفق اعمال شده است. با این حال، الگوریتم BSA ازسرعت همگرایی کند و گرفتار شدن در دام بهینه محلی رنج می‌برد [۱۷, ۱۸]. بنابراین، برای بهبود

[1] Evolutionary algorithm
[2] Genetic algorithm
[3] Differential evolution
[4] Covariance matrix adaptation evolution strategy
[5] Evolutionary programming
[6] Backtracking search optimization
[7] Pinar Civicioglu
[8] High-voltage direct current

کارایی کلی الگوریتم BSA انواع نسخه‌های متفاوت توسط محققان معرفی شده‌اند که به سه دسته کلی تقسیم می‌شوند: دسته اول مبتنی بر تنظیم مولفه‌ها از قبیل ضریب مقیاس و نرخ درهم کردن است که به‌ترتیب در معادلات جهش و برش استفاده می‌شوند [۱۸, ۱۹]، دسته دوم مبتنی بر افزودن عملگرهای جدید از قبیل استراتژی‌های یادگیری جدید و یا مکانیزم های مرسوم است [۱۷, ۲۰-۲۲] و دسته سوم مبتنی بر ترکیب با هریک از تکنیک‌ها یا الگوریتم‌های EA دیگر است [۱۰, ۱۶, ۱۸, ۲۳].

قابل ذکر است که یکی از دلایل اصلی همگرایی کند الگوریتم‌های EA، عدم توازن بین اکتشاف و بهره‌برداری است [۲۴] که الگوریتم BSA از این قاعده مستثنی نیست، زیرا دارای قدرت اکتشاف بالا و بهره‌برداری ضعیف است [۲۰]. یکی از ساده‌ترین روش‌ها، برای توازن بین اکتشاف و بهره‌برداری الگوریتم‌های EA این است که در مراحل اولیه فرآیند تکامل، الگوریتم مورد نظر باید دارای قدرت اکتشاف بالا باشد تا بتواند کل فضای جستجو را به‌صورت موثر کاوش کند و در تکرارهای آخر دارای قدرت بهره‌برداری بالایی باشد تا بتواند به جواب بهینه سراسری همگرا شود [۲۵]. علاوه براین، بعضی از استراتژی‌های یادگیری که تاکنون در معادلات جهش الگوریتم BSA اعمال شده‌اند دارای قدرت اکتشاف بالا و بعضی دیگر دارای قدرت بهره‌برداری بالا هستند. با این حال، مکانیزم اکثر معادلات جهش اصلاح شده، افزودن بردارهای تفاضلی به معادلات جهش است و تاکنون تغییرات کمی روی بردار پایه انجام شده‌است.

در این پژوهش، با توجه به انگیزش‌های فوق، یک نسخه بهبود یافته از BSA به‌نام IBSA پیشنهاد می‌شود. IBSA دارای دو جهش متفاوت است که هر کدام دارای ترکیب خطی جدیدی از بردارهای پایه هستند. از این‌رو، در مراحل اولیه فرآیند تکامل از جهش اول بدلیل قدرت اکتشاف بالا و در مراحل پایانی از جهش دیگر که دارای قدرت بهره‌برداری بالا است استفاده می‌شود. علاوه براین، بدلیل اینکه ضریب مقیاس به توازن بین اکتشاف و بهره‌برداری کمک می‌کند در IBSA فرمول بروزرسانی جدیدی نیز بدین منظور معرفی می‌شود. ساختار مقاله در ادامه به صورت زیر سازماندهی می‌شود: در بخش ۲، الگوریتم BSA پایه ای به صورت خلاصه معرفی می‌شود. روش پیشنهادی IBSA در بخش ۳ شرح داده می‌شود. در بخش ۴، الگوریتم IBSA با استفاده از توابع محک CEC 2019[9] ارزیابی و با الگوریتم‌های دیگر مقایسه شده و در بخش ۵ نتیجه‌گیری و جمع‌بندی پژوهش ارائه می‌شود.

۲- الگوریتم BSA پایه ای

BSA یک الگوریتم EA مبتنی بر جمعیت است با این تفاوت که دارای یک حافظه است که در آن اطلاعات مفید نسل گذشته به صورت تصادفی ذخیره می‌شوند. این اطلاعات نقش بسیار مهمی درتولید ماتریس جستجوی مسیر دارد که برای بروزرسانی اعضا استفاده می‌شود. BSA تنها دارای یک مولفه کنترلی به‌نام mixrate است که برای تولید جمعیت آزمایشی نهایی استفاده می‌شود. این مولفه مشخص کننده تعداد عناصر اعضا است که قرار است جهش داده شوند. BSA دارای پنج مرحله اصلی است که به اختصار در زیر شرح داده می‌شوند:

۱) مقدار دهی اولیه: در این مرحله جمعیت اصلی (P) و جمعیت گذشته (hP) براساس توزیع یکنواخت در محدوده فضای جستجوی مورد نظر به‌ترتیب براساس فرمول‌های ۱ و ۲ تولید می‌شوند.

$$P_{i,j} = low_j + rand.(up_j - low_j) \quad (۱)$$

$$hP_{i,j} = low_j + rand.(up_j - low_j) \quad (۲)$$

زیرنویس‌های i و j به‌ترتیب در بازه های [1, NP] و [1, D] انتخاب می‌شوند که NP مشخص‌کننده تعداد اعضای جمعیت و D تعداد ابعاد مسئله مورد نظر را مشخص می‌کند. rand یک عدد تصادفی با توزیع یکنواخت در [0, 1] است. low و up به‌ترتیب کران‌های پایین و بالای مسئله مورد نظر هستند.

۲) انتخاب -I : در ابتدای هر نسل hP ابتدا به‌صورت تصادفی با استفاده از P بروز رسانی می‌شود، سپس ترتیب اعضا با جایگشتی تصادفی تغییر می کند. به‌ترتیب، فرمول‌های ۳ و ۴ این عملیات را نشان می‌دهند.

$$hP = \begin{cases} P, & if\ r_1 < r_2 \\ hP, & otherwise \end{cases} \quad (۳)$$

$$hP = permuting(hP) \quad (۴)$$

r_1 و r_2 دو عدد تصادفی در [0, 1] هستند، همچنین permuting یک تابع جایگشت تصادفی است.

۳) جهش: در این مرحله جمعیت آزمایشی اولیه (M) براساس جمعیت‌های اصلی و تاریخی تولید می‌شود، معادله ۵ این فرآیند را نشان می‌دهد:

[9] 2019 IEEE Congress on Evolutionary Computation

$$M = P + F.(hP - P) \quad (۵)$$

با توجه به معادله جهش مذکور، مشاهده می‌شود ماتریس جستجوی مسیر براساس تفاوت بین جمعیت تاریخی و اصلی تولید می‌شود. از این‌رو، جمعیت تاریخی کل فرآیند تکامل را تحت تاثیر قرار می‌دهد. ضریب مقیاس F کنترل کننده دامنه ماتریس جستجوی مسیر است که براساس توزیع نرمال با میانگین صفر و انحراف از معیار ۳ تولید می‌شود، یعنی داریم: F=N(0,3).

۴) برش: در این مرحله شکل نهایی جمعیت آزمایشی (T) از ترکیب گسسته بین جمعیت اصلی (P) و جمعیت اولیه آزمایشی (M) براساس فرمول ۶ ایجاد می‌شود.

$$T_{i,j} = \begin{cases} P_{i,j}, & if\ map_{i,j} = 1 \\ M_{i,j}, & otherwise \end{cases} \quad (۶)$$

ماتریس map یک ماتریس با مرتبه NP×D است که مقدار اولیه آن ۱ است و فرآیند بروزرسانی آن براساس فرمول ۷ انجام می‌شود.

$$map_{i,u} = 0 \mid u = \begin{cases} combining\ (D, \lceil D * rand * mixrate \rceil), & if\ r_3 < r_4 \\ randi(D), & otherwise \end{cases} \quad (۷)$$

زیرنویس u مشخص‌کننده عناصر تصادفی است که قرار است صفر شوند. r_3 و r_4 دو عدد تصادفی در بازه [0, 1] هستند که در هر نسل برای همه اعضا یکسان در نظر گرفته می‌شوند. mixrate نرخ درهم کردن است و در اینجا برابر با ۱ است، $\lceil\ \rceil$ تابع سقف است. combining تابع ترکیب تصادفی است که به صورت تصادفی از فضای D بعدی تعداد $\lceil D * rand * mixrate \rceil$ عنصر را انتخاب می‌کند و randi یک عدد صحیح تصادفی را تولید می‌کند.

بعد از تولید جمعیت آزمایشی نهایی (T)، عناصر اعضایی که از حدود مجاز مسئله تخطی کرده‌اند، با استفاده از مکانیزم زیر دوباره به‌صورت تصادفی در فضای جستجوی مورد نظر تولید می‌شوند:

$$T_{i,j} = \begin{cases} low_j + rand.(up_j - low_j), & if\ T_{i,j} < low_j\ \vee\ T_{i,j} > up_j \\ T_{i,j}, & otherwise \end{cases} \quad (۸)$$

۵) انتخاب -II : در این مرحله جمعیت اصلی برای نسل بعد بروزرسانی می‌شود که این عملیات براساس انتخاب حریصانه بین اعضای متناظر جمعیت اصلی فعلی و جمعیت آزمایشی صورت می‌گیرد. معادله ۹ این رویه را نشان می‌دهد:

$$P_i = \begin{cases} T_i, & if\ f(T_i) < f(P_i) \\ P_i, & otherwise \end{cases} \quad (۹)$$

f نشان‌دهنده تابع برازندگی برای مسائل کمینه‌سازی است.

۳- IBSA پیشنهادی

۳. ۱- انگیزش

با توجه به آنچه که در بخش قبل ذکر شد، مشاهده می‌شود BSA دارای ساختار نسبتا ساده‌ای است که از اطلاعات تاریخی برای بهبود کارایی خود استفاده می‌کند. BSA با توجه به معادله ۵، دامنه وسیعی از فضای جستجو را می‌تواند کاوش کند، بنابراین، از اکتشاف خوبی برخوردار است. با این حال، BSA بدلیل عدم استفاده راهنمایی توسط اعضای بهتر از ضعف در بهره‌برداری رنج می‌برد که این پدیده منجر به کندی سرعت همگرایی می‌شود. مشکل کندی سرعت همگرایی با توازن بین اکتشاف و بهره‌برداری حل می‌شود. علاوه براین، BSA برای توازن بین اکتشاف و بهره‌برداری بایستی در اوایل فرآیند تکامل، دارای قدرت اکتشاف بیشتری نسبت به بهره‌برداری باشد و در اواخر فرآیند تکامل بالعکس، از قدرت بهره‌برداری بیشتری نسبت به اکتشاف برخوردار باشد. همچنین، از دیگر عواملی که توازن بین اکتشاف و بهره‌برداری الگوریتم BSA را تحت تاثیر قرار می‌دهد، ضریب مقیاس F است. از این‌رو، باید نسخه‌ای از الگوریتم BSA معرفی شود که از پیچیدگی کمی برای توازن بین اکتشاف و بهره‌برداری برخوردار باشد. بدین منظور، در ادامه یک نسخه بهبود یافته از BSA به‌نام IBSA معرفی می‌شود.

IBSA کارایی الگوریتم BSA را در عین سادگی بهبود می‌دهد، تنها تفاوت بین BSA و الگوریتم پیشنهادی IBSA در مرحله جهش است که دو جنبه بهبود در آن موجود است: اولا، استفاده از دو نوع روش جهش تطبیقی مبتنی بر بردار پایه، بدین‌صورت که الگوریتم در مراحل اولیه تکامل از اکتشاف خوبی برخوردار باشد و رفته رفته در مراحل پایانی دارای بهره‌برداری بالایی باشد تا به یک جواب بهینه همگرا شود. ثانیا، از یک رویکرد تنظیم تطبیقی پویا برای تعیین ضریب مقیاس F استفاده می‌شود که توازن بین اکتشاف و بهره‌برداری را تحت تاثیر قرار می‌دهد.

۳. ۲- جهش IBSA

مطالعاتی که تاکنون روی معادله جهش BSA انجام شده است شامل افزودن بردارهای تفاضلی است [۱۶-۱۸, ۲۰] و تغییرات کمی روی اصلاح بردار پایه صورت گرفته است [۲۶]. لذا، سعی شده است تا در IBSA انتخاب بردار پایه بگونه ای متفاوت از سایر نسخه‌های بهبودیافته BSA باشد. همچنین، قابل ذکر است که در الگوریتم IBSA ماتریس جستجوی مسیر بدون تغییر باقی می‌ماند و هدف، تولید جواب‌های با کیفیت و کارا با سرعت همگرایی معقول است. از این‌رو، جهش موجود در IBSA در هر نسل یکی از دو رفتار اکتشاف یا بهره برداری را با یک احتمال خطی که در طی نسل‌ها کاهش می‌یابد انتخاب می‌کند. بنابراین، در مراحل اولیه فرآیند تکامل، رویه‌ای که دارای اکتشاف است شانس بیشتری برای انتخاب شدن دارد. همانگونه که فرمول ۱۰ نشان می دهد این رویه ترکیب خطی از بردار پایه متناظر هر عضو و بردار تصادفی انتخاب شده از جمعیت است. سپس به تدریج در طی فرآیند تکامل رویه‌ای که دارای بهره‌برداری است شانس بیشتر برای انتخاب شدن دارد که این رویه براساس فرمول ۱۱ ترکیب خطی از بردار پایه متناظر هر عضو و بهترین عضو جمعیت است.

$$M_i = w_1 P_i + w_2 P_r + F_i.(hP_i - P_i) \quad (۱۰)$$

$$M_i = w_1 P_i + w_2 P_{best} + F_i.(hP_i - P_i) \quad (۱۱)$$

i شمارنده اعضای جمعیت است که i=1, 2, …, NP. زیرنویس r یک عدد تصادفی صحیح در [1, NP] است، بطوریکه: $r \neq i$. P_{best} بهترین عضو جمعیت در نسل فعلی است، w_1 و w_2 دو عدد تصادفی در [0, 1] به‌طوری که: $w_1 + w_2 = 1$. F_i با استفاده از یک برنامه تطبیقی پویا بروزرسانی می‌شود.

فرمول بروزرسانی ضریب مقیاس F در الگوریتم پیشنهادی، شبیه BSA استاندارد و براساس توزیع نرمال است، با این تفاوت که میانگین و انحراف از معیار در IBSA به‌صورت تطبیقی تنظیم می‌شوند. میانگین در طی نسل‌ها کاهش می‌یابد و انحراف از معیار براساس نرخ شکست (جواب‌های تولیدی آزمایشی بد) تنظیم می‌شود. اگر نرخ شکست بالا باشد، در این حالت ممکن است که الگوریتم در دام بهینه محلی گرفتار شده باشد بنابراین، جهت اکتشاف نقاط دیگر و پرش از آن، مقدار F افزایش، در غیر اینصورت با هدف بهره‌برداری بیشتر و همگرا شدن به جواب بهینه مقدار آن کاهش می‌یابد. فرمول‌های ۱۲-۱۴ این فرآیند را نشان می‌دهند.

$$\mu_F = F_{max} - (F_{max} - F_{min}) \times \frac{nfe}{Maxnfe} \quad (۱۲)$$

$$\sigma_F = \frac{\sum_{i=1}^{NP} fr_i}{NP} \mid fr_i = \begin{cases} 1, & if\ f(T_i) > f(P_i) \\ 0, & otherwise \end{cases} \quad (۱۳)$$

$$F_i = N(\mu_F, \sigma_F) \quad (۱۴)$$

F_{max} و F_{min} به‌ترتیب بیشترین و کمترین مقدار ضریب مقیاس F هستند. nfe تعداد ارزیابی‌های تابعی در حال حاضر و Maxnfe بیشینه تعداد ارزیابی‌های تابعی است.

شکل ۱ شبه کد الگوریتم IBSA را نشان می‌دهد.

۴- نتایج آزمایشات

در این بخش، جهت نشان دادن دقت و کارایی الگوریتم IBSA توابع محک CEC 2019 با ۱۰۰ رقم چالش [۲۷] و همچنین چندین الگوریتم فراابتکاری موفق برای مقایسه استفاده می‌شوند. بدین منظور در زیربخش ۴-۱ ابتدا توابع محک CEC 2019 و تنظیمات مولفه‌های الگوریتم‌های مقایسه شونده بیان و سپس نتایج آماری در زیربخش ۴-۲ ارائه می شوند.

۴. ۱- توابع محک CEC 2019 و تنظیمات مولفه‌ها

CEC 2019 از ۱۰ تابع محک چندوجهی تک هدفه با کران‌های مقید تشکیل شده است. هدف در این نوع توابع رسیدن به جواب بهینه با دقت ۱۰ رقم اعشار است. از این‌رو به این دنباله توابع چالش ۱۰۰ رقمی گویند. جدول ۱ نام هر تابع، تعداد ابعاد فضای جستجو و مقدار بهینه هر تابع را نشان می‌دهد. قابل ذکر است در [۲۷] هیچ شرایط خاتمه‌ای ارائه نشده است و این‌کار به عهده محقق واگذار شده است. از این‌رو، در پژوهش حاضر، شرط خاتمه براساس رسیدن به خطای کمتر از 10^{-9} یا رسیدن به بیشترین تعداد ارزیابی تابعی 5×10^5 است. یکی از مهم‌ترین معیارهای ارزیابی، محاسبه امتیاز کلی است که یک الگوریتم کسب می‌کند. امتیاز کلی برابر است با مجموع امتیازات تمام توابع. برای این منظور، برای هر تابع، تعداد

ارقام صحیح حدس زده شده روی ۲۵ تا از بهترین اجراها در مجموع ۵۰ بار اجرای مستقل شمارش و میانگین آن‌ها به عنوان امتیاز آن تابع محاسبه می‌شود که در این صورت عالی ترین امتیازی که یک الگوریتم برای هر تابع می‌تواند کسب نماید ۱۰ خواهد بود.

1: Initialize NP, Dim, Maxnfe, F_{max}, F_{min}, mixrate
% Initialize Population
2: Initialize target population P and historical population hP using Eqs. (1), (2), respectively
3: Evaluate individuals of P
4: While termination condition has not been met do
% selection-I
5: Update hP using Eqs. (3), (4)
% Mutation
6: If $rand < 1 - \frac{nfe}{Maxnfe}$
7: For i=1:NP do
8: Generate two random numbers w_1, w_2 in [0, 1] such that $w_1 + w_2 = 1$
9: Generate scale factor F_i based on normal distribution according to Eqs. (12)-(14)
10: Randomly select index r in [1, NP] as $r \neq i$
11: Apply mutation operator according to Eq. (10)
12: End
13: Else
14: For i=1:NP do
15: Generate two random numbers w_1, w_2 in [0, 1] such that $w_1 + w_2 = 1$
16: Generate scale factor F_i based on normal distribution according to Eqs. (12)-(14)
17: Apply mutation operator according to Eq. (11)
18: End
19: End
% Crossover
20: $map_{1:NP,1:D}=1$
21: Apply crossover operator for generate trial population T using Eqs. (7), (6)
22: Apply boundary control mechanism for T according to Eq. (8)
% Selection-II
23: Evaluate individuals of T
24: Apply the greedy selection between individuals of T & P via Eq. (9) to select better one as P
25: Output the best solution
26: End

شکل ۱- شبه‌کد الگوریتم IBSA

برای ارزیابی کارایی الگوریتم IBSA ، BSA استاندارد [۶]، یک نسخه بهبود یافته از آن LBSA[10] [۱۷] و چندین الگوریتم دیگر که دارای عملکرد خوبی هستند از قبیل DTT-PSO[11] [۲۸]، IABC[12] [۲۴] و jDE[13] [۲۹] برای مقایسه استفاده می‌شوند. جهت مقایسه عادلانه، تنظیم مقادیر مولفه‌های هر الگوریتم همان‌گونه است که در مقالات مدنظر آمده‌اند. در جدول ۲، پیکربندی الگوریتم IBSA و الگوریتم‌های دیگر لیست شده است. قابل ذکر است برای همه الگوریتم‌ها اندازه جمعیت برابر با ۵۰ در نظر گرفته می‌شود. همه آزمایشات با استفاده از نرم‌افزار Matlab 2017b روی یک رایانه با مشخصات Intel Core(TM) i7-8700 CPU @ 3.20GHz و RAM 8GB انجام گرفته‌اند.

[10] Learning backtracking search optimization
[11] Particle swarm optimization using dynamic tournament topology
[12] Improved artificial bee colony
[13] Self-adaptive Differential Evolution

جدول ۱- چالش ۱۰۰ رقمی توابع محک CEC 2019

شماره	نام تابع	بعد	فضای جستجو	مقدار بهینه
F_1	Storn's Chebyshev Polynomial Fitting Problem	9	$[-8192, 8192]^D$	1
F_2	Inverse Hilbert Matrix Problem	16	$[-16384, 16384]^D$	1
F_3	Lennard-Jones Minimum Energy Cluster	18	$[-4, 4]^D$	1
F_4	Rastrigin's Function	10	$[-100, 100]^D$	1
F_5	Griewangk's Function	10	$[-100, 100]^D$	1
F_6	Weierstrass Function	10	$[-100, 100]^D$	1
F_7	Modified Schwefel's Function	10	$[-100, 100]^D$	1
F_8	Expanded Schaffer's F6 Function	10	$[-100, 100]^D$	1
F_9	Happy Cat Function	10	$[-100, 100]^D$	1
F_{10}	Ackley Function	10	$[-100, 100]^D$	1

جدول ۲- مقدار مولفه‌های الگوریتم‌های مقایسه شونده

نام الگوریتم	مقدار مولفه‌ها
IBSA	F_{max}=1, F_{min}=0.4, mixrate=1
BSA	mixrate=1, F=N (0,3)
LBSA	mixrate=1, F=3.rand
DTT-PSO	P=0.05, M=6, K=0.1, φ=4.1
IABC	SR=0.005, $NP_{threshold}$=15, NP_{min}=10, SSP_{max}=0.9, SSP_{min}=0.1, limit=200
jDE	τ_1= τ_2=0.1, F_1=01, F_u=0.9

۴. ۲- نتایج آماری

جدول ۳ مقادیر دقت‌های متفاوتی را نشان می‌دهد که الگوریتم IBSA در مجموع ۵۰ بار اجرای مستقل روی هر تابع کسب کرده است. با توجه به ستون آخر، مشاهده می‌شود الگوریتم IBSA در توابع F_1، F_5، F_6 و F_{10} عالی ترین امتیازات را کسب کرده است به گونه ای که در تابع F_6 در همه اجراها موفق بوده است. همچنین در تابع F_3 اجرای خوبی از خود نشان داده است. با این حال برای بقیه توابع F_4، F_7 -F_9 در اکثر اجراها IBSA در دام بهینه محلی می‌افتد، چنانکه در توابع F_7 -F_9 ماکزیمم دقت ۲ است. علاوه براین، الگوریتم IBSA از حل تابع F_2 ناتوان است. با توجه به سطر آخر مشاهده می‌شود، الگوریتم IBSA از مجموع ۱۰۰ امتیاز، امتیاز ۵۴/۱۲ را کسب کرده است.

جدول ۳- تعداد ارقام صحیح حدس زده شده توسط الگوریتم IBSA روی توابع CEC 2019 با ۵۰ بار اجرای مستقل

تابع	تعداد ارقام صحیح											امتیاز
	0	1	2	3	4	5	6	7	8	9	10	
F_1	0	0	0	2	1	2	12	3	3	2	25	10
F_2	50	0	0	0	0	0	0	0	0	0	0	0
F_3	0	25	1	0	0	0	1	0	0	0	23	9.52
F_4	35	13	0	0	0	0	0	0	0	0	2	1.32
F_5	0	0	1	5	0	0	0	0	0	0	44	10
F_6	0	0	0	0	0	0	0	0	0	0	50	10
F_7	44	5	1	0	0	0	0	0	0	0	0	0.28
F_8	17	33	0	0	0	0	0	0	0	0	0	1
F_9	0	5	45	0	0	0	0	0	0	0	0	2
F_{10}	16	0	0	0	0	0	0	0	0	0	34	10
مجموع												**54.12**

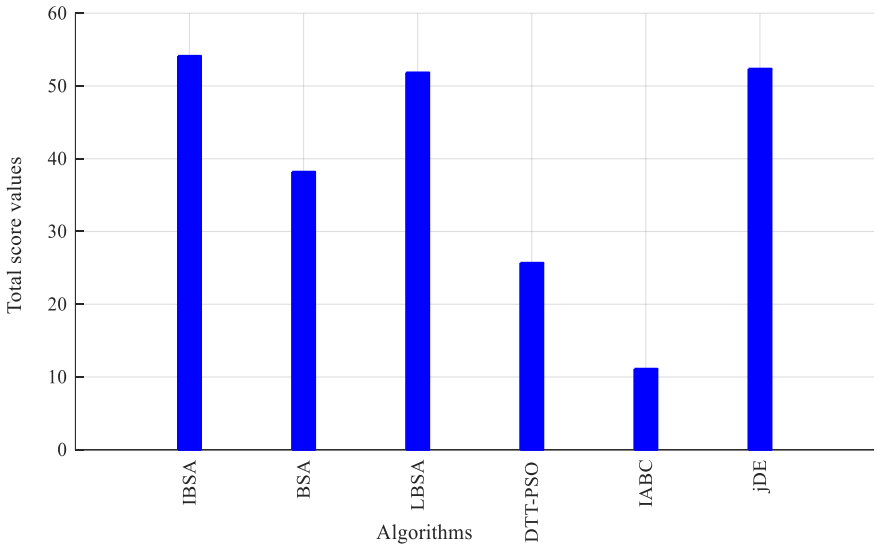
نتایج آماری الگوریتم IBSA و الگوریتم‌های دیگر در جدول ۴ ارائه می‌شود، که در این جدول «Mean»، «STD» و «NFE» به‌ترتیب میانگین خطای مشاهده شده توسط هر الگوریتم با ۵۰ بار اجرا در هر تابع، انحراف از معیار نتایج و میانگین تعداد ارزیابی‌های تابعی مورد نیاز را که باعث می‌شود الگوریتم مورد نظر به یک جواب رضایت بخش در هر تابع برسد را نشان می دهند. همچنین، از آزمون مولفه t با سطح معنی‌داری 0,05 برای نشان دادن یک مقایسه کلی و بی طرفانه بین الگوریتم IBSA با الگوریتم‌های دیگر استفاده شده است که نتایج آن در جدول ۴ بصورت علامات «+»، «≈» و «-» بیان شده است که به‌ترتیب برتری، برابر و بدتر بودن الگوریتم IBSA را نسبت به الگوریتم‌های دیگر نشان می‌دهند. قابل ذکر است بهترین نتایج به صورت فونت برجسته نشان داده می‌شوند.

با توجه به نتایج جدول ۴ مشاهده می‌شودکه الگوریتم IBSA براساس میانگین خطا در توابع F_2 ، F_5 و F_6 بهترین کارایی را از خود نشان می‌دهد.

بعلاوه، در بقیه توابع رقابت تنگاتنگی با الگوریتم‌هایی دارد که بهترین عملکرد را داشته‌اند، چنانکه رتبه دوم را براساس میانگین خطا به‌جز تابع F_9 کسب می‌کند. همچنین مشاهده می‌شود الگوریتم jDE به‌صورت معنی داری بهتر از بقیه الگوریتم‌ها در توابع F_1 و F_7 است. الگوریتم‌های DTT-PSO و IABC در مجموع بدتر از همه الگوریتم‌های دیگر هستند بگونه ای که در توابع F_1، F_2 و F_7 به‌صورت معنی داری بدتر از بقیه الگوریتم‌ها هستند. BSA و LBSA تقریبا عملکرد قابل قبولی از خود نشان می‌دهند. همچنین براساس انحراف از معیار الگوریتم IBSA پایداری خوبی را در همه توابع به‌جز F_7 از خود نشان می‌دهد. با توجه به NFE همه الگوریتم‌ها تقریبا در توابع F_2، F_7-F_9 به نتایج رضایت بخشی دست نیافته‌اند. نتایج نشان می‌دهند که در بقیه توابع الگوریتم‌های jDE ، IBSA و LBSA نرخ همگرایی خوبی نسبت به بقیه الگوریتم‌ها دارند. بعبارتی، در تابع F_1 الگوریتم jDE ، در تابع F_5 الگوریتم IBSA در تابع F_6 الگوریتم‌های jDE و IBSA و در تابع F_{10} الگوریتم‌های LBSA و IBSA دارای سریعترین نرخ همگرایی هستند. براساس امتیاز مشاهده می‌شود که الگوریتم IBSA نسبت به همه الگوریتم‌های دیگر دارای بیشترین امتیاز به‌جز در توابع F_4 و F_7 است. در تابع F_6 همه الگوریتم‌ها به‌جز IABC عالی‌ترین امتیاز را کسب کرده‌اند. گرچه در تابع F_2 همه الگوریتم‌ها هیچ امتیازی کسب نکرده‌اند اما براساس میانگین خطا مشاهده می‌شود الگوریتم IBSA به‌صورت معنی‌دار بهتر از بقیه بوده است. الگوریتم‌های IBSA، LBSA، jDE، BSA، DTT-PSO به‌ترتیب در حل ۴، ۲، ۱، ۱ تابع موفق عمل کرده‌اند، تنها الگوریتمی که در هیچ تابعی موفق عمل نکرده است الگوریتم IABC می‌باشد. در مجموع با توجه به شکل ۲ مشاهده می‌شود، الگوریتم IBSA بیشترین امتیاز را نسبت به بقیه الگوریتم‌ها کسب کرده است. الگوریتم‌های jDE، LBSA، BSA، DTT-PSO و IABC به ترتیب رتبه‌های دوم تا ششم را به دست می‌آورند. همچنین براساس آزمون t به وضوح برتری الگوریتم IBSA نسبت به بقیه الگوریتم‌ها مشاهده می‌شود. بعبارت دقیقتر، IBSA از هیچ الگوریتمی در هیچ تابعی به‌جز DTT-PSO در تابع F_9 و الگوریتم jDE در توابع F_4 و F_7 بدتر نبوده است. بنابراین با توجه به نتایج بالا ثابت می‌شود، الگوریتم IBSA براساس دقت جواب‌های تولیدی، پایداری، سرعت همگرایی و نرخ موفقیت بهتر از بقیه بوده است که این بدلیل توازن خوب بین اکتشاف و بهره‌برداری آن است.

جدول ۲- نتایج آماری مشاهده شده بین الگوریتم IBSA و الگوریتم‌های دیگر روی توابع CEC 2019 با ۵۰ بار اجرای مستقل

الگوریتم	معیار	تابع									
		F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
IBSA	Mean	2.06E-04	**9.71E+00**	2.05E-01	2.07E+00	**9.86E-04**	**8.20E-10**	8.60E+01	7.42E-01	7.77E-02	6.07E+00
	STD	1.10E-03	**1.06E+01**	2.06E-01	1.04E+00	2.77E-03	**1.61E-10**	1.01E+02	4.19E-01	2.11E-02	9.23E+00
	NFE	419499.9	500000	**427124.2**	491522.3	**154915.9**	47594.08	500000	500000	500000	272713.4
	Score	**10**	0	**9.52**	1.32	**10**	**10**	0.28	**1**	**2**	**10**
BSA	Mean	1.52E-03	1.64E+01	**1.65E-01**	2.91E+00	7.90E-03	8.27E-10	8.67E+01	1.45E+00	1.30E-01	1.07E+01
	STD	2.83E-03	1.16E+01	1.96E-01	1.28E+00	9.07E-03	1.77E-10	9.66E+01	2.83E-01	3.13E-02	9.93E+00
	NFE	497687.9	500000	500000	497591.6	419968.5	191176	500000	500000	500000	433476.3
	Score	5.12	0	4.44	0.52	8.84	**10**	0.2	0.12	1.24	7.72
	t-test	+	+	≈	+	+	≈	≈	+	+	+
LBSA	Mean	3.84E-03	5.33E+01	2.22E-01	2.39E+00	5.86E-03	2.25E-05	8.29E+01	**7.39E-01**	7.57E-02	**3.01E+00**
	STD	9.93E-03	3.63E+01	2.05E-01	1.14E+00	6.94E-03	1.55E-04	9.78E+01	4.34E-01	1.64E-02	**7.04E+00**
	NFE	360771.4	500000	445141.1	486328.8	328518.3	65255.96	500000	500000	500000	**241470.6**
	Score	9.24	0	8.36	1.2	9.72	**10**	0.32	**1**	**2**	**10**
	t-test	+	+	≈	≈	+	≈	≈	≈	≈	≈
DTT-PSO	Mean	1.15E+04	1.27E+02	3.85E-01	1.23E+01	1.03E-02	2.74E-01	6.34E+02	2.38E+00	**6.54E-02**	1.58E+01
	STD	2.93E+04	4.88E+01	**9.82E-02**	3.67E+00	9.09E-03	6.46E-01	2.16E+02	5.51E-01	2.68E-02	8.48E+00
	NFE	500000	500000	471767.5	500000	355099.2	99354.88	500000	500000	500000	396351.7
	Score	0	0	2.08	0	7.2	**10**	0	0	**2**	4.4
	t-test	+	+	+	+	+	+	+	+	-	+
IABC	Mean	3.64E+05	1.73E+03	2.52E-01	4.61E+00	1.17E-03	3.80E-01	1.38E+02	1.74E+00	9.35E-02	8.95E+00
	STD	1.81E+05	4.34E+02	1.87E-01	1.08E+00	**2.57E-03**	1.70E-01	6.60E+01	**1.73E-01**	**1.44E-02**	8.29E+00
	NFE	500000	500000	500000	500000	497602.3	500000	500000	500000	500000	500000
	Score	0	0	1.88	0	6.24	1	0	0	**2**	0
	t-test	+	+	≈	+	≈	+	+	+	+	≈
jDE	Mean	**8.62E-10**	1.16E+01	3.05E-01	**5.85E-01**	6.26E-03	9.33E-03	**3.59E+00**	7.87E-01	6.92E-02	1.23E+01
	STD	**1.85E-10**	2.07E+01	1.82E-01	**7.25E-01**	6.06E-03	6.60E-02	**4.24E+00**	2.63E-01	2.19E-02	9.18E+00
	NFE	**152242.8**	500000	460699.6	**421932.8**	340372.7	**43709.6**	**499881.8**	500000	500000	444196.1
	Score	**10**	0	5.36	**9.4**	8.32	**10**	**1.16**	**1**	**2**	5.12
	t-test	≈	≈	+	-	+	≈	-	≈	≈	+



شکل۲- مجموع امتیازات مشاهده شده بین الگوریتم‌های مقایسه‌شونده

۵- نتیجه و جمع‌بندی

در این مقاله، یک نسخه بهبود یافته از الگوریتم BSA به‌نام IBSA با هدف توازن بین اکتشاف و بهره‌برداری معرفی شده است. الگوریتم پیشنهادی تنها در فاز جهش با الگوریتم BSA دارای تفاوت است. بعبارت دقیقتر، IBSA سعی نموده است تا فاز جهش را از دو جنبه بهبود دهد. جنبه اول، استفاده از دو نوع جهش مبتنی بر تغییر بردار پایه با رویکرد انتخاب تطبیقی، بدین صورت که در هر نسل با یک احتمال خطی که در طی نسل‌ها کاهش می‌یابد، رویه‌ای که دارای اکتشاف بیشتر است شانس بیشتر برای انتخاب شدن دارد و رفته رفته در طی فرآیند تکامل رویه‌ای که دارای بهره‌برداری بیشتر است با شانس بیشتری انتخاب می‌شود تا به یک جواب بهینه همگرا شود. جنبه دوم، استفاده از یک رویکرد تطبیقی پویا برای تنظیم ضریب مقیاس F استفاده شد که توازن اکتشاف و بهره‌برداری را نیز تحت تاثیر قرار می‌دهد. جهت ارزیابی کارایی الگوریتم، ۱۰ مسئله چندوجهی CEC 2019 با ۱۰۰ رقم چالش استفاده شدند که اخیرا معرفی شده‌اند. همچنین برای مقایسه کارایی الگوریتم IBSA از انواع الگوریتم‌های شناخته شده و کارا استفاده شد از قبیل BSA پایه‌ای، یک نسخه بهبودیافته و موفق از آن LBSA و سه الگوریتم فراابتکاری DTT-PSO، IABC و jDE که در مقالات اصلی خود نتایج رضایت بخشی به‌دست آورده‌اند. نتایج مقایسات نشان دادند که الگوریتم IBSA نسبت به سایر الگوریتم‌های مقایسه شده کارایی بهتری برای حل مسائل چالش برانگیز CEC 2019 براساس دقت، پایداری و سرعت همگرایی دارد.

۶- مراجع

[1]. Back, T.; "*Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*", Oxford university press,1996.
[2]. Holland John, H.; "*Adaptation in natural and artificial systems*", Ann Arbor: University of Michigan Press, No. 584, 1975.
[3]. Storn, R. and K. Price; "*Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces*", Journal of global optimization, No. 189, pp. 341-359, 1997.
[4]. Hansen, N. and A. Ostermeier; "*Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation*", *Proceedings of IEEE international conference on evolutionary computation*, pp. 312-317, 1996.
[5]. Yao, X..,Y. Liu, and G. Lin; "*Evolutionary programming made faster*", IEEE Transactions on Evolutionary computation, No. 383, pp. 82-102, 1999.

[6]. Civicioglu, P.; "*Backtracking Search Optimization Algorithm for numerical optimization problems*", Applied Mathematics and Computation, No. 145, pp. 24, 2013.

[7]. Modiri-Delshad, M. and N.A. Rahim; "*Multi-objective backtracking search algorithm for economic emission dispatch problem*", Applied Soft Computing, No. 398, pp. 479-494, 2016.

[8]. Zhao, W., et al.; "*An improved backtracking search algorithm for constrained optimization problems*", *International Conference on Knowledge Science, Engineering and Management*, pp. 222-233, 2014.

[9]. Zhang, C., et al.; "*Backtracking Search Algorithm with three constraint handling methods for constrained optimization problems*", Expert Systems with Applications, No. 391, pp. 7831-7845, 2015.

[10]. Su, Z..,H. Wang, and P. Yao; "*A hybrid backtracking search optimization algorithm for nonlinear optimal control problems with complex dynamic constraints*", Neurocomputing, No. 586, pp. 182-194, 2016.

[11]. Lin, Q., et al.; "*A hybrid backtracking search algorithm for permutation flow-shop scheduling problem*", Computers & Industrial Engineering, No. 392, pp. 437-446, 2015.

[12]. Modiri-Delshad, M. and N.A. Rahim; "*Multi-objective backtracking search algorithm for economic emission dispatch problem*", Applied Soft Computing, No. 398, pp. 479-494, 2016.

[13]. Modiri-Delshad, M. and N.A. Rahim; "*Solving non-convex economic dispatch problem via backtracking search algorithm*", Energy, No. 390, pp. 372-381, 2014.

[14]. Ahmed, M.S., et al.; "*Real time optimal schedule controller for home energy management system using new binary backtracking search algorithm*", Energy and Buildings, No. 587, pp. 215-227, 2017.

[15]. Ayan, K. and U. Kılıç; "*Optimal power flow of two-terminal HVDC systems using backtracking search algorithm*", International Journal of Electrical Power & Energy Systems, No. 389, pp. 326-335, 2016.

[16]. Brévilliers, M., et al.; "*Fast Hybrid BSA-DE-SA Algorithm on GPU*", *International Conference on Swarm Intelligence Based Optimization*, pp. 75-86, 2016.

[17]. Chen, D., et al.; "*Learning backtracking search optimisation algorithm and its application*", Information Sciences, No. 144, pp. 71-94, 2017.

[18]. Chen, D., et al.; "*A learning and niching based backtracking search optimisation algorithm and its applications in global optimisation and ANN training*", Neurocomputing, No. 226, pp. 579-594, 2017.

[19]. Askarzadeh, A. and L.d.S. Coelho; "*A backtracking search algorithm combined with Burger's chaotic map for parameter estimation of PEMFC electrochemical model*", International Journal of Hydrogen Energy, No. 185, pp. 11165-11174, 2014.

[20]. Chen, D., et al.; "*Backtracking search optimization algorithm based on knowledge learning*", Information Sciences, No. 419, pp. 202-226, 2019.

[21]. Lin, J.; "*Oppositional backtracking search optimization algorithm for parameter identification of hyperchaotic systems*", Nonlinear Dynamics, No. 590, pp. 209-219, 2015.

[22]. Wang, X. and L. Tang; "*An adaptive multi-population differential evolution algorithm for continuous multi-objective optimization*", Information Sciences, No. 505, pp. 124-141, 2016.

[23]. Wang, S., et al.; "*Adaptive backtracking search optimization algorithm with pattern search for numerical optimization*", Journal of Systems Engineering and Electronics, No. 187, pp. 395-406, 2016.

[24]. Ghambari, S. and A. Rahati; "*An improved artificial bee colony algorithm and its application to reliability optimization problems*", Applied Soft Computing, No. 166, pp. 736-767, 2018.

[25]. Chen, Y., et al.; "*Particle swarm optimizer with two differential mutation*", Applied Soft Computing, No. 151, pp. 314-330, 2017.

[26]. Zhao, L., et al.; "*Improved Backtracking Search Algorithm Based on Population Control Factor and Optimal Learning Strategy*", Mathematical Problems in Engineering, No. 394, 2017.

[27]. Price, K., et al.; "*The 100-digit challenge: Problem definitions and evaluation criteria for the 100-digit challenge special session and competition on single objective numerical optimization*", Nanyang Technological University,, Singapore, No. 580, 2018.

[28]. Wang, L..,B. Yang, and J. Orchard; "*Particle swarm optimization using dynamic tournament topology*", Applied Soft Computing, No. 154, pp. 584-596, 2016.

[29]. Sheskin, D.J.; "*Handbook of parametric and nonparametric statistical procedures*", crc Press,2003.