مجله مدل‌سازی پیشرفته ریاضی
سال ۱۴۰۰، دوره ۱۱، شماره ۳، ص ۵۱۵–۵۳۰

شاپا چاپی: ۸۰۸۸–۲۲۵۱
DOI: 10.22055/JAMM.2021.37465.1932

روش بهینه‌سازی دوفازی مبتنی بر الگوریتم‌های مکاشفه‌ای بیگ‌بنگ و سیاه‌چاله

نگار ایزدی [۱] ×، محمد تقی دستجردی [۲]

(۱,۲) گروه ریاضی، دانشکده علوم، دانشگاه زنجان،زنجان، ایران
دبیر مسئول: محمد هادی فراهی

تاریخ دریافت: ۱۴۰۰/۳/۳ تاریخ پذیرش: ۱۴۰۰/۵/۲۴

چکیده: این تحقیق یک الگوریتم دوفازی را برای حل مسایل بهینه‌سازی معرفی و مطالعه می‌کند که ایده اصلی آن مبتنی بر الگوریتم‌های فراابتکاری بیگ‌بنگ و سیاه‌چاله است. در فاز اول این الگوریتم مورچه‌های تصنعی یک منطقه مستطیلی مشبک را در جهت‌های موازی اسکن می‌کنند که بهترین نقاط تعیین‌شده در مسیر مورچه‌ها به‌عنوان نقاط آغازین برای فاز دوم این الگوریتم استفاده می‌شوند. در فاز دوم، الگوریتم‌های بیگ‌بنگ و سیاه‌چاله سعی در بررسی جواب‌های دقیق‌تر در همسایگی نقاط آغازین با کاهش شعاع همسایگی دارند. مثال‌های عددی تأیید می‌کنند که این الگوریتم قادر است جواب بهینه را با دقت مطلوب و هزینه‌های محاسباتی کم‌تری به‌دست آورد.

واژه‌های کلیدی: بهینه‌سازی، الگوریتم‌های مکاشفه‌ای، الگوریتم سیاه‌چاله، الگوریتم بیگ‌بنگ.

رده‌بندی ریاضی: 90C26; 90C30

۱ مقدمه

در مطالعه برخی روش‌های حل مسائل بهینه‌سازی، به مفروضاتی چون مشتق توابع و نقاط آغازین نیاز است که این نقاط آغازین باید درناحیه همگرایی باشند. بیش‌تر الگوریتم‌های بهینه‌سازی از اطلاعات مشتق تابع استفاده می‌کنند که به آنها الگوریتم‌های مشتق‌محور می‌گویند. مشکل این الگوریتم‌ها این است که اگر تابع هدف ناپیوسته و یا ناهموار باشد کارآمد نیستند. ضعف دیگر این الگوریتم‌ها این است که ممکن است در نقاط بهینه محلی گیر کنند، حال آنکه روش‌هایی که مشتق‌محور نیستند مثل الگوریتم‌های فراابتکاری هیچ‌گاه در نقاط بهینه محلی گیر نمی‌کنند. در واقع از الگوریتم‌های فراابتکاری به‌عنوان یک روش بهینه‌سازی سراسری می‌توان بهره برد [۱۱]. این روش‌ها عمدتا از یک فرایند طبیعی، فیزیکی و یا شیمیایی مانند الگوریتم ژنتیک (GA) [۲]، بهینه‌سازی کلنی مورچه‌ها، (ACO) [۳]، بهینه‌سازی ازدحام ذرات، (PSO) [۸]، بهینه‌سازی غذایابی باکتری‌ها (BFO) [۱۰]، و شبیه‌سازی سخت کردن فلزات (SA) [۹] و غیره اقتباس می‌شوند. در سال‌های اخیر، الگوریتم‌های فراابتکاری متعددی برای حل مسائل بهینه‌سازی معرفی شده‌اند که عمده این روش‌ها براساس جستجوهای تصادفی است. همه الگوریتم‌های فراابتکاری فوق‌الذکر به‌صورت موازی با چندین نقطه آغازین، ناحیه شدنی را جستجو می‌کنند. در واقع این الگوریتم‌ها مبتنی بر جمعیت‌اند. منظور از جمعیت، گروهی متشکل از عوامل ساده، مستقل و ناهمگام است که با یکدیگر برای یافتن جواب‌های بهینه یک مسأله بهینه‌سازی، همکاری و تعامل دارند. در یک الگوریتم مبتنی بر جمعیت، هر یک از اعضا یک سری عملیات خاص را اجرا

×نویسنده مسئول مقاله
رایانامه: negarizadi@znu.ac.ir (N.Izadi), tdast@znu.ac.ir (M.T. Dastjerdi)

می‌کند و اطلاعات خود را با دیگران به اشتراک می‌گذارد. این عملیات تا حدودی ساده‌اند اما با این حال تأثیر جمعی آنها، نتایج شگفت آوری ایجاد می‌کند که معروف به هوش ازدحام است [۴، ۱۳] . در این الگوریتم‌ها فعل و انفعالات محلی بین عوامل، یک نتیجه سراسری را فراهم می‌کند که به سیستم اجازه می‌دهد، بدون استفاده از کنترل کننده مرکزی، مسئله بهینه‌سازی را حل کند. دو جنبه زیر دراغلب الگوریتم‌های مبتنی بر جمعیت مشترک است:

۱–اکتشاف: توانایی ارزیابی جواب‌های شدنی که در همسایگی یک جواب فعلی نیستند.

۲–بهره برداری: توانایی یافتن جواب بهینه که در همسایگی یک جواب شدنی است.

دریک الگوریتم فراابتکاری بخش اول تکرارها، ناحیه شدنی را جستجو می‌کند تا جواب‌های جدید پیدا کند و بخش دوم تکرارها عملیات بهره برداری را انجام می‌دهند. از سوی دیگر، جمعیت این الگوریتم، برای تحقق بخشیدن به اهداف اکتشاف و بهره برداری، سه مرحله زیر را در هر تکرار طی می‌کند:

۱– خودسازگاری: در این مرحله هر یک از اعضا عملکرد خود را بهبود می‌بخشد.

۲– همکاری: در این مرحله اعضا به انتقال اطلاعات به یکدیگر می‌پردازند.

۳– رقابت: در این مرحله جواب‌های ضعیف‌تر حذف می‌شوند و جای خود را به جواب‌های بهتر می‌دهند.

اجرای این سه مرحله معمولاً به‌صورت تصادفی و مبتنی بر یک تابع توزیع احتمال می‌باشد. به‌عنوان مثال در الگوریتم PSO این سه مرحله با الهام از رفتار دسته جمعی پرندگان برای یافتن غذا در طبیعت، مدل سازی می‌شوند. به این صورت که در آغاز کار، جمعیتی از ذرات به‌صورت تصادفی در ناحیه شدنی تشکیل می‌شوند و این الگوریتم با به‌روز کردن موقعیت این ذرات، درهر تکرار سعی در یافتن جواب بهینه می‌نماید؛ درواقع این مرحله رقابت است. در هر گام، موقعیت مکانی هر ذره، که معادل موقعیت یک پرنده در طبیعت است، با استفاده از دو داده به‌روز می‌شود. اولین داده، بهترین موقعیتی است که تاکنون ذره (پرنده) موفق به رسیدن به آن شده است؛ این مرحله را خودسازگاری گوییم. این موقعیت شناخته و ذخیره می‌شود و آن را در الگوریتم با pbest نمایش می‌دهند. دومین داده‌ای که توسط الگوریتم مورد استفاده قرار می‌گیرد، بهترین موقعیتی است که تا کنون توسط جمعیت ذرات به‌دست آمده است که آن را gbest می‌گویند این مرحله همان مرحله همکاری است. تمام الگوریتم‌های مبتنی بر جمعیت، نتایج رضایت بخشی را ارائه می‌دهند و هیچ الگوریتم فراابتکاری وجود ندارد که بتواند عملکردی برتر از دیگران را در حل تمام مسائل بهینه‌سازی ارائه دهد. هدف از این مقاله ارائه یک الگوریتم جدید فراابتکاری برای حل مسئله بهینه‌سازی زیر است.

$$\begin{aligned} &\min\{\mathrm{f}(\mathrm{X}) \ : \ \mathrm{X} \in \mathrm{W}\} \\ &\mathrm{W} = \{\mathrm{X} \in \mathbb{R}^{\mathrm{n}} : -\infty < \mathrm{l_k} \leq \mathrm{x_k} \leq \mathrm{u_k} < +\infty, \mathrm{k} = ۱, ۲, \ldots \mathrm{n}\} \end{aligned} \tag{۱.۱}$$

که در آن $\mathrm{f} : \mathbb{R}^{\mathrm{n}} \longrightarrow \mathbb{R}$ یک تابع غیر خطی است و $\mathrm{l_k}$ و $\mathrm{u_k}$ به‌ترتیب کران‌های پایین و بالا برای مولفه $\mathrm{x_k}$ هستند.
واضح است که می‌توان مسائل ماکزیمم‌سازی را به‌صورت زیر به یک مسئله مینیمم‌سازی تبدیل کرد:

$$\max \mathrm{f}(\mathrm{X}) = -\min(-\mathrm{f}(\mathrm{X}))$$

در این مسئله موقعیت ذره i ام در ناحیه n –بعدی W را با $\mathrm{X_i}$ که $\mathrm{i} = ۱, ۲, \ldots, \mathrm{n}$ نشان می‌دهیم، مقدار تابع هدف در این نقطه را با $\mathrm{f}(\mathrm{X_i}) = \mathrm{f_i}$ نشان داده و آن را تابع شایستگی یا تابع تناسب می‌نامیم.

در این مقاله یک الگوریتم دوفاز بر پایه الگوریتم‌های فراابتکاری ارائه می‌کنیم که جواب بهینه سراسری مسئله (۱.۱) را با دقت بهتر می‌یابد. سازماندهی این مقاله به شرح زیر است: در بخش ۲ ، ما الگوریتم استاندارد بیگ‌بنگ را ارائه می‌دهیم. الگوریتم‌های بهینه‌سازی ازدحام ذرات سیاه‌چاله تصادفی و الگوریتم سیاه‌چاله در بخش ۳ معرفی می‌شوند. در ادامه، ما الگوریتم جدید جمعیتی خود، یعنی الگوریتم دوفاز را در بخش ۴ معرفی می‌کنیم. در بخش ۵ به بررسی همگرایی الگوریتم‌های فراابتکاری می‌پردازیم. سرانجام در بخش ۶ ، نتایج حاصل از اجرای الگوریتم دوفاز را روی برخی توابع آزمون، ارائه می‌کنیم که این نتایج کارایی الگوریتم را تایید می‌کنند.

۲ شرح کوتاه از الگوریتم بیگ‌بنگ–بیگ کرانچ

الگوریتم بیگ‌بنگ–بیگ‌کرانچ $\mathrm{BB-BC}$ توسط ارل و اکسین [۵] در سال۲۰۰۶ ارائه شد که هر گام این الگوریتم شامل دو مرحله زیر است:

۱–مرحله بیگ‌بنگ BB

۲– مرحله بیگ کرانچ BC

در تکرار اول و در مرحله بیگ‌بنگ، مشابه سایر الگوریتم‌های فراابتکاری، N نقطه آغازین به‌طور تصادفی از کل فضای جستجو با یک تابع توزیع یکنواخت در نظر گرفته می‌شوند. مرحله BC که بعد از BB اعمال می‌شود، مانند یک عملگر عمل می‌کند که N ورودی و فقط یک خروجی دارد که به خروجی آن مرکز جرم می‌گویند و آن را با نماد $\mathrm{X^c}$ نشان می‌دهند و از رابطه زیر به‌دست می‌آید.

$$X^{c} = \frac{\sum_{i=1}^{N} \frac{1}{f_i} X_i}{\sum_{i=1}^{N} \frac{1}{f_i}} \quad (1.2)$$

در تکرارهای بعدی، در مرحله بیگ‌بنگ، نقاط جدید با اضافه و یا کم نمودن یک سری اعداد تصادفی به مولفه‌های مرکز جرم به دست می‌آیند. اندازه این اعداد تصادفی در هرتکرار کاهش می‌یابد؛ یعنی در هر تکرار نقاطی که در BB ایجاد می‌شوند نسبت به نقاط ایجاد شده توسط BB درتکرار بعدی، فاصله بیش‌تری تا مرکز جرم دارند.

یکی از کاربردهای الگوریتم بیگ‌بنگ- بیگ کرانچ تقریب یک مسئله خطی نامتغیر با زمان مرتبه بالا به مدل مرتبه پایین آن است که این روش ترکیب الگوریتم بیگ‌بنگ- بیگ کرانچ و روش تطبیق لحظه ای است که در مرجع [۱] به تفصیل مطالعه شده است. همچنین یک الگوریتم ترکیبی متشکل از الگوریتم بیگ‌بنگ- بیگ کرانچ و الگوریتم دیفرانسیل تکاملی توسط پرایگو در [۱۲] ارائه شده است که قابلیت‌های اکتشاف و بهره‌برداری الگوریتم اصلی را برای یافتن جواب بهینه سراسری افزایش داده است. نتایج این کار نشان می دهد الگوریتم ترکیبی جدید از الگوریتم اصلی بیگ‌بنگ- بیگ کرانچ عملکرد بهتری دارد.

۳ بررسی اجمالی الگوریتم سیاه‌چاله

۱.۳ بهینه‌سازی سیاه‌چاله

این روش بهینه‌سازی که از پدیده سیاه‌چاله الهام گرفته شده، توسط حاتملو [۷] معرفی شده است. در این الگوریتم ابتدا N نقطه به تصادف به‌عنوان نقاط آغازین در ناحیه شدنی انتخاب می‌شوند. سپس نقطه‌ای که بهترین مقدار تابع هدف را دارد به عنوان سیاه‌چاله انتخاب می‌شود و سایر نقاط ستاره‌ها را تشکیل می‌دهند. سپس، هر ستاره بر اساس مکان فعلی‌خود و یک عدد تصادفی به سمت سیاه‌چاله حرکت می‌کند. به‌عبارت دیگر داریم:

$$X_i(t+1) = X_i + \mathrm{rand}\,(X_{BH} - X_i(t)) \quad (i = 1, 2, \ldots N) \quad (1.3)$$

که در آن $X_i(t)$ و $X_i(t+1)$ به‌ترتیب مکان i امین ستاره در تکرارهای t و $t+1$ هستند، X_{BH} محل سیاه‌چاله در فضای جستجو است، rand یک عدد تصادفی در بازه $[0, 1]$ و N تعداد ستاره‌ها است.

بعد از حرکت به سمت سیاه‌چاله، ممکن است یک ستاره در موقعیت جدید خود، تابع شایستگی بهتری نسبت به سیاه‌چاله داشته باشد. در چنین حالتی، سیاه‌چاله به محل آن ستاره حرکت می‌کند و الگوریتم BH با سیاه‌چاله در مکان جدید ادامه خواهد یافت. سپس در تکرار بعدی ستاره‌ها به سمت این مکان جدید حرکت می‌کنند. علاوه بر این، هنگام حرکت ستاره‌ها به سمت سیاه‌چاله، احتمال رسیدن به یک همسایگی مطلوب سیاه‌چاله وجود دارد. هر ستاره (هر جواب) که به این همسایگی سیاه‌چاله می‌رسد، توسط سیاه‌چاله جذب می‌شود. هر بار که یک ستاره می‌میرد یا توسط سیاه‌چاله جذب می‌شود، یک جواب دیگر (ستاره) به طور تصادفی در فضای جستجو ایجاد می‌شود که این کار برای ثابت نگه داشتن تعداد جواب‌ها انجام می‌شود. تکرار بعدی بعد از جابه‌جایی همه ستاره‌ها انجام می‌شود. شعاع همسایگی مطلوب در الگوریتم سیاه‌چاله با استفاده از معادله زیر محاسبه می‌شود:

$$R = \frac{f_{BH}}{\sum_{i=1}^{N} f_i} \quad (2.3)$$

که در آن f_{BH} و f_i به‌ترتیب تابع شایستگی سیاه‌چاله و تابع شایستگی i امین ستاره‌اند. وقتی فاصله بین هریک از ستاره‌ها (جواب‌ها) و سیاه‌چاله (جواب با بهترین تابع هدف) از R کم‌تر باشد، آن جواب سقوط می‌کند و یک جواب جدید به طور تصادفی در فضای جستجو ایجاد می‌شود. مراحل اصلی الگوریتم BH در الگوریتم ۱ بیان شده است.

۲.۳ بهینه‌سازی ازدحام ذرات سیاه‌چاله تصادفی

این الگوریتم از پدیده سیاه‌چاله الهام گرفته شده و مبتنی بر الگوریتم PSO [۸] است که توسط ژانگ [۱۴] معرفی شده است. این الگوریتم را الگوریتم بهینه‌سازی ازدحام ذرات سیاه‌چاله تصادفی $RBH-PSO$ می‌نامند. ذکر این نکته حائز اهمیت است که این الگوریتم با الگوریتم سیاه‌چاله BH متفاوت است. در این الگوریتم ابتدا N ذره (نقطه) به‌صورت تصادفی در ناحیه شدنی انتخاب می‌شوند.در ادامه نقاط pbest

Algorithm 1 Black Hole

Initialize a population of stars with random locations in the search space
Loop
For each star, evaluate the objective function
Select the best star that has the best fitness value as the black hole
Change the location of each star according to the equation (1.3)
If a star arrives at a location with a lower cost than the black hole, swap their locations.
If a star crosses the black hole's neighborhood in a radius of (2.3), replace it with a new star at a random location in search space.
If a termination criterion (a maximum number of iterations or a sufficiently good fitness) is met, exit the loop
End loop

و gbest مطابق الگوریتم PSO انتخاب شده و به روز رسانی می‌شوند. در هر تکرار در همسایگی gbest به شعاع R یک سیاه‌چاله به طور تصادفی با توزیع یکنواخت ایجاد می‌کنیم و برای این سیاه‌چاله یک عدد آستانه در نظر می‌گیریم. اگر ذره‌ای در همسایگی این سیاه‌چاله به شعاع R قرار بگیرد، توسط سیاه‌چاله جذب می‌شود و اگر ذره‌ای خارج از این همسایگی باشد، برای آن یک عدد حقیقی مثبت به‌تصادف در نظر می‌گیریم. اگر این عدد کم‌تر یا مساوی عدد آستانه سیاه‌چاله باشد، ذره به سمت سیاه‌چاله حرکت می‌کند و اگر این عدد بیش‌تر از عدد آستانه سیاه‌چاله باشد، الگوریتم PSO مکان جدید ذره را محاسبه می‌کند.

۴ طرح کلی الگوریتم دوفاز

تصادفی بودن انتخاب نقاط آغازین در اکثر الگوریتم‌های فرابتکاری باعث اتلاف انرژی (افزایش حجم محاسبات) می‌شود و در نتیجه همگرایی را کند می‌کند. الگوریتم ارائه‌شده در این بخش به‌دلیل بررسی کل ناحیه جواب در فاز اول، از اتلاف انرژی می‌کاهد. فاز دوم این الگوریتم، در همسایگی نقاط تعیین شده در فاز اول، از یک الگوریتم بهینه‌سازی مثل $\mathrm{BB-BC}$ یا BH استفاده می‌کند تا پس از چند تکرار جواب‌ها را بهبود ببخشد. شعاع همسایگی مورد استفاده در فاز دوم، متناسب با تکرارهای الگوریتم کاهش می‌یابد و تضمین می‌کند که الگوریتم با دقت مطلوب به نقطه بهینه همگرا شود. یکی دیگر از اشکالات الگوریتم‌های فرابتکاری این است که برای دستیابی به جواب‌هایی با تابع هدف بهتر، الگوریتم باید چندین بار اجرا شود. در این حالت دامنه خطا افزایش می‌یابد. در نتیجه، در مسائلی که جواب بهینه سراسری آنها ناشناخته است، پذیرش یک پاسخ به‌عنوان جواب بهینه سراسری ممکن است باعث بروز خطای زیادی شود. اما جواب‌های به‌دست‌آمده توسط الگوریتم معرفی‌شده در این بخش، درهمسایگی کوچکی از جواب دقیق قرار دارند و بهینه‌ترند.

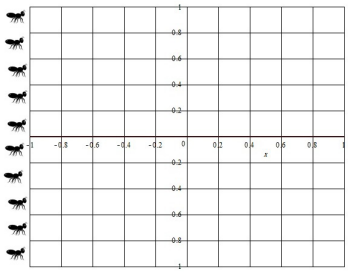
۱.۴ فاز اول: جستجوی مورچه‌های تصنعی

در این فاز تعدادی از عوامل به‌نام مورچه‌های تصنعی در ناحیه شدنی مستطیل شکل در جهت‌های موازی حرکت می‌کنند. هر مورچه می‌تواند فقط روی یک مسیر از شبکه به جلو حرکت کند. به اشکال ۱ و ۲ توجه شود. مورچه‌های تصنعی در حین حرکت در مسیرهایشان، نقاطی را که دارای بهترین شایستگی‌اند، تشخیص می‌دهند و موقعیت این نقاط را در حافظه خود ذخیره می‌کنند. نقاط به‌دست آمده در این فاز، نقاط آغازین فاز بعدی خواهند بود.

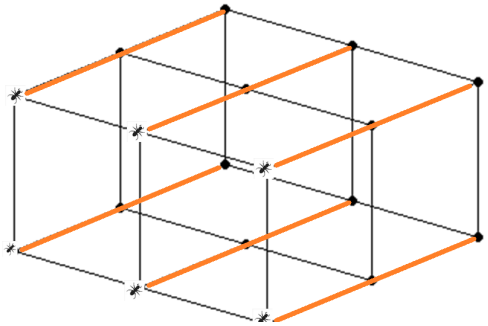
درالگوریتم‌های فرابتکاری ناحیه جواب به‌طور تصادفی جستجو می‌شود. در واقع نقاط آغازین در ناحیه جواب توسط عوامل نابینا انتخاب می‌شوند، در حالی که در این فاز الگوریتم به‌منظور یافتن زیرناحیه‌های امیدوار کننده، ناحیه شدنی را اسکن می‌کند که زیرناحیه‌های امیدوار کننده، همسایگی نقاطی با بهترین مقدار تابع هدف‌اند که مورچه‌های تصنعی در طول مسیر خود یافته‌اند. اجرای فاز دوم در نواحی امیدوارکننده باعث می‌شود که جواب‌های به‌دست آمده از دقت یکنواختی برخوردار باشند. مزیت دیگر این الگوریتم قابلیت پیاده‌سازی آن در محاسبات موازی است. الگوریتم ۲، این فاز را به وضوح نشان می‌دهد.

۲.۴ فاز دوم: استفاده از بیگ‌بنگ

در این فاز، الگوریتم $\mathrm{BB-BC}$ برای بهبود نقاط ارائه شده در فاز قبلی وارد عمل می‌شود. درتکرار اول و در بخش BB نقاط به‌دست‌آمده از فاز اول مورد استفاده قرار می‌گیرند. در بخش BC از این نقاط برای محاسبه مرکز جرم استفاده می‌شود. در تکرارهای بعدی، در بخش



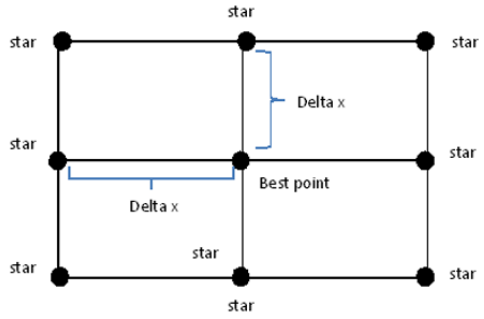
شکل ۱: شبکه دوبعدی



شکل ۲: شبکه سه بعدی

Algorithm 2 Ant Scanning

Mesh–grid(L,U,Dx).
For all ants **do**
scan related line by determined step size(Dx).
best position(i) = position of the best fitness function value in i–th line .
end for.
Return best position



شکل ۳: algorithm NBH in point best around stars Creating

BB در همسایگی مرکز جرم و با شعاع همسایگی مناسب، تعداد معینی نقاط تصادفی در جهات مختلف ایجاد می‌شوند که بخش BC از این نقاط جدید برای محاسبه مرکز جرم استفاده می‌کند. همگرایی الگوریتم با تعداد مناسب تکرار و نرخ کاهش شعاع تضمین می‌شود. جزئیات بیش‌تر فاز دوم الگوریتم به شرح زیر است:

در تکرار اول مقدار اولیه شعاع همسایگی را همان طول گام مورچه‌های تصنعی، یعنی Dx ، درنظر می‌گیریم:

$$r_۱ = Dx$$

شعاع همسایگی تکرار i ام الگوریتم BB – BC از رابطه زیر به‌دست می‌آید:

$$r_i = \frac{r_۱(maxiter + ۱ - i)^p}{maxiter^p} \qquad i = ۱, ۲, ...maxiter \qquad (۱.۴)$$

که در آن maxiter تعداد تکرار الگوریتم BB – BC است و p پارامتری است که نرخ کاهش اندازه شعاع را تعیین می‌کند. مقدار p اغلب به مسئله بستگی دارد؛ اگر $p = ۱$ باشد، شعاع به‌صورت خطی کاهش می‌یابد. در این مسئله مقادیر $p = ۲, ۳, ۵, ۷$ درنظر می‌گیریم. الگوریتم ۳ بیانگر فاز دوم است که برپایه BB – BC است.

۳.۴ فاز دوم: استفاده ازالگوریتم جدید سیاه‌چاله

فاز دوم الگوریتم جدید سیاه‌چاله New Black Hole(NBH) برای بهبود نقاط ارائه‌شده در فاز اول وارد عمل می‌شود. علی‌رغم الگوریتم BH که در بخش ۱.۳ اشاره شد، فاز NBH با جمعیتی که به‌صورت تصادفی ایجاد شده کار نمی‌کند درواقع در این فاز نقاط آغازین در

Algorithm 3 BB–BC

Determine (nn, maxiter,p)
nn = number of neighbors in each promising area.
maxiter = number of iterations
p = Radius size reduction rate
Input (bestposition, Dx) % The best positions were obtained in phase 1
For i =1 to maxiter do
If i=1 **do** use the best positions to calculate the center of mass by 1.2
else
By adding or subtracting normal random numbers, determine nn neighbors in a promising area
Calculate the fitness function of each neighbors
Calculate center of mass by (1.2)
end for
Return center of mass

اطراف بهترین نقطه به‌دست‌آمده توسط مورچه‌های تصنعی در فاز اول هستند. در این روش جواب‌های اولیه با فاصله Dx از بهترین نقطه در هر بعد به‌دست می‌آیند. شکل ۳ یک مثال دوبعدی را به‌صورت نمادین نشان می‌دهد. تفاوت دیگر بین BH و NBH در این است که، موقعیت سیاه‌چاله در الگوریتم BH هنگامی تغییرمی‌کند که موقعیت همه ستارگان با توجه به محل سیاه‌چاله به‌روز شوند. درحالی که در الگوریتم NBH ابتدا موقعیت یک ستاره به‌روز می‌شود؛ سپس مقدار تابع شایستگی ستاره در مکان جدید با مقدار تابع شایستگی سیاه‌چاله مقایسه می‌شود و چنانچه ستاره تابع شایستگی بهتری داشته باشد، مختصات آن به سیاه‌چاله اطلاق می‌شود و موقعیت بقیه ستاره‌ها با توجه به سیاه‌چاله جدید به‌روزرسانی می‌شود. به‌طورخلاصه موقعیت سیاه‌چاله به‌صورت پویا تغییر می‌کند. الگوریتم ۴، فاز دومی را که بر پایه NBH است، بیان می‌کند.

Algorithm 4 New Black Hole

Input(bestpoint,nbh,Dx)
Create stars located at a distance of Dx in each dimension around the bestpoint
Calculate the fitness value of all stars and determine the Black hole between the stars and the bestpoint
For i = 1 to nbh **do**
For each star **do**
Move all the stars towards the black hole using the equation (1.3)
Check if the new location of the star is in W
Compare the fitness value of the black hole with the star
Swap their location, if necessary
End
End
Return best-fit point

به‌دلیل عملکرد الگوریتم NBH در یک فضای کوچک، ستاره‌ها هرگز توسط سیاه‌چاله بلعیده نمی‌شوند و تا آنجا که ممکن است می‌توانند به سیاه‌چاله نزدیک شوند. تعداد تکرارهای استفاده شده در NBH متناسب با هر میزان دقت دلخواه است و توسط کاربر تعیین می‌شود. الگوریتم‌های ۵ و ۶ به‌ترتیب الگوریتم‌های دوفاز بر پایه BB – BC و BH اند.

۵ همگرایی الگوریتم‌های فراابتکاری

الگوریتم‌های فراابتکاری در دو دهه گذشته از نظر عملکرد و کارایی پیشرفت‌های زیادی داشته‌اند. از حیث تحلیل‌های نظری این الگوریتم‌ها، یکی از اساسی‌ترین موضوعات این است که آیا این الگوریتم‌ها به جواب بهینه همگرا می‌شوند یا خیر؟ گوجار در [۶] به بررسی همگرایی الگوریتم‌های فراابتکاری به یک جواب به اندازه کافی خوب پرداخته است که آن را با نماد x_t^{bsf} نشان می دهیم. x_t^{bsf} در واقع بهترین تقریب

Algorithm 5 Main Function calls BB–BC

Input (L,U,Dx,fitness function).
bestgrid = Ant Scanning (L,U,Dx).
For bestgrid(i) in bestgrid do
  bestneighbor(i) = BB–BC (bestgrid(i),Dx) .
End.
Find bestneighbor(j) that has best fitness.

Algorithm 6 Main Function calls BH

Input (L,U,Dx,fitnessfunction,maxiter,nbh)
nbh is number of iteration in NBH
bestgrid = Ant Scanning (L, U, Dx)
For each bestpoint in each line **do**
  For $i = 1$ to maxiter
    bestpoint = NBH (bestpoint,nbh,Dx)
    update Dx by equation (1.4)
  End
End

پیشنهادی در حال حاضر برای جواب بهینه در تکرار t ام است. اگر مقدار بهینه تابع هدف در مسئله ۱.۱ f^* باشد، با فرض محدود بودن ناحیه W، نشان می‌دهیم مقدار $f(x_t^{bsf})$ وقتی $t \to \infty$ به f^* همگرا می‌شود. برای این منظور تابع شاخص I را به‌صورت زیر تعریف می‌کنیم.

$$Z_t = I(f(x_t^{bsf})) = \begin{cases} 1 & \text{if } f(x_t^{bsf}) = f^* \\ 0 & \text{otherwise} \end{cases}$$

اگر جواب بهینه در تکرار t پیدا شود، آنگاه $Z_t = ۱$ و در غیر این صورت $Z_t = ۰$. همچنین زمان اولین برخورد را با $T_۱$ نشان می‌دهیم و به‌صورت زیر تعریف می کنیم:

$$T_۱ = \min\{t \geq ۱ : Z_t = ۱\}$$

در ادامه فرض می‌کنیم E نماد امید ریاضی و P_r تابع احتمال باشد، آنگاه

$$m_t = E(Z_t) = P_r(Z_t = ۱) = p_r(T_۱ \leq t) \quad t = ۱, ۲, ...$$

در این صورت $f(x_t^{bsf})$ به f^* همگرا می‌شود اگر و تنها اگر $m_t \to ۱$ وقتی $t \to \infty$. اگر حالت ساده جستجوی تصادفی را در نظر بگیریم و فرض کنیم که احتمال رخ دادن جواب بهینه در یک تکرار از الگوریتم P باشد، که $۰ < P \leq ۱$، با توجه به مستقل بودن تکرارها و توزیع هندسی استاندارد داریم:

$$P_r(Z_t = ۰) = (۱ - P)^t$$

$$\lim_{t \to +\infty} P_r(Z_t = ۰) = ۰$$

بنابراین $m_t \to ۱$ وقتی $t \to \infty$. به‌عبارت دیگر همگرایی به مقدار بهینه وقتی $t \to \infty$ با احتمال ۱ اتفاق می‌افتد. برای جزئیات بیش‌تر مرجع [۶] دیده شود. بنابراین با توجه به آنچه گذشت وقتی الگوریتم درحالتی که فقط شامل جستجوی تصادفی است، همگرایی آن وقتی $t \to \infty$ با احتمال ۱ اتفاق می‌افتد. از طرف دیگر وقتی در الگوریتم‌های دوفاز کل ناحیه توسط مورچه‌های تصنعی مورد بررسی قرار می‌گیرد، مقدار P، یعنی احتمال مشاهده جواب بهینه در یک تکرار، نسبت به احتمال مشاهده جواب بهینه در حالت جستجوی تصادفی افزایش می یابد و یا حداقل تغییر نمی‌کند. در نتیجه با استدلالی مشابه آنچه گذشت همگرایی الگوریتم‌های دوفاز نتیجه می‌شود. به‌عبارت دیگر الگوریتم‌های دوفاز وقتی $t \to \infty$ همگرا به جواب مسئله با احتمال ۱ است.

Table 1: Test Functions

Function name	Domain	Minimum	parameters
Schaffer	[-100,100]	$x^* = (0,0)$ $f^* = 0$	$p = 2$ $nn = 3$
Rastrigin	[-5.12,5.12]	$x^* = (0,0)$ $f^* = 0$	$p = 2$ $nn = 3$
Rosenbrock	[-100,100]	$x^* = (1,1)$ $f^* = 0$	$p = 3$ $nn = 5$
Levy	[-10,10]	$x^* = (1,1)$ $f^* = 0$	$p = 2$ $nn = 3$
Ackley	[-32.768, 32.768]	$x^*(0,0)$ $f^* = 0$	$p = 2$ $nn = 3$
Drop-Wave	[-5.12, 5.12]	$x^* = (0,0)$ $f^* = -1$	$p = 7$ $nn = 5$
Special test function	[-10,10]	$x^* = (0,0)$ $f^* = 0$	$p = 2$ $nn = 3$

Table 2: BB-BC on schaffer function

Dx	maxiter	x_1	x_2	f
0.13	50	3.22851×10^{-5}	1.44106×10^{-5}	1.2499×10^{-12}
0.033		1.01316×10^{-6}	-1.48658×10^{-4}	2.22019×10^{-13}
0.033	70	3.08258×10^{-6}	5.49498×10^{-6}	3.96971×10^{-14}
0.003		0.0	0.0	0.0

Table 3: BB–BC on Rastrigin Function

Dx	maxiter	x_1	x_2	f
0.13	30	2.64547×10^{-6}	5.09492×10^{-5}	5.16380×10^{-7}
0.13	50	9.17583×10^{-5}	2.38430×10^{-5}	1.78310×10^{-6}
0.003		0.0	0.0	0.0

۶ پیاده سازی

در این بخش، جزئیات مختلف اجرای الگوریتم دوفاز را روی دسته‌ای از توابع که به‌عنوان توابع آزمون درمسائل بهینه‌سازی شناخته شده‌اند، مورد بررسی قرار می‌دهیم. این توابع آزمون معمولاً دارای نقاط بهینه سراسری و تعدادی نقاط بهینه محلی با محدودیت‌های ساده‌اند. دامنه این توابع و نقاط بهینه سراسری آنها و پارامترهای مورد استفاده در الگوریتم، در جدول ۱ ذکر شده‌اند و همچنین نمودار این توابع و برخی ویژگی‌های آنها در ضمیمه این مقاله ارائه شده است. نتایج عددی حاصل از اجرای الگوریتم دوفاز بر پایه $BB-BC$ بر روی توابع آزمون در حالت دوبعدی در جداول ۲ ، ۳ ، ۴ ، ۵ ، ۶ ، ۷ و ۸ ارائه شده است. با استفاده از الگوریتم BH روی توابع آزمون ارائه شده در جدول ۱، نقاط بهینه‌ای به‌دست آمده‌اند، که در جداول ۹ ، ۱۰ ، ۱۱ ، ۱۲ ، ۱۳ ، ۱۴ ، ۱۵ ذکر شده‌اند. تمامی الگوریتم‌های مورد مطالعه درمحیط Matlab پیاده‌سازی شده‌اند.

لازم‌به‌ذکر است که الگوریتم‌های دوفاز مذکور را می‌توان روی توابع آزمون به‌ازای ابعاد دلخواه بکار برد. در این تحقیق برای سهولت و وضوح بیش‌تر اغلب از توابع آزمون دوبعدی استفاده شده است. البته به‌عنوان نمونه الگوریتم دوفاز بر پایه بیگ‌بنگ بیگ کرانچ را روی دو تابع راستریگین و روزنبروک در حالت پنج‌بعدی پیاده‌سازی کرده‌ایم که نتایج آن در جداول ۱۶ و ۱۷ ارائه شده است.

۷ نتیجه‌گیری

با توجه به آن‌چه گذشت، الگوریتم‌های دوفاز مبتنی بر $BB-BC$ و BH هزینه محاسباتی بسیار کمی دارند و در یک ناحیه نسبتاً کوچک می‌توانند جواب‌هایی با دقت بالا را تعیین کنند. همچنین دیدیم که جواب‌های حاصل از پیاده‌سازی الگوریتم‌های استاندارد $BB-BC$ و BH بر روی مثال‌هایی که در بخش قبل ذکر شد دارای طیف گسترده ای از خطا می‌باشند و همانطور که بیان شد، این از اشکالات روش‌های فراابتکاری است و چنانچه یکی از این جواب‌ها پذیرفته شود ممکن است خطای زیادی رخ دهد. با این حال با اجرای الگوریتم دوفاز بر روی

Table 4: BB–BC on Rosenbrock Function

Dx	maxiter	$\|x_1 - 1\|$	$\|x_2 - 1\|$	f
0.011	70	2.00783×10^{-4}	4.02553×10^{-4}	4.04034×10^{-8}
0.003	50	1.62378×10^{-4}	3.25719×10^{-4}	3.64544×10^{-8}
0.0015		9.69347×10^{-7}	2.52907×10^{-6}	1.75646×10^{-12}
0.003	70	6.47156×10^{-5}	1.29739×10^{-4}	4.19783×10^{-9}

Table 5: BB–BC on Levy Function

Dx	maxiter	$\|x_1 - 1\|$	$\|x_2 - 1\|$	f
0.13	30	2.90273×10^{-4}	1.01771×10^{-4}	9.51876×10^{-8}
0.03		3.61892×10^{-5}	8.09492×10^{-5}	1.87887×10^{-9}
0.13	50	3.78707×10^{-5}	7.89551×10^{-5}	1.99861×10^{-9}
0.03		2.49589×10^{-6}	6.04911×10^{-6}	9.27577×10^{-12}
0.011	50	8.46194×10^{-8}	1.70004×10^{-5}	1.80714×10^{-11}

Table 6: BB–BC on Ackley Function

Dx	maxiter	x_1	x_2	f
0.3	20	0.0	0.0	-3×10^{-39}

Table 7: BB–BC on Drop Wave Function

Dx	maxiter	x_1	x_2	$\|f + 1\|$
0.1	50	2.97430×10^{-7}	1.18631×10^{-7}	3.71692×10^{-12}
0.05		2.12861×10^{-8}	-4.99360×10^{-8}	1.06783×10^{-13}
0.1	80	1.15674×10^{-9}	9.45810×10^{-10}	0.0
0.05		2.64122×10^{-9}	1.69901×10^{-9}	3.32067×10^{-16}
0.1	100	1.35008×10^{-9}	-8.78500×10^{-11}	0.0

Table 8: BB–BC on Special test function

Dx	maxiter	x_1	x_2	f
0.3	50	2.01105×10^{-5}	2.91750×10^{-5}	3.43251×10^{-8}
0.3	100	2.02966×10^{-5}	9.45799×10^{-6}	8.90490×10^{-9}
0.03		3.67922×10^{-7}	1.36408×10^{-7}	2.56373×10^{-12}

این توابع به دقت بهتری دست یافتیم و نشان دادیم که با تعداد تکرارهای مختلف به هر میزان دقت از جواب می‌توان دست یافت.

Table 9: NBH on Schaffer Function

Dx	maxiter	x_1	x_2	f
0.13	50	1.04612×10^{-5}	2.10391×10^{-5}	0.55207×10^{-12}
0.033		1.39860×10^{-5}	$4.52032 - \times 10^{-5}$	2.16049×10^{-13}
0.033	70	4.64960×10^{-6}	4.76960×10^{-6}	4.32370×10^{-13}
0.003		0.0	0.0	0.0

Table 10: NBH on Rastrigin Function

Dx	maxiter	x_1	x_2	f
0.13	30	1.14857×10^{-9}	1.06002×10^{-9}	0.0
0.13	50	1.61971×10^{-9}	2.25987×10^{-9}	0.0
0.033		2.89574×10^{-10}	9.42731×10^{-10}	0.0

Table 11: NBH on Rosenbrock Function

Dx	maxiter	$\lvert x_1 - 1\rvert$	$\lvert x_2 - 1\rvert$	f
0.011	70	2.00783×10^{-4}	2.00783×10^{-4}	4.04034×10^{-8}
0.003	50	1.62378×10^{-4}	3.25719×10^{-4}	3.64544×10^{-8}
0.0015		9.69347×10^{-7}	2.52907×10^{-6}	1.75646×10^{-12}
0.003	70	6.47156×10^{-5}	1.29739×10^{-4}	4.19783×10^{-9}

Table 12: NBH on Levy Function

Dx	maxiter	$\lvert x_1 - 1\rvert$	$\lvert x_2 - 1\rvert$	f
0.13	30	2.62390×10^{-4}	6.96503×10^{-4}	1.07569×10^{-7}
0.03		1.56250×10^{-4}	1.56240×10^{-4}	2.89176×10^{-8}
0.13	50	1.59283×10^{-4}	4.43910×10^{-4}	4.407877×10^{-7}
0.03		5.13753×10^{-5}	1.17942×10^{-4}	3.82088×10^{-9}
0.011	50	1.77635×10^{-15}	1.44328×10^{-15}	0.0

Table 13: NBH on Ackley Function

Dx	maxiter	x_1	x_2	f
0.3	20	5.09947×10^{-13}	3.62088×10^{-12}	-1.03428×10^{-12}

ضمیمه

در این بخش توابع مورد استفاده در بخش پیاده سازی الگوریتم معرفی می‌شوند.

Table 14: NBH on Drop Wave Function

Dx	maxiter	x_1	x_2	$\lvert f+1\rvert$
0.1	30	2.93801×10^{-3}	1.27601×10^{-3}	0.0
0.05		1.83567×10^{-3}	-1.26745×10^{-3}	0.0
0.1	50	3.65739×10^{-3}	1.26745×10^{-3}	0.0
0.05		2.35840×10^{-4}	2.358401×10^{-4}	0.0
0.012	100	5.12000×10^{-5}	-5.12000×10^{-5}	0.0

Table 15: NBH on Special test function

Dx	maxiter	x_1	x_2	f
0.3	50	9.47674×10^{-4}	1.28000×10^{-3}	6.78840×10^{-5}
0.3	100	1.48078×10^{-4}	1.28000×10^{-3}	5.55330×10^{-6}
0.03		1.28000×10^{-4}	1.28000×10^{-4}	7.84900×10^{-7}

Table 16: BB–BC on Rastrigin function maxiter = 50

x and f	Dx = 0.3	Dx = 0.03	Dx = 0.03
x_1	5.53591×10^{-9}	1.31321×10^{-11}	1.21621×10^{-12}
x_2	5.26253×10^{-9}	1.31321×10^{-11}	1.21621×10^{-12}
x_3	5.34801×10^{-9}	1.31321×10^{-11}	1.21621×10^{-12}
x_4	5.26803×10^{-9}	1.31321×10^{-11}	1.21621×10^{-12}
x_5	3.51230×10^{-9}	1.31321×10^{-11}	1.21621×10^{-12}
f	2.44921×10^{-11}	0	0

Table 17: BB–BC on Rosen function maxiter = 50

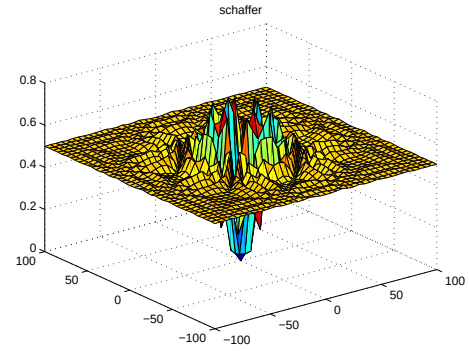
	Dx = 0.3	Dx = 0.03	Dx = 0.015
x_1	9.13827×10^{-7}	3.79400×10^{-8}	2.56272×10^{-11}
x_2	5.77634×10^{-7}	2.49001×10^{-8}	4.15762×10^{-10}
x_3	1.07602×10^{-6}	6.51311×10^{-9}	3.09483×10^{-10}
x_4	7.17250×10^{-7}	9.29732×10^{-9}	4.79768×10^{-10}
x_5	1.049320×10^{-6}	1.97429×10^{-8}	3.44209×10^{-10}
f	6.56916×10^{-5}	5.77007×10^{-8}	6.95724×10^{-10}

تابع Schaffer

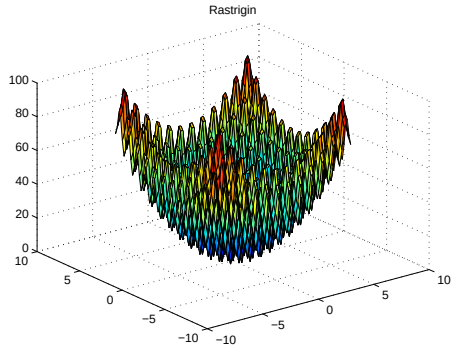
این تابع در حالت دوبعدی به‌صورت زیر تعریف می‌شود:

$$f(x_۱, x_۲) = ۰.۵ + \frac{\sin^۲(x_۱^۲ - x_۲^۲) - ۰.۵}{\left(۱ + ۰.۰۰۱(x_۱^۲ + x_۲^۲)\right)^۲}$$

تابع Schaffer یک تابع چندمتغیره است که برای به دام انداختن الگوریتم‌ها در مینیمم‌های محلی استفاده می‌شود. در حالت دوبعدی در ناحیه مربع $x_i \in [-۱۰۰, ۱۰۰] \quad i = ۱, ۲$ تنها مینیمم سراسری آن $f^* = ۰$ است که در $x^* = (۰, ۰)$ اتفاق می‌افتد. نمودار آن در شکل ۴ مشاهده می‌شود.



شکل ۴: Schaffer



شکل ۵: Rastrigin

تابع Rastrigin

تابع Rastrigin چندین مینیمم محلی دارد و نقاطی که مینیمم‌های محلی در آن اتفاق می‌افتند به طور منظم توزیع شده‌اند. ضابطه این تابع در حالت دوبعدی به‌صورت زیر است:

$$f(x_۱, x_۲) = ۲۰ + x_۱^۲ + x_۲^۲ - ۱۰\cos(۲\pi x_۱) - ۱۰\cos(۲\pi x_۲)$$

که در مربع $x_i \in [-۵.۱۲, ۵.۱۲] \quad i = ۱, ۲$ تنها مینیمم سراسری آن $f^* = ۰$ است که در $x^* = (۰, ۰)$ رخ می‌دهد. نمودار آن در شکل ۵ مشاهده می‌شود.

تابع Rosenbrock

تابع Rosenbrock یک تابع غیر محدب است که اغلب به‌عنوان یک تابع آزمون برای الگوریتم‌های بهینه‌سازی استفاده می‌شود که مینیمم سراسری آن در داخل یک ناحیه سهمی شکل قرار دارد که بعضی از الگوریتم‌های عددی به‌کندی به مینیمم آن همگرا می‌شوند. ضابطه آن در حالت دوبعد به‌صورت زیر است:

$$f(x_۱, x_۲) = (x_۱ - ۱)^۲ + ۱۰۰(x_۲ - x_۱^۲)^۲$$

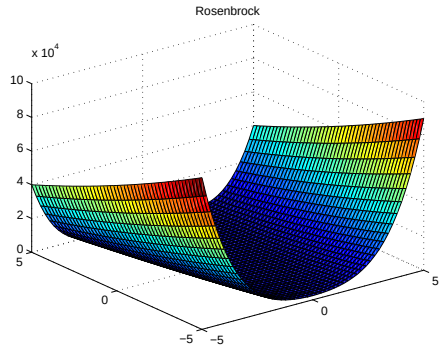
و مینیمم سراسری آن در $x = (۱, ۱)$ برابر $f^* = ۰$ است که در ناحیه مربع $x_i \in [-۱۰۰, ۱۰۰] \quad i = ۱, ۲$ اتفاق می‌افتد و نمودار تابع فوق در شکل ۶ نشان داده شده است.

تابع Levy

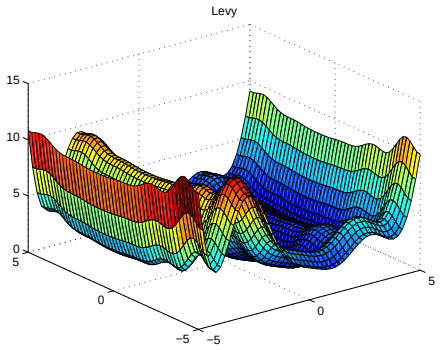
تابع Levy در ناحیه $[-۱۰, ۱۰]^۲$ با ضابطه زیر تعریف می‌شود:

$$f(x_۱, x_۲) = \sin^۲(\pi w_۱) + (w_۱ - ۱)^۲\left(۱ + ۱۰\sin^۲(\pi w_۱ + ۱)\right) + (w_۲ - ۱)^۲\left(۱ + \sin^۲(۲\pi w_۲)\right)$$

که در آن $w_i = ۱ + \frac{x_i - ۱}{۴}$ برای $i = ۱, ۲$ می‌باشد. تنها مینیمم سراسری این تابع برابر $f^* = ۰$ است که در $(x_۱, x_۲) = (۱, ۱)$ اتفاق می‌افتد. شکل ۷ نمودار تابع فوق را نشان می‌دهد.



شکل ۶: Rosenbrock



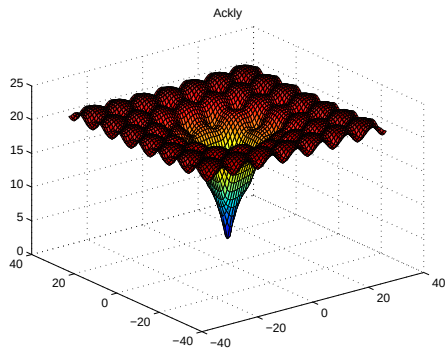
شکل ۷: Levy

تابع Ackley

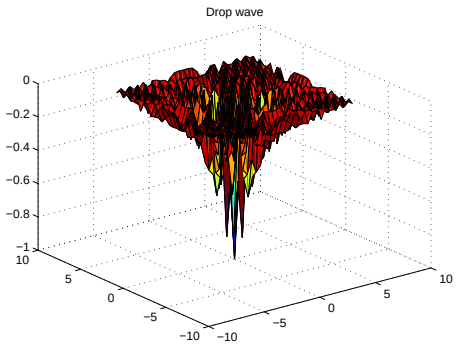
این تابع در حالت دوبعدی به‌صورت زیر تعریف می‌شود.

$$f(x_۱, x_۲) = -a\exp\left(-b\sqrt{\frac{۱}{۲}(x_۱^۲ + x_۲^۲)}\right) - \exp\left(\frac{۱}{۲}\big(\cos(cx_۱) + \cos(cx_۲)\big)\right) + a + \exp(۱)$$

که در آن $a = ۲۰$ ، $b = ۰.۲$ ، $c = ۲p$ است. تنها مینیمم سراسری این تابع $f^* = ۰$ در $(x_۱, x_۲) = (۰, ۰)$ می‌باشد که در ناحیه مربع شکل $x_i \in [-۳۲.۷۶۸, ۳۲.۷۶۸] \quad i = ۱, ۲$ اتفاق می‌افتد و نمودار این تابع در شکل ۸ نشان داده شده است. از این تابع به‌طور گسترده برای آزمایش الگوریتم‌های بهینه‌سازی استفاده می‌شود، به‌ویژه برای آزمایش این مطلب که الگوریتم در مینیمم‌های محلی به دام نیفتد.



شکل ۸: Ackley



شکل ۹: Drop–wave

تابع Drop–Wave

ضابطه این تابع در حالت دوبعدی به‌صورت زیر تعریف می‌شود:

$$f(x_۱, x_۲) = -\frac{۱ + \cos\left(۱۲\sqrt{x_۱^۲ + x_۲^۲}\right)}{۰.۵(x_۱^۲ + x_۲^۲) + ۲}$$

نمودار آن در ناحیه مربعی شکل $[-۵.۱۲, ۵.۱۲]^۲$ در شکل ۹ ارائه شده است. به‌سهولت می‌توان دریافت که تنها مینیمم سراسری آن برابر $f^* = -۱$ است که در نقطه $x = (۰, ۰)$ اتفاق می‌افتد.

تابع Special–Test–Function

این تابع در حالت دوبعدی به‌صورت زیر تعریف می‌شود:

$$f(x_۱, x_۲) = x_۱^۲ + ۲x_۲^۲ - ۰.۳\cos(۳\pi x_۱) - ۰.۴\cos(۴\pi x_۲) + ۰.۷$$

نمودار این تابع در شکل ۱۰ نشان داده شده است، مینیمم سراسری آن $(x_۱, x_۲) = (۰, ۰)$ و $f(x_۱, x_۲) = ۰$ است.

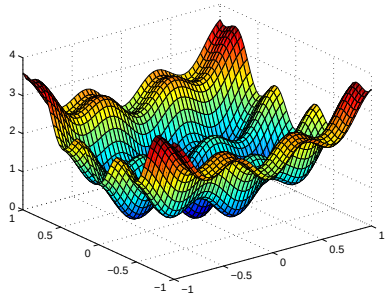


Figure 10: Special Test Function

فهرست منابع

[1] Biradar, Sh. Hote, Y. V. Saxena, S. *Reduced-order modeling of linear time invariant systems using big bang big crunch optimization and time moment matching method*, Applied Mathematical Modelling, **40**(15) (2016) 7225–7244.

[2] Coley, D. A. *An Introduction to genetic algorithm for scientist and engineering*, World Scientific Publishing Co, 1999.

[3] Dorigo, M. Maria Gambardella, L. *Ant colony system; A cooperative learning approach to the traveling salesman problem*, IEEE Transactions on Evolutionary Computation, **1**(1) (1997) 53–66.

[4] Dorigo, M. Maniezzo, V. Colorni, A. *The ant system: optimization by a colony of cooperating agents*, IEEE Transactions on Systems, Man, and Cybernetics–Part B, **26**(1) (1996) 29–41.

[5] Erol, O.K. Eksin, I. *A new optimization method: Big Bang-Big Crunch*, Journal of Advances in Engineering Software, **37** (2006) 106–111.

[6] Gutjahr, W. J. *Convergence Analysis of Metaheuristics*, Matheuristics: Hybridizing Metaheuristics and Mathematical Programming, Springer US, (2010) 159–187.

[7] Hatamlou, A. *Black hole: A new heuristic optimization approach for data clustering*, Information Sciences **222** (2013) 175–184.

[8] kennedy,J. Eberhart, R. *Particle swarm optimization*, IEEE International Conference on Neural Networks, **4** (1995) 1942–1948.

[9] Krikpatrick, S. Gelatt, C.D. Vecchi, M.P. *Optimization by simulated annealing*, Science, New Series, **220** (4598) (1983) 671–680.

[10] Liu, Y. Passino, K.M. *Biomimicry of social foraging bacteria for distributed optimization: models, principles and emergent behaviors*, Journal of Optimization Theory and Applications, **115**(3) (2002) 603–628.

[11] Madi, M. Markovi, D. Radovanovi, M. *Comparison of metaheuristic algorithms for solving machining optimization problems* , Facta universitatis - series: Mechanical Engineering, **11**(1) (2013) 29–44.

[12] Prayogo, D. Cheng, M.Y. Wu, Y. W. Arief Herdany, A. Prayogo, H. *Differential Big Bang –Big Crunch algorithm for construction–engineering design optimization*, Automation in Construction, **85** (2018) 290–304.

[13] Tarasewich, P. McMullen, P.R. *Swarm intelligence: power in numbers*, Communication of ACM, **45** (2002) 62–67.

[14] Zhang, J. Liu, K. Tan, Y. and He, X. *Random black hole particle swarm optimization and its application*, IEEE conference neural networks and signal processing, (2008) 359–365.

Two Phase Optimization Method Based on Meta heuristic Algorithms, Big Bang-Big Crunch and Black Hole

Negar Izadi[1, *] **, Mohammad Taghi Dastjerdi**[2]

[1,2] Department of Mathematics, Faculty of Science, Zanjan University, Zanjan, Iran

Received: 2021/5/24 Accepted: 2021/8/15

Communicated by: Mohammad Hadi Farahi

Abstract: This research proposes a two-phase algorithm whose main idea is based on meta heuristic algorithms, Big Bang and Black Hole. In the first phase of this algorithm, the artificial ants scan the reticulated rectangular region in parallel directions. The best points in the ant's navigations are used as starting points for the second stage of this algorithm. Big Bang and Black Hole algorithms, as an exploitation phase, try to investigate more accurate answers in the neighborhood of the starting points by reducing the neighborhood radius. Numerical examples confirm that this algorithm is capable to achieve an optimal solution with the desired accuracy and low computational costs.

Keywords: Optimization , Meta heuristic algorithm, Black Hole algorithm, Big Bang- Big Crunch method.

©2021 Shahid Chamran University of Ahvaz, Ahvaz, Iran. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution-NonComertial 4.0 International (CC BY-NC 4.0 license) (http://creativecommons.org/licenses/by-nc/4.0/).

[*] Corresponding author.
E-mail addresses: negarizadi@znu.ac.ir (N. Izadi), tdast@znu.ac.ir (M.T. Dastjerdi).