پژوهش نفت شماره ۱۲۹، خرداد و تیر ۱۴۰۲، صفحه ۴۲-۲۴

استفاده از الگوریتم بهبود یافته بهینه‌سازی باران جهت شبیه‌سازی حرکت دوغاب سیال در شکاف و ماتریکس

حجت نوری* و علی عالی انوری

گروه مهندسی معدن، دانشکده گروه مهندسی، دانشگاه کاشان، ایران

تاریخ دریافت: ۱۴۰۱/۰۸/۰۱ تاریخ پذیرش: ۱۴۰۱/۱۱/۲۳

چکیده

الگوریتم بهینه‌سازی باران (ROA) یک الگوریتم مبتنی بر جمعیت می‌باشد که با شبیه‌سازی حرکت قطرات باران به یافتن جواب بهینه برای مسائل پیچیده بهینه‌سازی می‌پردازد. این الگوریتم با حرکت دادن قطرات باران به‌سمت نقاط کمترین با توجه به قطر قطرات باران، قادر است با سرعت و دقت قابل قبولی نقاط کمترین یا بیشترین یک تابع یا مسأله بهینه‌سازی را پیدا کند. جهت بهبود قابلیت جستجو و اکتشاف در این الگوریتم، یک جستجوی تصادفی پیش از شروع به حل مسأله توسط این الگوریتم به این روش اضافه شد که این موضوع از بمب‌های خوشه‌ای الهام گرفته شده است. بدین ترتیب قبل از شروع به بهینه‌سازی توسط ROA، ابتدا نقاط تصادفی در اطراف این قطره باران انتخاب می‌شود و جستجو از نقطه‌ای شروع می‌شود که مقدار کمتری داشته باشد. به همین علت نام الگوریتم جدید به الگوریتم بهبود یافته بهینه‌سازی باران IROA تغییر یافت. کارایی بهینه‌ساز پیشنهادی از طریق بهینه‌سازی یک مسأله شبیه‌سازی در مهندسی معدن (شبیه‌سازی حرکت دوغاب سیمان در ماتریکس و شکاف) آزمایش شد و عملکرد آن با چندین الگوریتم فراابتکاری شناخته شده مقایسه شد. نتایج نشان می‌دهد که IROA قادر است با ارائه سرعت همگرایی سریع‌تر و همچنین کارآمدتر در مقایسه با سایر بهینه‌سازهای موفق منجر به‌دست‌یابی به به جواب‌های دقیق‌تر در مسائل پیچیده بهینه‌سازی شود.

کلمات کلیدی: الگوریتم بهینه‌سازی باران، الگوریتم فراابتکاری، دوغاب سیمان، سرعت همگرایی، ماتریکس

مقدمه

با توسعه کامپیوترهای قدرتمند، روش‌های جدید و نوینی برای حل برخی از مسائل پیچیده در حوزه مهندسی ارائه گردید. برخی از این روش‌ها که الگوریتم‌های فراابتکاری نامیده می‌شوند، مثل الگوریتم خفاش یا الگوریتم کلونی مورچگان از طبیعت الهام گرفته شده‌اند و یک پدیده طبیعی را جهت پیدا کردن بهترین جواب شبیه‌سازی می‌کنند. امروزه این الگوریتم‌ها قادرند روشی قدرتمند جهت حل بسیاری از مسائل پیچیده مهندسی را ارائه دهند

*مسؤول مکاتبات
آدرس الکترونیکی hnouri_ir@yahoo.com
شناسه دیجیتال: (DOI:10.22078/PR.2023.4953.3209)

هر چند ممکن است گاهی در کم‌ترین‌های نسبی با مشکل مواجه می شوند. اصطلاح الگوریتم‌های فراابتکاری یک مفهوم بسیار بالاتر نسبت به‌واژه مورد استفاده برای الگوریتم‌های ابتکاری را نشان می‌دهد و امروزه این الگوریتم‌ها برای حل طیف وسیعی از مسائل بهینه‌سازی استفاده می‌شوند. اخیراً الگوریتم‌های فراابتکاری زیادی برای حل مسائل غیرقابل حل به‌کار گرفته شده‌اند. جذابیت استفاده از این الگوریتم‌ها در این است که آن‌ها قادر هستند جواب بهینه یا حل مسائل بسیار پیچیده را در زمانی بسیار کوتاه به‌دست آورند [۱]. مسائل بهینه‌سازی که توجه روش‌های فراابتکاری را به‌خود جلب کرده‌اند واریانس زیادی دارند، از توابع یک تا چند هدفه، پیوسته تا گسسته و یا محدود تا غیرمحدود. حل این مسائل به دلیل رفتار پیچیده آن‌ها کار ساده‌ای نیست. الگوریتم‌های دقیق اکثراً چند جمله‌ای نیستند و همچنین ارائه بهترین راه حل‌ها توسط این الگوریتم ها معمولاً دارای زمان اجرای غیرعملی و یا الزامات محاسباتی بسیار زیاد هستند. الگوریتم‌های فراابتکاری یک راه حل عملی و کاربردی برای بسیاری از این مسائل ارائه می‌دهند و برای دست‌یابی به راه حل‌های تقریبی/ بهینه در زمان اجرای عملی، برای مسائل بهینه‌سازی بسیار سخت طراحی شده‌اند [۲ و ۳]. اکثر الگوریتم‌های هنرمندانه فراابتکاری ارائه شدند و به نام الگوریتم‌ها فراابتکاری کلاسیک شناخته می‌شوند که برخی از آن‌ها عبارتند از:

الگوریتم‌های ژنتیک (GA) [۴]، تکامل دیفرانسیل (DE) [۵]، برنامه‌نویسی ژنتیک (GP) [۶]، جستجوی تابو (TS) [۷]، روش جستجوی تصادفی تطبیقی هدفمند (GRASP) [۸]، جستجوی محلی تکراری (ILS) [۹]، الگوریتم جهش قورباغه مختلط (SFLA) [۱۰]، ذرات بهینه‌سازی ازدحام [۱۱]، جستجوی پراکنده (SS) [۱۲]، بهینه‌سازی کلونی مورچه ها (ACO) [۱۳]. علی‌رغم دست‌آوردهای الگوریتم‌های کلاسیک فراابتکاری، رویکردهای تکاملی جدید و بدیع نیز در دو دهه گذشته با موفقیت ظهور کردند. تحقیق در مورد الگوریتم‌های فراابتکاری در این دوران، تعداد زیادی از الگوریتم‌های فراابتکاری جدید را با الهام از فرآیندهای تکاملی یا رفتاری خاص معرفی می‌کند. در بسیاری از موارد، این موج جدید رویکردهای فراابتکاری، بهترین راه حل‌ها را برای برخی از مجموعه‌های توابع بنچمارک ارائه می‌دهد. برخی از این الگوریتم‌ها که به نام الگوریتم‌های فراابتکاری نسل جدید[1] شناخته می‌شوند، عبارتند از: الگوریتم بهینه‌سازی باران (ROA) [۱۴]، الگوریتم بهینه‌سازی نهنگ (WOA) [۱۵]، جستجوی ارگانیسم‌های همزیست (SOS) [۱۶]، الگوریتم گرگ خاکستری (GWA) [۱۷]، بهینه‌سازی عنکبوت اجتماعی (SSO) [۱۸]، آموزش بهینه‌سازی مبتنی بر یادگیری (TLBO) [۱۹]، الگوریتم کرم شب تاب (FA) [۲۰]، الگوریتم خفاش (BA) [۲۱]، الگوریتم جستجوی گرانشی (GSA)، تغذیه باکتری‌ها (BFO) [۲۲]، بهینه‌سازی مبتنی بر جغرافیای زیستی (BFO) [۲۳]، کلونی زنبورهای مصنوعی (ABC) [۲۴]. در این تحقیق یکی از جدیدترین الگوریتم‌های فراابتکاری در دسته اخیر یعنی الگوریتم بهینه‌سازی باران (ROA)، انتخاب شد و با ایجاد مقداری بهبود در روش یافتن جواب یک الگوریتم پر سرعت‌تر و با قابلیت بیشتر بوجود آمد که به نام الگوریتم بهبود یافته بهینه‌سازی باران[2] (IROA) اسم‌گذاری شد. این الگوریتم قادر است با سرعتی بسیار بیشتر و در زمانی کمتر نسبت به ROA و با تعداد NFE بسیار کمتر به جواب برخی از مسائل پیچیده بهینه‌سازی دست یابد. پس از معرفی این الگوریتم و مقایسه کارایی آن نسبت به برخی از الگوریتم‌های فراابتکاری قدرتمند، یک مسأله بسیار پیچیده شبیه‌سازی مربوط به حرکت دوغاب سیمان در شکاف و ماتریکس با استفاده از این الگوریتم بهبود یافته حل خواهد شد.

1. New Generation
2. Improved Rain Optimization Algorithm

مبانی الگوریتم بهینه‌سازی باران (ROA)

قبل از پرداخت به IROA بهتر است کمی به مبانی و روش کار الگوریتم باران پرداخته شود. همان‌گونه که از نام این الگوریتم مشخص است، حل مسأله توسط این الگوریتم از پدیده باران الهام گرفته شده است. قطرات باران در هنگام بارندگی به‌سمت نقاط کم ارتفاع‌تر حرکت می‌کنند و در نقاط پست و دارای ارتفاع کمتر جمع می‌شوند. در مسائل بهینه‌سازی نیز جواب یا جواب‌های مسأله در واقع نقاط بیشترین و کمترین هستند و معمولاً نقاطی که کمترین هزینه را در بر می‌گیرند به‌عنوان جواب مسأله انتخاب می‌شوند. به‌طور مثال از بین تمام مسیرهایی که می‌توان از شهر A به شهر B رسید، مسیری که کمترین فاصله را دارد به‌عنوان مسیر بهینه انتخاب می‌شود. بنابراین اگر یک مسأله بهینه‌سازی دو بعدی مشابه سطح زمین که دارای ناهمواری‌های زیادی می‌باشد، در نظر گرفته شود، نقاطی که قطرات باران در آن فرود می‌آیند را می‌توان به‌عنوان برخی از جواب‌های احتمالی این مسأله در نظر گرفت. بنابراین در الگوریتم بهینه‌سازی باران، سطح زمین به‌عنوان صورت مسأله و قطرات باران به‌عنوان جواب‌های احتمالی و شانسی مسأله در نظر گرفته می‌شود. این قطرات تحت سازوکارهای متفاوت که در ادامه توضیح داده می‌شود می‌توانند حرکت کنند، تغییر شکل دهند یا از بین بروند. به همین ترتیب، قطرات جدید (یا جواب‌های احتمالی جدید) تولید می‌شوند و در مسیر تکامل قرار می‌گیرند. این فرآیند تا جایی ادامه می‌یابد که یکی از قطرات به نقطه مورد نظر که همان جواب مسأله می‌باشد برسد.

هنگامی‌که بارندگی آغاز می‌شود، قطرات باران در نقاط مختلف زمین پراکنده می‌گردند و پس از آن دو اتفاق مهم ممکن است برای هر قطره باران رخ دهد: اگر قطره باران بر زمین مسطح فرود آید ممکن است قسمتی از آن توسط خاک جذب شود و یا قسمتی از آن توسط آفتاب بخار گردد. در این صورت با گذشت زمان اندازه قطره باران می‌تواند کاهش یابد. اگر این قطره باران بر سطحی شیب‌دار فرود آید، می‌تواند به‌سمت نقاط پایین‌تر حرکت کند و در همین حال ممکن است هنگام حرکت به‌سمت نقاط پایین‌تر با قطرات دیگر باران برخورد کند و قطر آن افزایش یابد و در عین حال تعداد جمعیت اولیه کاهش یابد. با حرکت قطرات باران به‌سمت نقاط پایین‌تر، احتمال برخورد قطرات باران با یکدیگر و تشکیل قطرات بزرگ‌تر بیشتر می‌شود و این کار تا جایی ادامه می‌یابد که هر قطره یا مجموعه قطرات بتوانند خود را به یک نقطه کمترین یا یک مانع در مسیر حرکت برسانند. در این صورت حرکت قطرات متصل شده به یکدیگر متوقف می‌شود و با ساکن شدن قطرات دوباره فرآیند جذب یا تبخیر آن‌ها شروع می‌شود. در حقیقت هر قطره یا مجموعه قطرات در نقاط کمترین نسبی تجمع می‌یابند و تشکیل دریاچه‌های فصلی را می‌دهند. با گذشت زمان و پس از توقف بارندگی این دریاچه‌ها کوچک و کوچک‌تر می‌شوند و مختصات آن نقطه کمترین (جواب مسأله) دقیق‌تر به‌دست خواهد آمد. نکته قابل توجه در الگوریتم باران این است که هر قطره باران تنها یک نقطه یا یکی از جواب‌ها نیست بلکه هر قطره باران دارای یک مختصات مربوط به مرکز قطره و شعاع قطره می‌شود. به این ترتیب هر قطره بلافاصله پس از رسیدن به زمین، یک سطح کوچک را تر می‌کند و در محدوده شعاع خود، می‌تواند اطراف خود را بررسی کند و در حقیقت دارای نوعی آگاهی به اطراف خود است. ROA قادر است تمام اکسترمم‌های نسبی یک تابع را پیدا کند و به این ترتیب نقطه قوت و نقطه تمایز این الگوریتم نسبت به بقیه الگوریتم‌ها در همین مسأله است، چون اکثر الگوریتم‌های بهینه‌سازی تنها به یک جواب می‌رسند ولی در مسائل عملی بسیار پیش می‌آید که به تمام نقاط اکسترمم نیاز باشد.

پیاده‌سازی ROA

در این قسمت به پیاده‌سازی الگوریتم باران براساس آنچه در قسمت قبل گفته شد، پرداخته می‌شود. هر جواب احتمالی مسأله را می‌توان با یک قطره باران (pop_i) مدل کرد. هر قطره باران دارای یک مختصات برحسب ابعاد مسأله و یک قطر می‌باشد که هر دوی آن‌ها یعنی مختصات نقطه مورد نظر و شعاع قطره باران به‌صورت تصادفی و در یک بازه مشخص انتخاب می‌شود. بسته به این‌که مسأله چقدر پیچیده باشد، تعداد قطرات اولیه باران (n_{pop}) که نشان‌دهنده جمعیت اولیه است به‌صورت اختیاری انتخاب خواهد شد. تعداد حدس‌های اولیه (قطرات اولیه باران) پارامتر بسیار مهمی است و می‌تواند به‌شدت بر فرآیند رسیدن به جواب موثر باشد. اگر جمعیت اولیه مقدار کمی باشد، ممکن است جواب مسأله پیدا نشود و اگر جمعیت اولیه زیاد باشد، زمان حل مسأله و در نتیجه هزینه حل مسأله به‌صورت قابل توجهی افزایش خواهد یافت. پس از آن برای هر قطره باران، مقدار تابع هدف محاسبه می‌شود. تابع هدف در حقیقت همان مسأله مورد نظر است که الگوریتم به دنبال حل آن است. پس از پیاده‌سازی الگوریتم به‌راحتی می‌توان تابع هدف را تغییر داد و به این ترتیب می‌توان هر تابع دلخواهی را به‌عنوان تابع هدف انتخاب کرد. این تابع می‌تواند ساده یا بسیار پیچیده باشد و با تغییر این تابع زمان و هزینه حل مسأله تغییر خواهد کرد.

اگر مسأله در فضای دو بعدی حل شود، مختصات محل قرارگیری هر قطره باران تنها از دو متغیر (x,y) تشکیل شده است اما هنگامی‌که مسأله در فضای n بعدی $[x_1,x_2,\ldots,x_n]$ حل می‌شود، محل هر قطره باران از n متغیر تشکیل شده است:

$$var = [var_1, var_2, \ldots, var_n] \quad (۱)$$

پس از انتخاب جمعیت اولیه و پیدا کردن مقدار تابع هدف به‌ازای هر یک از این نقاط، نقطه‌ای که دارای کمترین مقدار تابع هدف باشد به‌عنوان جواب اولیه مسأله انتخاب می‌شود. این جواب ممکن است با جواب اصلی مسأله فاصله زیادی داشته باشد ولی در هر صورت از بقیه جواب‌های حدس زده شده بهتر است. در ادامه در تعداد مشخصی تکرار سعی می‌شود جواب مسأله بهبود پیدا کند. برای این منظور در هر تکرار هر قطره باران، بسته به اندازه شعاع خود (R_n)، مقدار تابع هدف را در هر یک از کرانه‌های خود به‌ازای هر متغیر چک می‌کند. بنابراین اگر تابع هدف را با CostFunction نشان دهیم، برای تابعی با n متغیر، در هر تکرار برای هر قطره باران 2n مرتبه تابع هدف محاسبه می‌شود:

$$Cost_{var_1} = CostFunction(var_1 \pm R_1, var_2, \ldots, var_n) \quad (۲)$$

$$Cost_{var_2} = CostFunction(var_1, var_2 \pm R_2, \ldots, var_n) \quad (۳)$$

$$Cost_{var_n} = CostFunction(var_1, var_2, \ldots, var_n \pm R_n) \quad (۴)$$

برای درک بهتر فرض کنید که مسأله در فضای دو بعدی قرار دارد و در حقیقت قطره باران برروی یک سطح قرار گرفته است. اگر سطح مورد نظر که قطره روی آن قرار گرفته است، افقی نباشد و شعاع قطره به اندازه کافی کوچک باشد، مقدار تابع هدف در کرانه‌های قطره با مقدار تابع هدف در مرکز آن متفاوت خواهد بود. اگر مختصات مرکز قطره (x,y) باشد، مختصات کرانه‌ها در جهت x برابر با (x+R,y) و (x-R,y) خواهد بود. به همین ترتیب مختصات کرانه‌ها در جهت y برابر با (x, y+R) و (x,y-R) می‌باشد. مقدار تابع هدف در این چهار نقطه چک می‌شود و از بین این چهار نقطه، دو نقطه با ارتفاع کمتر (مقدار تابع هدف کمتر) نسبت به مرکز قطره مشخص می‌شود. به این ترتیب دو جهت برای حرکت قطره مشخص می‌شود که اگر قطره باران به‌اندازه شعاع خود در هر یک از این دو جهت حرکت کند، مقدار تابع هدف کاهش می‌یابد. نهایتاً برآیند این دو جهت به‌عنوان جهت

اصلی حرکت برای قطره تعریف می‌شود و مرکز قطره باران به اندازه شعاع قطره در این جهت حرکت می‌کند. به این ترتیب موقعیت قطره باران کمی بهبود پیدا می‌کند و به نقطه کمترین نزدیک‌تر می‌شود. جهت کاهش هزینه محاسبات، تازمانی‌که مقدار تابع هدف با حرکت قطره در جهت قبلی کاهش پیدا می‌کند، قطره در این جهت حرکت داده می‌شود. برای این کار، از مرکز قطره به اندازه شعاع قطره در جهت قبلی حرکت می‌کنیم، مقدار تابع هدف را در این نقطه محاسبه می‌کنیم و درصورتی‌که مقدار تابع هدف در نقطه جدید از مقدار تابع هدف در جهت قبلی کمتر بود، موقعیت قطره باران را به نقطه جدید تغییر می‌دهیم. همین عملیات برای تمام قطرات انجام می‌شود. به این ترتیب در هر تکرار موقعیت هر قطره ممکن است تغییرات زیادی داشته باشد و با سرعت زیاد به‌سمت نقاط کمترین حرکت کند. چگونگی انتخاب بهترین جهت برای حرکت هر قطره باران توسط تابع نشان داده شده در شکل ۱ نشان داده می‌شود: به این ترتیب هر قطره باران یک جهت به‌سمت نقاط پائین‌تر پیدا می‌کند و سپس قطره باران به‌اندازه شعاع خود در جهت مشخص شده در مرحله قبل حرکت می‌کند. نکته اینجاست تا زمانی که مقدار تابع هدف در حال کاهش می‌باشد، این قطره در جهت مشخص شده حرکت می‌کند (شکل ۲). درصورتی‌که قطره باران دیگر نتواند به نقاط پائین‌تر هدایت شود، قطر قطره باران با نسبت از قبل مشخص شده‌ای کاهش می‌یابد. به‌طور مثال اگر قطره باران در جهت مشخص شده حرکت کند و مقدار تابع هدف افزایش یابد، قطره بدون اینکه حرکت را انجام دهد کوچک‌تر می‌شود. این امر باعث می‌شود که در تکرار بعدی جستجو محلی قطره با دقت بیشتری انجام شود. اگر اندازه قطره باران از مقدار مشخصی کوچک‌تر شود این قطره حذف خواهد شد. قطرات نزدیک به هم، به یکدیگر متصل می‌شوند و تشکیل یک قطره باران را می‌دهند. این کار باعث می‌شود سرعت رسیدن به‌جواب به‌شدت افزایش یابد. به این منظور در هر تکرار پس اینکه موقعیت جدید هر قطره پیدا شد، فاصله هر قطره باران با تمام قطرات دیگر محاسبه می‌شود و قطراتی که از یک مقدار مشخص (مثلا دو برابر شعاع یک قطره) به هم نزدیک باشند به‌هم متصل می‌شوند و تشکیل یک قطره را می‌دهند.

```
for k=1:i
    pop_new=pop;
    pop_new.Position(k)=pop_new.Position(k)+Rᵢ;
    pop_new.Cost=cost_function(pop_new.Position);
    pop.Direction(k)=(pop.Cost-pop_new.Cost)/(pop.Position(k)-
pop_new.Position(k));
end
Return pop.Direction
```

شکل ۱ الگوریتم و تابع استفاده شده جهت هدایت یک نقطه به‌سمت نقطه کمترین

```
while(popm.Cost<pop.Cost)

    pop=popm;
    popm.Position=ChangePosition(pop.Position,pop.Radd,pop.Direction,nVar,
    VarMin, VarMax);
    popm.Cost=cost_function(popm.Position);
end
```

شکل ۲ الگوریتم و تابع استفاده شده برای حرکت دادن یک نقطه در یک جهت مشخص مادامیکه مقدار تابع هدف درحال کاهش است

همچنین در هر تکرار تعدادی از قطرات موجود که دارای مقدار تابع هدف بیشتری نسبت به بقیه قطرات هستند، حذف می‌شوند و به‌جای آن‌ها قطرات جدید جایگزین می‌شوند. تعداد قطرات حذف شده و یا تعداد قطرات جدید اضافه شده تاثیر زیادی بر سرعت رسیدن به جواب نهایی دارد. به این ترتیب این مکانیزم باعث می‌شود همیشه بارندگی ادامه داشته باشد و نقاط جدید کشف شوند. براساس آنچه در این قسمت گفته شد می‌توان الگوریتم مربوط به‌روش بهینه‌سازی باران را در شکل ۳ مشاهده نمود.

الگوریتم بهبود یافته باران IROA

در الگوریتم بهینه‌سازی باران ROA، همان‌گونه که در قسمت قبل توضیح داده شد، در ابتدا مکان قطرات باران به‌صورت رندوم انتخاب می‌شود و سپس هر قطره باران طی پروسه‌ای که شرح داده شد به‌سمت نقاط مینمم حرکت می‌کند و در نهایت با کوچک‌تر شدن قطره باران در هر تکرار دقت جواب افزایش می‌یابد. نکته منفی که در این الگوریتم وجود داشت و گاهی باعث عدم یافتن پاسخ با هزینه معقول می‌گشت این است که ممکن است قطر قطرات باران خیلی سریع و قبل از اینکه به منطقه اکسترمم مورد نظر برسیم بسیار کوچک شود و تعداد NFE [1] به‌شدت افزایش یابد. همچنین در بسیاری از موارد قطره باران باید مسیر بسیار طولانی را به‌صورت لاکپشتی تا منطقه مورد نظر طی کند. بنابراین برای حل این مشکل بر آن شدیم که قطره باران انتخاب شده به‌صورت رندم، ابتدا یک جستوی کلی انجام دهد و پس از آن به‌روش ROA به‌دنبال مقدار دقیق جواب بگردیم. به‌گفته دیگر، الگوریتم در هر تکرار برای هر قطره باران به دو روش جستجو را انجام خواهد داد:

۱- جستجوی کلی

۲- جستجوی موضعی

جستجوی محلی در واقع همان الگوریتم باران می‌باشد ولی جستجو کلی به‌صورت زیر انجام می‌شود:

هر قطره باران در ابتدا و قبل از اینکه تحت الگوریتم باران حرکت کند.

objective function f(X), $X=(x_1,x_2,\ldots,x_n)$

Input initial tuning parameters such as population number (nPop), maximum iteration (MaxIt), number of variables (nVar), the domain of variables ([VarMin, VarMax]), initial droplet's radius (InitR) and number of jointed droplets (size), rain speed (Speed) and Soil adsorption Constant (α).

Initialize droplets position, radius and size.

Evaluate each droplet with the objective function to obtain the cost of each droplet and sort population based on cost.

Main loop:
While (iteration number < MaxIt)
For (each droplet)
Change each variable x_i to x_i+R_i and x_i-R_i and evaluate the new position by the objective function.
If the new cost is smaller then the previous cost, accept a new position for x_i.

while (cost reduces)
move the droplet at the same direction with the same velocity,
reduce size of droplet depending on the soil adsorption properties
joint near droplets to each other, change size of new droplets
end while

end for
omit weak droplets depending on soil adsorption
generate new droplets depending on rain speed
end while

Sort populations based on cost.

Show results and visualizations.

شکل ۳ شبه کدهای مربوط به الگوریتم بهینه‌سازی باران

1. Number of Function Evaluation

به تعداد مشخصی قطره کوچک‌تر تبدیل شده و در فضای جستجو پراکنده می‌شود؛ درست مثل اینکه یک بمب خوشه‌ای در یک نقطه منفجر شود (شکل ۴).

شکل ۴ چگونگی تبدیل یک بمب بزرگ‌تر به بمب‌های کوچک‌تر در بمب‌های خوشه‌ای

در این حالت تعدادی بمب کوچک‌تر به‌صورت تصادفی در اطراف بمب اصلی پراکنده می‌شوند. اینکه هر قطره باران به چند قطره کوچک‌تر تقسیم شود و تا چه فاصله‌ای از قطره اصلی پراکنده شود، جزو پارامترهای ورودی شبکه خواهد بود. بدین ترتیب، در مرحله اول هر قطره باران به تعدادی قطره کوچک‌تر که در اطراف آن پراکنده شده‌اند تبدیل می‌شود و مقدار تابع هدف برای هر یک از قطرات کوچکتر محاسبه می‌شود و قطره‌ای که دارای کوچک‌ترین مقدار تابع هدف باشد، با قطره اصلی جایگزین می‌گردد. سپس این قطره تحت الگوریتم باران به حرکت در آمده و به یک نقطه بهتر می‌رسد. همین پروسه برای بقیه قطرات باران اتفاق می‌افتد تا به آخرین قطره برسیم. در تکرار بعدی دوباره همین سناریو انجام خواهد شد. یعنی در ابتدا یک جستجوی کلی و سپس یک جستجوی محلی انجام خواهد شد. شکل ۵ الگوریتم مربوط به IROA را بهتر نشان می‌دهد. بنابراین در هر تکرار با دو شعاع روبرو هستیم: شعاع جستجوی کلی (R_1) و شعاع قطره باران (R_2) که هر دوی آنها در هر تکرار می‌توانند کاهش یابند و یا آنکه ثابت بمانند. پیشنهاد می‌شود مقدار R_1 در اولین تکرار برابر با نصف دامنه جستجو انتخاب شود:

$$R_1 = \frac{\text{var}_{\max} - \text{var}_{\min}}{2} \quad (۵)$$

هرگاه الگوریتم در حین انجام جستجوی کلی برای یک قطره باران، در یک تکرار موفق به بهبود مکان قطره باران شود، می‌توان مقدار R_1 را مقداری کاهش داد تا در تکرار بعدی حتی شعاع جستجوی کلی تنگ‌تر شود و این امر باعث بهبود نتایج نهایی گردد.

objective function f(X), X=(x_1, x_2,...,x_n)

Input initial tuning parameters such as population number (nPop), maximum iteration (MaxIt), number of variables (nVar), the domain of variables ([VarMin, VarMax]), Initial radius of overall search(R_1), initial droplet's radius (R_2) and number of jointed droplets (size), rain speed (Speed) and Soil adsorption Constant(α).

Initialize droplets position, radius and size.

Evaluate each droplet with the objective function to obtain the cost of each droplet and sort population based on cost.

Main loop:
While (iteration number < MaxIt)
For (each droplet)
Find randomly *nCluster* droplets around this droplet.
Evaluate new droplets with the objective function.
Find the droplet with minimum cost.
Replace this droplet with the main droplet and reduce R_1 to $0.8R_1$
End For
For (each droplet)
Change each variable x_i to x_i+R_i and x_i-R_i and evaluate the new position by the objective function.
If the new cost is smaller than the previous cost, accept a new position for x_i.
while (cost reduces)
move the droplet at the same direction with the same velocity,
reduce size of droplet depending on the soil adsorption properties
joint near droplets to each other, change size of new droplets
end while
end for
omit weak droplets depending on soil adsorption
generate new droplets depending on rain speed
end while

Sort populations based on cost.

Show results and visualizations.

شکل ۵ شبه کدهای مربوط به الگوریتم بهبودیافته بهینه‌سازی باران

$$R_1 = \alpha R_1 \quad (۶)$$

در این تحقیق مقدار α برابر با ۰/۸ در نظر گرفته شده است.

اعتبارسنجی و مقایسه

پیاده‌سازی الگوریتم بهینه یافته باران با توجه به مطالب ارائه شده در سکشن قبل کار سختی نیست. کدهای مربوط به IROA توسط نرم‌افزار متلب پیاده‌سازی شد تا بتوان نتایج به‌دست آمده از این الگوریتم برای حل توابع بنچ[1] مارک مختلف با دیگر الگوریتم‌های مطرح در این زمینه از جمله الگوریتم ژنتیک، الگوریتم PSO، الگوریتم خفاش BA و نیز الگوریتم باران مقایسه گردد و توانایی حل مسأله و سرعت و قدرت این الگوریتم مشخص شود. بنابراین در این قسمت کارایی این الگوریتم از سه نقطه نظر مورد بررسی قرار خواهد گرفت:

۱- بررسی عملکرد IROA با استفاده از یک تابع معیار مشخص با جزئیات کامل و مقایسه عملکرد آن با الگوریتم باران

۲- بررسی عملکرد الگوریتم IROA در حل ۲۶ تابع معیار مهم و معروف و مقایسه نتایج به‌دست آمده با الگوریتم‌های دیگر

۳- بررسی عملکرد IROA در حل یک مسأله شبیه‌سازی مربوط به تزریق دوغاب سیمان در یک سیستم متشکل از ماتریکس و شکاف

بررسی عملکرد IROA در حل تابع Eggcreate

جهت بررسی نحوه علکرد IROA در پیدا کردن جواب‌های مسائل بهینه‌سازی، تابع Eggcreate به‌عنوان یک تابع معیار مهم انتخاب گردید. این تابع به‌صورت زیر تعریف می‌گردد:

$$f(x) = \sum_{i=1}^{D} x_i^2 + 25\sin^2(x_i) \quad (۷)$$

$$x \in [-5,5]^D$$

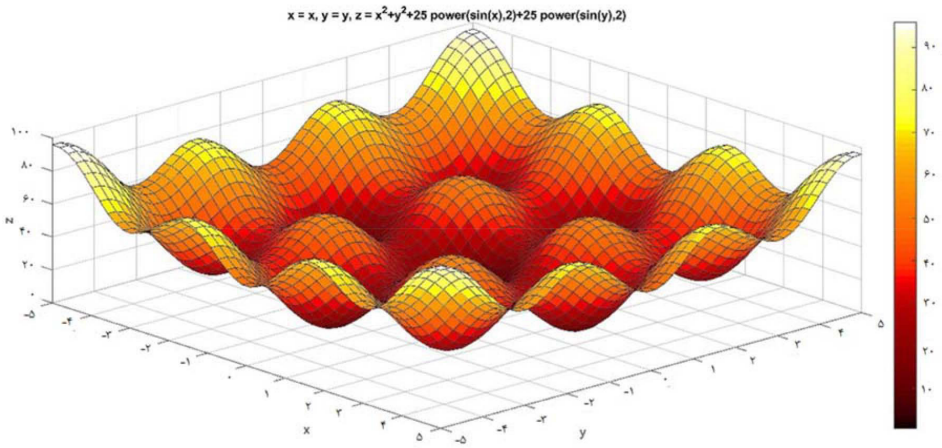
شکل ۶ تابع eggcrate را در سه‌بعد و در دامنه تعریف شده بالا نشان می‌دهد: این تابع دارای یک کمترین مطلق با مقدار صفر در $x_1=0,x_2=0$ می‌باشد. همچنین علاوه‌بر این نقطه ۸ کمترین نسبی دیگر مطابق جدول ۱ برای این تابع قابل تشخیص است: الگوریتم IROA با پارامترهای زیر برای حل این مسأله اجرا شد:

تعداد جمعیت اولیه (nPop): ۱۰۰

بعد متغیرها (D): ۳

مقدار کمترین فضای جستجو[2] (VarMin): ۵-

مقدار ماکسیمم فضای جستجو[3] (VarMax): ۵



شکل ۶ نمای سه‌بعدی تابع Eggcrete

1. Benchmark Function
2. Minimum of Search Space
3. Maximum of Search Space

جدول ۱ مقادیر و موقعیت کم‌ترین‌های نسبی در تابع Eggcrete

x *	f (x*)
[۳ و ۰]	۹/۵
[۰ و ۳-]	۹/۵
[۰ و ۳]	۹/۵
[۳- و ۰]	۹/۵
[۳ و ۳]	۹/۵
[۳- و ۳]	۹/۵
[۳ و ۳-]	۹/۵
[۳- و ۳-]	۹/۵

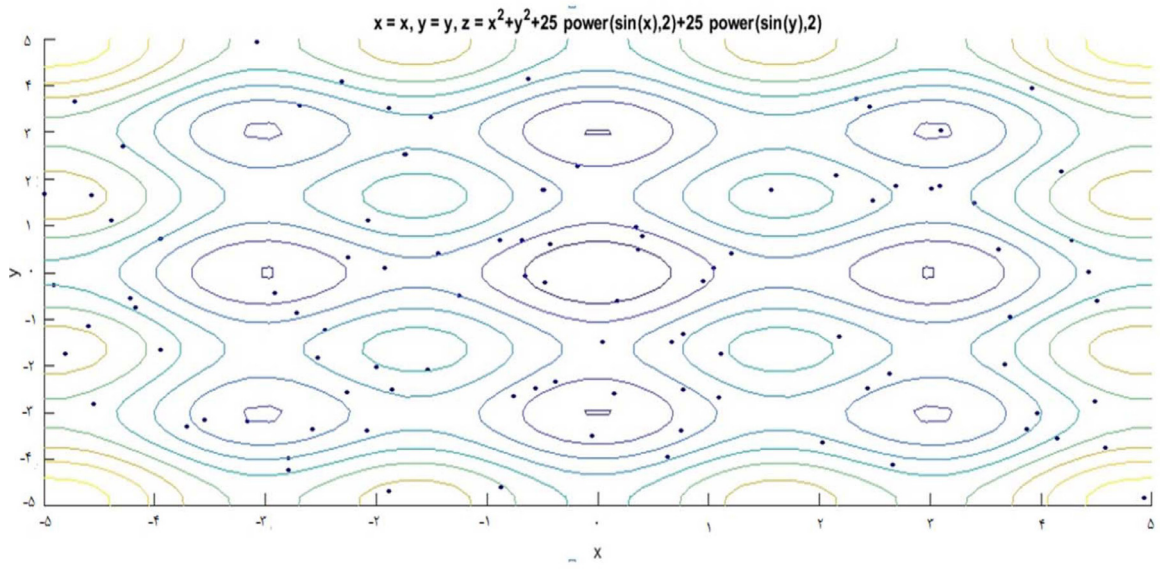
تعداد بیشترین تکرار (MaxIt): ۵۰۰

شعاع اولیه برای جستجوی کلی (R_1): ۵

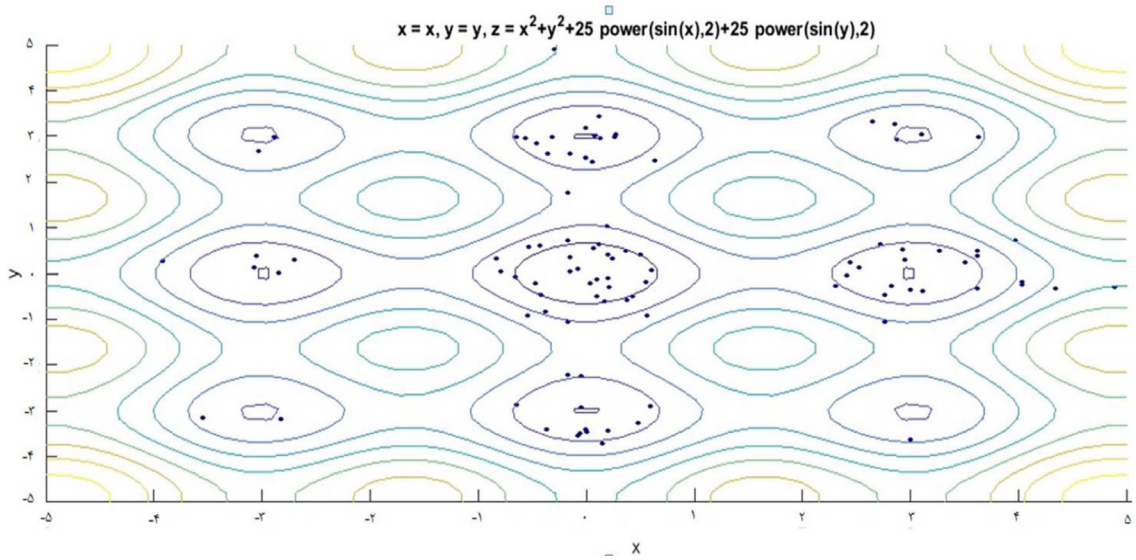
شعاع اولیه قطرات باران (R_2): ۰/۱

تعداد قطرات ایجاد شده به‌ازای هر قطره در جستجوی کلی (nCluster): ۲۰

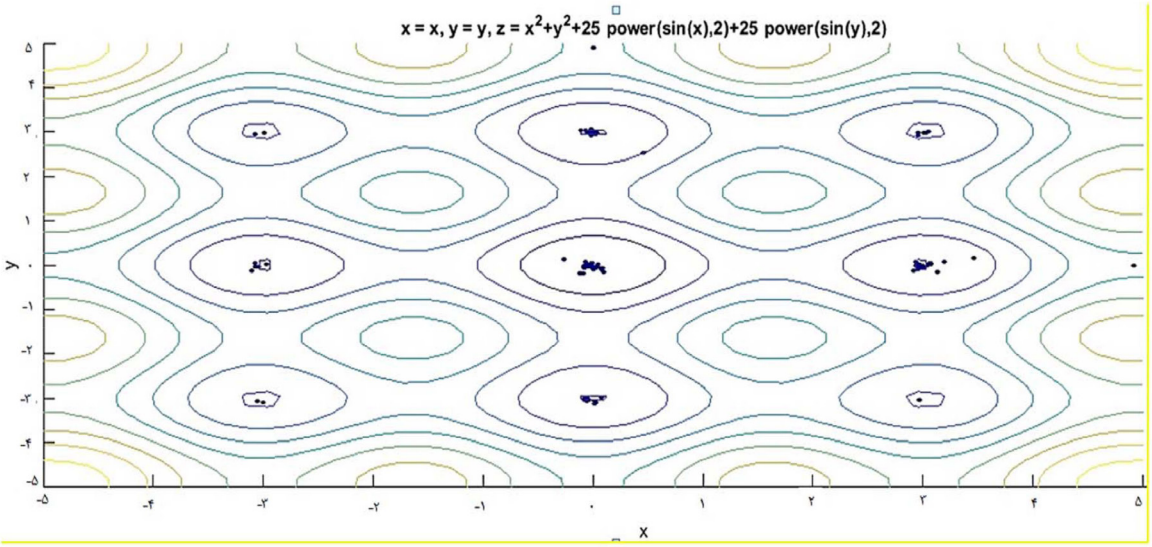
جمعیت اولیه ایجاد شده مطابق شکل ۷ روی فضای جستجوی پراکنده شده‌اند. پس از تنها یک بار انجام جستجوی کلی موقعیت اکثر نقاط بهبود یافته است و اکثر نقاط مطابق شکل ۸ به‌سمت نقاط کمترین حرکت کرده‌اند. در هر تکرار بعد از این‌که جستجوی کلی در فضای جستجو انجام شد، الگوریتم باران وارد عمل می‌شود. پس از اعمال الگوریتم باران روی نقاط در مرحله قبل و انجام جستجوی محلی موقعیت جمعیت اولیه در پایان تکرار اول به‌صورت شکل ۹ درخواهد آمد. بعد از ۱۰ تکرار الگوریتم موفق به پیدا کردن جواب مسأله شده است و مقدار خطای جواب به کمتر از $1e^{-1}$ رسیده است که این جواب با مقدار ارزیابی‌های انجام شده برای تابع هدف (NFE) برابر با ۲۴۰۲۸ به‌دست آمده است. وقتی همین مسأله با شرایط مشابه توسط الگوریتم باران حل شد، این الگوریتم موفق شد پس از ۴۹ تکرار و با مقدار NFE برابر با ۳۴۷۵۰ جواب مسأله را پیدا کند. این موضوع نشان می‌دهد که الگوریتم بهبود یافته باران از نظر تعداد تکرار حدود ۸۰٪ و از نظر NFE حدود ۳۰٪ نسبت به الگوریتم باران در حل این مسأله سریع‌تر عمل می‌کند. شکل ۱۰ چگونگی کاهش تابع هزینه به‌ازای تعداد ارزیابی های تابع هزینه (NFE) را نشان می‌دهد. هریک از خطوط عمودی خط چین نشان‌دهنده یک تکرار می‌باشد.



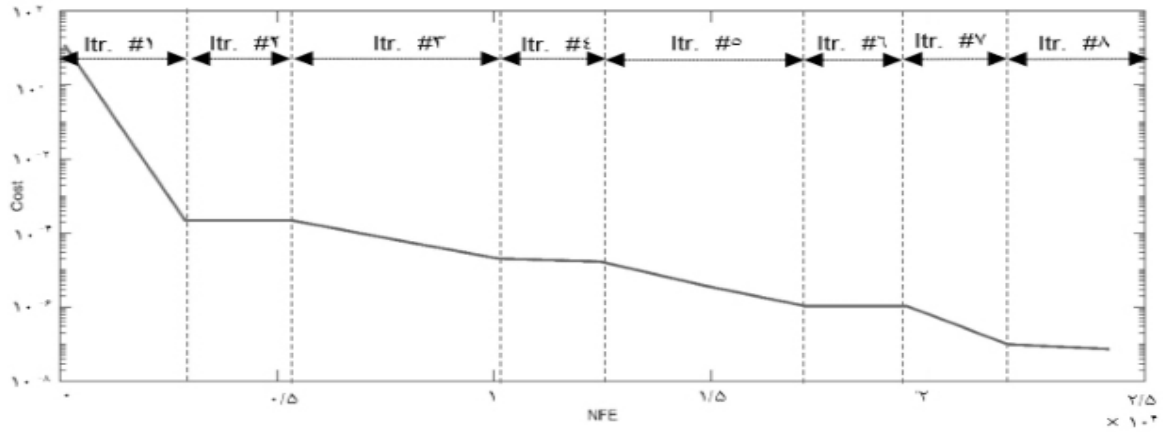
شکل ۷ موقعیت نقاط تصادفی اولیه انتخاب شده در روی فضای جستجو



شکل ۸ موقعیت نقاط تصادفی اولیه پس از تنها یک بار انجام جستجوی کلی (تا اینجا تنها نیمی از تکرار اول انجام شده است)



شکل ۹ موقعیت نقاط تصادفی اولیه پس از تنها یک تکرار (تا اینجا تنها در یک تکرار جستجوی کلی و جستجوی موضعی انجام شده است)



شکل ۱۰ چگونگی کاهش تابع هزینه به‌ازای تعداد ارزیابی‌های تابع هزینه (NFE)

همان‌گونه که مشاهده می‌شود مقدار بهترین جواب از ۱۲/۱۹۹۲ در ابتدای تکرار اول به ۲/۱۹۴ e-4 در انتهای تکرار اول رسیده است و در نهایت پس از ۸ تکرار بهترین جواب به‌دست آمده به ۷/۳۷۲۱ e-8 رسیده است. مقدار NFE در انتهای کار این الگوریتم برابر با ۲۴۰۲۸ بوده است. نکته مهم دیگر اینکه در هر تکرار تعداد NFE ممکن است با تکرارهای بعدی متفاوت باشد که همین امر نشان می‌دهد این الگوریتم به‌صورت هوشمند عمل می‌کند.

حل ۲۶ تابع معیار توسط الگوریتم بهینه یافته باران

در این قسمت ۲۶ تابع معیار مطابق با جدول ۲ جهت بررسی کارایی و عملکرد الگوریتم بهینه‌سازی بهبود یافته باران IROA انتخاب شد و کارایی این الگوریتم نسبت به چند الگوریتم مهم دیگر مثل الگوریتم ژنتیک (GA)، الگوریتم ازدحام ذرات (PSO)، الگوریتم خفاش (BA) و الگوریتم باران مقایسه شد. این توابع معیار قبلا توسط الگوریتم‌های ذکر شده توسط برخی از محققین حل شده‌اند و نتایج آن منتشر شده است [۱۴, ۱۶, ۲۵].

حل ۲۶ تابع معیار توسط الگوریتم بهینه یافته باران

در این قسمت ۲۶ تابع معیار مطابق با جدول ۲ جهت بررسی کارایی و عملکرد الگوریتم بهینه‌سازی بهبود یافته باران IROA انتخاب شد و کارایی این الگوریتم نسبت به چند الگوریتم مهم دیگر مثل الگوریتم ژنتیک (GA)، الگوریتم ازدحام ذرات (PSO)، الگوریتم خفاش (BA) و الگوریتم باران مقایسه شد. این توابع معیار قبلا توسط الگوریتم‌های ذکر شده توسط برخی از محققین حل شده‌اند و نتایج آن منتشر شده است [۱۴, ۱۶, ۲۵]. جهت حل این توابع از الگوریتم‌های ذکر شده، تعداد تابع ارزیابی بالاتر از ۵×۱۰^{۵} مجاز نیست. همچنین در حل این توابع هر عدد کمتر از ۱×۱۰^{-۱۲} برابر با صفر فرض می‌شود. همان‌گونه که قبلاً ذکر شد، پیاده‌سازی کدها توسط نرم‌افزار متلب و با استفاده از یک لب تاپ 8GHz انجام شد. تعداد جمعیت اولیه N_{pop} برای تمام الگوریتم‌ها برابر با ۵۰ در نظر گرفته شد. پارامترهای تنظیمی برای الگوریتم بهبود یافته باران و نیز بقیه الگوریتم‌های استفاده شده مطابق با جدول ۳ است. نتایج به‌دست آمده جهت حل توابع معیار توسط الگوریتم‌های یاد شده، در جدول ۳ آورده شده است. همان‌گونه که قابل مشاهده است، نمره و رتبه هر الگوریتم در دو ستون آخر جدول زیر آورده شده است. نتایج نشان می‌دهد که IROA, ROA, BA, PSO, GA به‌ترتیب بهترین عملکرد را در حل توابع معیار از خود نشان داده‌اند.

بررسی عملکرد IROA در حل یک مسأله شبیه‌سازی مربوط به تزریق دوغاب سیمان

بیان مسأله

امروزه به‌خاطر گسترش روز افزون نیاز به ساخت سازه‌های زیرزمینی، مشکلات و چالش‌های جدیدی در این زمینه ظهور و بروز یافته است که یکی از آن‌ها موضوع هجوم آب[1] به‌داخل تونل‌ها می‌باشد [۲۶]. مطالعات نشان می‌دهد مشکلات مهم و گوناگونی در اثر هجوم آب به‌داخل تونل‌ها می‌توانند ایجاد شود و به‌خاطر پیچیدگی و ناشناخته بودن این پدیده ممکن است حوادث دردناک و زیان‌های اقتصادی شدیدی به‌وجود آید. از طرف دیگر، ورودی آب به تونل‌ها می‌تواند باعث کاهش منابع آب زیرزمینی و کاهش سطح آب‌های زیرزمینی در منطقه مورد نظر و نهایتاً کاهش ذخیره آب در یک منطقه شود [۲۷]. علاوه‌بر این در حالت‌های خیلی شدید مشکلات زمین‌شناسی مثل ریزش دیواره‌های تونل و ترک‌های سطحی[2] در روی زمین ممکن است مشاهده شود. یکی از جدیدترین روش‌هایی که امروزه برای جلوگیری از ورود آب به معدن استفاده می‌شود، عملیات تزریق[3] می‌باشد. در این راستا تلاش‌های قابل توجهی جهت بهبود شرایط و روش اجرا، مواد مورد استفاده و دستگاه‌های مورد نیاز انجام شده است.

1. Water Inrush
2. Fissure
3. Grouting

جدول ۲ توابع معیار استفاده شده در این تحقیق و مقدار کمترین مطلق هریک از آن‌ها [۲۵].

No	Name	Range	D	Formulation	Min
۱	Rastrigin	[-۱۲/۵,۱۲/۵]	n	$f_1(x)=10n+\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)]$	۰
۲	De Jong (Sphere)	[-۱۲/۵,۱۲/۵]	n	$f_2(x)=\sum_{i=1}^{n}x_i^2$	۰
۳	Griewank	[-۶۰۰,۶۰۰]	n	$f_3(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	۰
۴	Beale	[-۵/۴,۵/۴]	۲	$f_4(x)=(1.5-x_1+x_1x_2)^2+(2.25-x_1+x_1x_2^2)^2+...$ $...(2.625-x_1+x_1x_2^3)^2$	۰
۵	Easom	[-۱۰۰,۱۰۰]	۲	$f_5(x)=-\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2-(x_2-\pi)^2)$	-۱
۶	Matyas	[-۱۰,۱۰]	۲	$f_6(x)=0.26(x_1^2+x_2^2)-0.48x_1x_2$	۰
۷	Bo-hachevsky1	[-۱۰۰,۱۰۰]	۲	$f_7(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)-0.4\cos(4\pi x_2)+0.7$	۰
۸	Booth	[-۱۰,۱۰]	۲	$f_8(x)=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2$	۰
۹	Michalewicz2	[π ,۰]	۲	$f_9(x)=-\sum_{i=1}^{D}\sin(x_i)(\sin(ix_i^2/\pi))^{20}$	-۸۰۱۳/۱
۱۰	Schaffer	[-۱۰۰,۱۰۰]	۲	$f_{10}(x)=0.5+\frac{\sin^2(\sqrt{x_1^2+x_2^2})-0.5}{(1+0.001(x_1^2+x_2^2))^2}$	۰
۱۱	Six Hump Camel Back	[-۵,۵]	۲	$f_{10}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	-۰۳۱۶۳/۱
۱۲	Boachevsky2	[-۱۰۰,۱۰۰]	۲	$f_{12}(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)(4\pi x_2)+0.3$	۰
۱۳	Boachevsky3	[-۱۰۰,۱۰۰]	۲	$f_{13}(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1+4\pi x_2)+0.3$	۰
۱۴	Shubert	[-۱۰,۱۰]	۲	$f_{14}(x)=(\sum_{i=1}^{5}i\cos(i+1)x_1+i)(\sum_{i=1}^{5}i\cos((i+1)x_2+i))$	-۷۳/۱۸۶
۱۵	Colville	[-۱۰,۱۰]	۴	$f_{15}(x)=100(x_1^2-x_2)^2+(x_1-1)^2+(x_3-1)^2+90(x_3^2-x_4)^2+..$ $..10.1(x_2-1)^2+(x_4-1)^2+19.8(x_2-1)(x_4-1)$	۰
۱۶	Michalewicz5	[π ,۰]	۵	$f_{16}(x)=-\sum_{i=1}^{D}\sin(x_i)(\sin(ix_i^2/\pi))^{20}$	-۶۸۷۷/۴
۱۷	Zakharov	[-۵,۱۰]	۱۰	$f_{17}(x)=\sum_{i=1}^{D}x_i^2+(\sum_{i=1}^{D}0.5ix_i^2)^2+(\sum_{i=1}^{D}0.5ix_i^2)^4$	۰
۱۸	Michale-wicz10	[π ,۰]	۱۰	$f_{18}(x)=-\sum_{i=1}^{D}\sin(x_i)(\sin(ix_i^2/\pi))^{20}$	-۶۶۰۲/۹
۱۹	Step	[-۱۲/۵,۱۲/۵]	۳۰	$f_{19}(x)=\sum_{i=1}^{D}(x_i+0.5)^2$	۰
۲۰	SumSquares	[-۱۰,۱۰]	۳۰	$f_{20}(x)=\sum_{i=1}^{D}ix_i^2$	۰

ادامه جدول ۲

۲۱	Quartic	[۱,۲۸,۱,۲۸-]	۳۰	$f_{21}(x)=\sum_{i=1}^{D} ix_i^4+\text{Rand}$	۰
۲۲	Schwefel 2.22	[۱۰,۱۰-]	۳۰	$f_{22}(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	۰
۲۳	Schwefel 1.2	[۱۰۰,۱۰۰-]	۳۰	$f_{23}(x)=\sum_{i=1}^{D}(\sum_{j=1}^{D} x_j)^2$	۰
۲۴	Rosenbrock	[۳۰,۳۰-]	۳۰	$f_{24}(x)=\sum_{i=1}^{D}100(x_{i+1}-x_i^2)^2+(x_i-1)^2$	۰
۲۵	Dixon-Price	[۱۰,۱۰-]	۳۰	$f_{25}(x)=(x_1-1)^2+\sum_{i=2}^{D} i(2x_i^2-x_i-1)^2$	۰
۲۶	Ackley	[۳۲,۳۲-]	۳۰	$f_{26}(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D}x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{D}\cos(2\pi x_i))+20+e$	۰

جدول ۳ مقایسه عملکرد الگوریتم‌های گوناگون روی توابع معیار مختلف

f	D	Min		GA	PSO	BA	ROA	IROA
			Mean	۵۲/۹۲۲۵۹ (۳)	۴۳/۹۷۷۱۳۶۹ (۲)	۰ (۱)	۰ (۱)	۰ (۱)
$f_1(x)$	۳۰	۰	SD	۴/۵۶۴۸۶	۱۱/۱۱۷۲۸۶۷۶	۰	۰	۰
			Mean	(۲)۳+۱E/۱۱	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)
$f_2(x)$	۳۰	۰	SD	۷۴/۲۱۴۴۷	۰	۰	۰	۰
			Mean	۱۰/۶۳۳۴۶ (۳)	۰/۰۱۷۳۹ (۲)	۰ (۱)	۰ (۱)	۰ (۱)
$f_3(x)$	۳۰	۰	SD	۱/۱۶۱۴۶	۰/۰۲۰۸۱	۰	۰	۰
			Mean	۰ (۱)	۰ (۱)	۱/۸۸ E-۰۵ (۲)	۰ (۱)	۰ (۱)
$f_4(x)$	۲	۰	SD	۰	۰	۹۴/۱ E-۰۵	۰	۰
			Mean	-۱ (۱)	-۱ (۱)	-۰/۹۹۹۹۴ (۲)	-۱ (۱)	-۱ (۱)
$f_5(x)$	۲	-۱	SD	۰	۰	۴/۵۰ E- ۰۵	۰	۰
			Mean	۰ (۱)	۰ (۱)	(۱) ۰	۰ (۱)	۰ (۱)
$f_6(x)$	۲	۰	SD	۰	۰	۰	۰	۰
			Mean	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)
$f_7(x)$	۲	۰	SD	۰	۰	۰	۰	۰
			Mean	۰ (۱)	۰ (۱)	۰/۰۰۰۵۳ (۲)	۰ (۱)	۰ (۱)
$f_8(x)$	۲	۰	SD	۰	۰	۰/۰۰۰۷۴	۰	۰
			Mean	-۱/۸۰۱۳ (۱)	-۱/۵۷۲۸۷ (۲)	-۱/۸۰۱۳ (۱)	-۱/۸۰۱۳ (۱)	-۱/۸۰۱۳
$f_9(x)$	۲	-۱/۸۰۱۳	SD	۰	۰/۱۱۹۸۶	۰	۰	۰

ادامه جدول ۳

			Mean	۰/۰۰۴۲۴ (۲)	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)
$f_{10}(x)$	۲	۰	SD	۰/۰۰۰۴۷۶	۰	۰	۰	۰
			Mean	-۱/۰۳۱۶	-۱/۰۳۱۶	-۱/۰۳۱۶	-۱/۰۳۱۶	-۱/۰۳۱۶
$f_{11}(x)$	۲	-۱/۰۳۱۶	SD	۰	۰	۰	۰	۰
			Mean	۰/۰۶۸۲۹ (۲)	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)
$f_{12}(x)$	۲	۰	SD	۰/۰۷۸۲۲	۰	۰	۰	۰
			Mean	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)
$f_{13}(x)$	۲	۰	SD	۰	۰	۰	۰	۰
			Mean	-۱۸۶/۷۳ (۱)	-۱۸۶/۷۳ (۱)	-۱۸۶/۷۳ (۱)	-۱۸۶/۷۳ (۱)	-۱۸۶/۷۳(۱)
$f_{14}(x)$	۲	-۱۸۶/۷۳	SD	۰	۰	۰	۰	۰
			Mean	۰/۰۱۴۹۴ (۴)	۰ (۱)	۱/۱۱۷۶ (۵)	۰/۰۰۰۵۳ (۲)	۰/۰۰۰۵۳(۲)
$f_{15}(x)$	۴	۰	SD	۰/۰۰۷۳۶	۰	۰/۴۶۶۲۳	۰/۰۰۰۳۲	۰/۰۰۰۳۲
			Mean	-۴/۶۴۴۸۳ (۲)	-۲/۴۹۰۸۷ (۳)	-۴/۶۸۷۷ (۱)	-۴/۶۸۷۷ (۱)	-۴/۶۸۷۷(۱)
$f_{16}(x)$	۵	-۴/۶۸۷۷	SD	۰/۰۰۹۷۸۵	۰/۰۲۵۶۹۵	۰	۰	۰
			Mean	۰/۰۱۳۳۶ (۲)	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)
$f_{17}(x)$	۱۰	۰	SD	۰/۰۰۴۵۳	۰	۰	۰	۰
			Mean	-۹/۶۶۰۲ (۳)	-۴/۰۰۷۱۸ (۴)	-۹/۶۶۰۲ (۱)	-۹/۶۶۰۲ (۱)	-۹/۶۶۰۲ (۱)
$f_{18}(x)$	۱۰	-۹/۶۶۰۲	SD	۰/۱۴۱۱۲	۰/۰۵۰۲۶۳	۰	۰	۰
			Mean	۱/۱۷ E+۰۳ (۳)	۰ (۱)	۵/۱۲۳۷ (۲)	۰ (۱)	۰ (۱)
$f_{19}(x)$	۳۰	۰	SD	۷۶/۵۶۱۴۵	۰	۰/۰۳۹۲۰۹	۰	۰
			Mean	۴۸/۱ E+۰۲ (۲)	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)
$f_{20}(x)$	۳۰	۰	SD	۱۲/۴۰۹۲۹	۰	۰	۰	۰
			Mean	۰/۱۸۰۷ (۴)	۰/۰۰۱۱۶ (۳)	۱/۷۲ E-۰۶ (۲)	۰ (۱)	۰ (۱)
$f_{21}(x)$	۳۰	۰	SD	۰/۰۲۷۱۲	۰/۰۰۰۲۸	۱/۵۸ E-۰۶	۰	۰
			Mean	۱۱/۰۲۱۴ (۳)	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)
$f_{22}(x)$	۳۰	۰	SD	۱/۳۸۶۸۶	۰	۰	۰	۰
			Mean	۷/۴۰ E +۰۳ (۲)	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)
$f_{23}(x)$	۳۰	۰	SD	۱/۱۴ E+۰۳	۰	۰	۰	۰
			Mean	۱/۹۶ E +۰۵ (۵)	۱۵/۰۸۸۶۱۷ (۳)	۲۸/۸۳۴ (۴)	۹/۴۵۳۶ (۲)	۵/۲۳۲۱ (۱)
$f_{24}(x)$	۳۰	۰	SD	۳/۸۵ E+۰۴	۲۴/۱۷۰۱۹۶	۰/۱۰۵۹۷	۳/۴۳۸۱	۲/۵۶۴
			Mean	۲۲/۱ E +۰۳ (۳)	۰/۶۶۶۶۷ (۲)	۰/۶۶۶۶۷ (۲)	۰ (۱)	۰ (۱)
$f_{25}(x)$	۳۰	۰	SD	۰۲+E ۲/۶۶	E-۰۸	۱/۱۶ E-۰۹	۰	۰
			Mean	۱۴/۶۷۱۷۸ (۳)	۰/۱۶۴۶۲ (۲)	۰ (۱)	۰ (۱)	۰ (۱)
$f_{26}(x)$	۳۰	۰	SD	۰/۱۷۸۱۴	۰/۴۹۳۸۷	۰	۰	۰
Score				۴۸	۳۶	۳۵	۲۸	۲۷
Rank Final				۵	۴	۳	۲	۱

یکی از نکاتی که می‌تواند دارای اهمیت ویژه‌ای باشد، این موضوع است که سیال تزریق شده به‌داخل شکاف‌ها، جهت جلوگیری از ورود آب به‌داخل تونل، قادر است تا چه فاصله‌ای حرکت کند. البته این مسأله وابستگی زیادی به دبی سیال تزریقی و نیز نوع سیال تزریقی به‌داخل شکاف دارد. در ادامه با استفاده از الگوریتم IROA به حل این مسأله پیچیده پرداخته خواهد شد.

اصول شبیه‌سازی حرکت سیال در محیط متخلخل و شکاف

شبیه‌سازی حرکت سیال در یک محیط متخلخل و به‌صورت مشابه یک شکاف با ترکیب سه رابطه مهم یعنی رابطه بقای جرم، رابطه حرکت سیال در محیط متخلخل و نیز رابطه حالت قابل انجام می‌باشد. سیال مورد استفاده جهت تزریق در یک شکاف یک جریان تک فاز از یک سیال تراکم‌ناپذیر می‌باشد که ویسکوزیته این سیال نسبت به زمان متغیر می‌باشد.

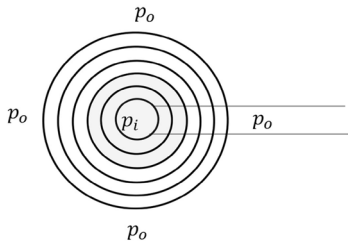
شبیه‌سازی یک شکاف

برای شبیه‌سازی حرکت یک سیال تراکم‌ناپذیر در یک شکاف دو حالت مختلف را می‌توان در نظر گرفت:

۱- حالتی که دیواره‌های دو طرف شکاف دارای تراوایی صفر باشند و سیال نتواند وارد ماتریکس اطراف شکاف شود.

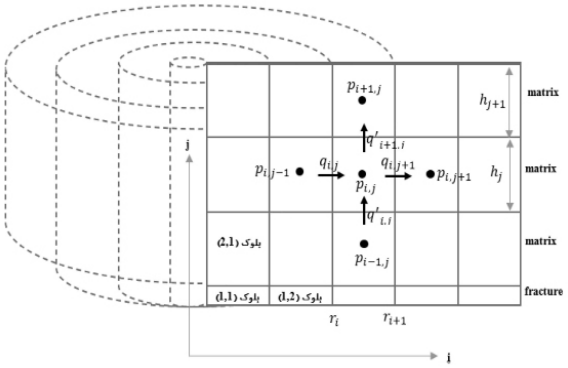
۲- حالتی که دیواره‌های دو طرف شکاف یک محیط متخلخل و دارای تراوایی مشخص باشد و سیال تزریق وارد دیواره‌ها شود.

شبیه‌سازی اول در واقع یک حالت خاص از شبیه‌سازی کلی‌تر دوم است و برای جامع‌تر بودن شبیه‌سازی، در ادامه این حالت بحث خواهد شد. شکل ۱۱ نمای یک شکاف را وقتی سیال از یک گمانه به‌داخل آن تزریق می‌شود از بالا نشان می‌دهد: سیال با حرکت به‌داخل شکاف به‌صورت شعاعی در شکاف حرکت می‌کند و به پیش می‌رود.



شکل ۱۱ شکل یک شکاف وقتی یک گمانه در جهت عمود بر آن حفر شود و حرکت شعاعی سیال تزریقی به‌داخل شکاف

هر یک از دوایر یک بلوک را نشان می‌دهد که در طول شبیه‌سازی باید مشخصات آن را به‌صورت دقیق تعیین کرد. اگر نمای عمودی برش نشان داده شده در شکل ۱۲ را نگاه کنیم، بلوک‌های مورد استفاده در شبیه‌سازی بهتر مشخص می‌شوند:



شکل ۱۲ بلوک بندی مورد استفاده در شبیه‌سازی یک شکاف و ماتریس اطراف آن که هریک از بلوک‌ها به‌صورت پوسته‌های استوانه‌ای تو در تو خواهند بود

همان‌گونه که در مقطع عمودی شکل ۱۲ دیده می‌شود، منطقه شبیه‌سازی به تعدادی بلوک تقسیم شده است که بلوک‌ها در i ردیف و j ستون جای داده شده‌اند. هر بلوک به شکل پوسته‌ای از استوانه است و سیال می‌تواند به‌صورت افقی و عمودی بین بلوک‌ها جابه‌جا شود. حرکت افقی سیال تحت مدل شعاعی و حرکت عمودی سیال تحت مدل خطی انجام می‌شود. به مرکز هر بلوک یک فشار تخصیص داده شده است که برای بلوک سطر i و ستون j با $p_{i,j}$ نشان داده می‌شود.

از طرف دیگر سیال ورودی به یک بلوک در جهت عمودی با $q_{i,j}$ و سیال خروجی از یک بلوک در جهت عمودی با q'_{i+1j} نشان داده می‌شود. جهت سادگی در شبیه‌سازی یک طرف شکاف را ناتراوا فرض می‌کنیم و تراوایی طرف دوم را دو برابر در نظر می‌گیریم. مشخصات هر بلوک از جمله ارتفاع بلوک، شعاع داخلی بلوک، شعاع خارجی بلوک، تراوایی عمودی بلوک، تراوایی شعاعی بلوک و ویسکوزیته سیال داخل هر بلوک از قبل مشخص است و هر یک توسط یک ماتریس دو بعدی i در j ذخیره می‌شود. نکته خیلی مهم در این شبیه‌سازی این است که ویسکوزیته سیال تزریقی تابعی از زمان است و با گذشت زمان بیشتر می‌شود.

برای یک بلوک مشخص (i,j) داریم:

$$q_{i,j} = q_{i+1,j} + q'_{i,j} - q'_{i,j+1} = 0 \qquad (۸)$$

این شرط باید برای تمامی بلوک‌ها بررسی شود و با حل هم‌زمان دستگاه روابط به‌دست آمده برای تمامی بلوک‌ها می‌توان فشار را در هر بلوک محاسبه کرد. برای مثال برای حل شبیه‌سازی یک شکاف با تعداد بلوک‌های n×n باید دستگاه روابط زیر حل شود:

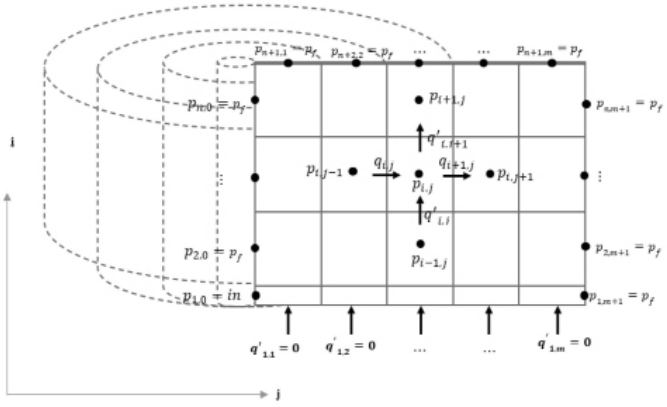
$$\begin{cases} q_{1,1} - q_{2,1} + q'_{1,1} - q'_{1,2} = 0 \\ q_{1,2} - q_{2,2} + q'_{1,2} - q'_{1,3} = 0 \\ \vdots \\ q_{i,j} - q_{i+1,j} + q'_{i,j} - q'_{i,j+1} = 0 \\ \vdots \\ q_{n,n} - q_{n+1,n} + q'_{n,n} - q'_{n,n+1} = 0 \end{cases} \qquad (۹)$$

شرایط مرزی

شرایط مرزی تاثیر زیادی در حل مسأله فوق دارد. در این کار فرض بر این است که یک گمانه به‌صورت عمودی از یک شکاف عبور کرده است و تا حد مورد نیاز به سنگ اطراف شکاف نفوذ کرده است. بنابراین فشار سیال وردی به شکاف و ماتریکس برابر است با pin که مقدار آن به فشار تزریق پمپ مورد استفاده بستگی دارد و البته حد بالایی آن به فشار شکست در سنگ دیواره معدن بستگی دارد. با توجه به اینکه شکاف را از یک طرف شبیه‌سازی نمودیم، دبی ورودی از سمت ماتریس به‌داخل شکاف از یک طرف صفر در نظر گرفته می‌شود (در شکل زیر دبی ورودی از پایین به‌داخل شکاف صفر در نظر گرفته شده است). فشار در دوطرف دیگر مدل برابر با فشار آب موجود در سازند می‌باشد. شکل ۱۳ شرایط مرزی حل مسأله را بهتر نشان می‌دهد:

محاسبه فشار در هر بلوک

همان‌گونه که قبلا گفته شد، برای محاسبه فشار در هر بلوک با در نظر گرفته شرایط مرزی و با حل دستگاه روابط داده شده در قسمت قبل می‌توان فشار در هر بلوک را محاسبه نمود. برای حل دستگاه روابط در این کار سعی می‌کنیم از یک الگوریتم بهینه‌سازی فراابتکاری جدید به نام الگوریتم بهبودیافته بهینه‌سازی باران استفاده نماییم.

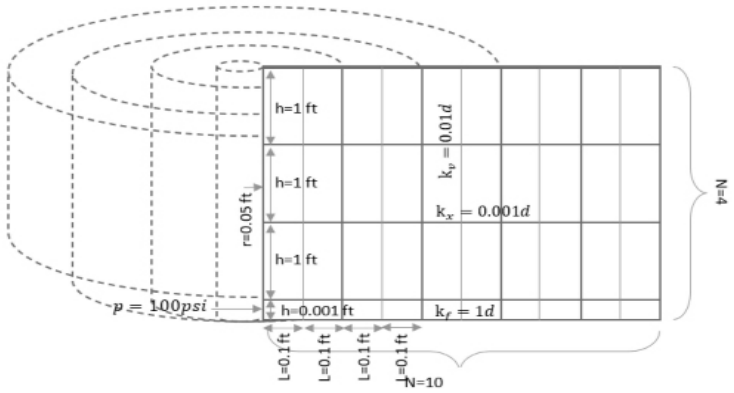


شکل ۱۳ شرایط مرزی برای حل مسأله شبیه‌سازی

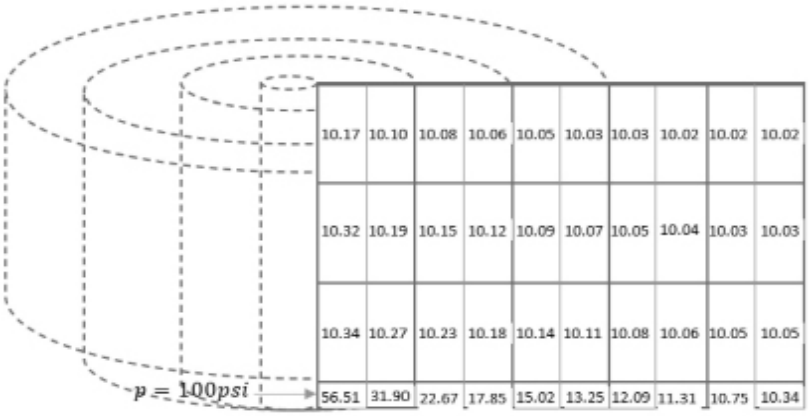
در این روش با حدس زدن فشار در هر بلوک و بررسی شرط مجموع دبی برابر با صفر در هر بلوک مقدار فشار بلوک را به‌دست آوریم.

بحث

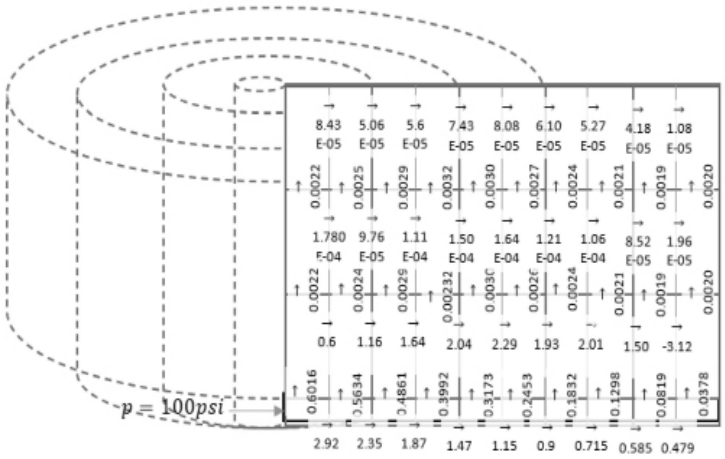
جهت انجام شبیه‌سازی در این‌کار، از یک شبکه‌بندی ۱۰ در ۴ استفاده شد به‌طوری‌که تعداد بلوک‌های استوانه‌ای تو در تو ۱۰ عدد بود و هر استوانه خود به ۴ بلوک تقسیم شد. قطر شکاف برابر با ۰/۰۰۱ ft و ارتفاع بلوک‌های عمودی برابر با یک فوت انتخاب گردید. همچنین ضخامت هر بلوک استوانه‌ای برابر با ۰/۱ ft در نظر گرفته شد. در این مدل تراوایی شکاف برابر با ۱۰ دارسی و تراوایی افقی و عمودی هر بلوک به‌ترتیب ۰/۰۰۱ و ۰/۰۱ D در نظر گرفته شد. از طرف دیگر فشار سیال تزریقی به‌داخل شکاف برابر با ۱۰۰ psi و فشار اولیه آب موجود در بلوک‌ها قبل از تزریق برابر با ۱۰ psi در نظر گرفته شد. شکل ۱۴ این سایزبندی را بهتر نشان می‌دهد. جهت ساده سازی این مدل تراوایی یک طرف شکاف (پایین شکاف) برابر با صفر و تراوایی عمودی بلوک‌ها دو برابر در نظر گرفته شد. مدل ایجاد شده در این کار با استفاده از الگوریتم بهینه‌سازی باران حل شد و فشار در مرکز هر بلوک محاسبه شد. برای این کار ویسکوزیته سیال ۱ cp در نظر گرفته شد. مدل ایجاد شده در این کار با استفاده از الگوریتم بهینه‌سازی باران حل شد و فشار در مرکز هر بلوک محاسبه شد. برای این کار ویسکوزیته سیال ۱ cp در نظر گرفته شد. فشار به‌دست آمده در هر بلوک پس از ۱۰۰ تکرار توسط الگوریتم باران و با انتخاب جمعیت اولیه برابر با ۵۰ در شکل ۱۵ دیده می‌شود. دبی افقی و دبی عمودی در بین بلوک‌ها در شکل ۱۶ داده شده است.



شکل ۱۴ خصوصیات هندسی و ژئوفیزیکی مدل ایجاد شده جهت انجام شبیه‌سازی



شکل ۱۵ فشار به‌دست آمده در هر بلوک هنگامی‌که سیالی با ویسکوزیته ۱ cp و فشار ۱۰۰ psi به‌داخل شکاف تزریق می‌گردد



شکل ۱۶ دبی افقی و عمودی بین بلوک‌ها هنگامی‌که سیالی با ویسکوزیته cp ۱ و فشار psi ۱۰۰ به‌داخل شکاف تزریق می‌گردد

نتیجه‌گیری

در این تحقیق یک الگوریتم فرابتکاری جدید به نام الگوریتم بهبود یافته بهینه‌سازی باران که از پدیده بارندگی الهام گرفته شده بود، معرفی گردید. این الگوریتم در واقع یک ویرایش جدید از الگوریتم باران است و با ایجاد تغییراتی در این الگوریتم به وجود آمد. این الگوریتم برای حل چندین تابع معیار گوناگون و یک مسأله شبیه‌سازی در حوزه مهندسی معدن استفاده شد. همچنین عملکرد این الگوریتم با الگوریتم‌های دیگر مثل الگوریتم ژنتیک، الگوریتم ازدحام ذرات، الگوریتم خفاش و نیز الگوریتم باران مقایسه شد و نتایج زیر به‌دست آمد:

الگوریتم بهینه‌سازی بهبود یافته باران دارای سرعت بسیار بالاتری نسبت به الگوریتم بهینه‌سازی باران می‌باشد و قادر است با تعداد تکرارهای کمتر و نیز تعداد ارزیابی‌های کمتر تابع هدف به جواب مسأله برسد. در همین راستا جهت حل تابع معیار Eggcrete، الگوریتم بهبود یافته باران از نظر تعداد تکرار حدود ۸۰٪ و از نظر NFE حدود ۳۰٪ نسبت به الگوریتم باران در حل این مسأله سریع‌تر عمل کرد. الگوریتم بهبود یافته باران، در حل توابع معیار پیچیده قادر بود عملکرد بهتری نسبت به الگوریتم ژنتیک، الگوریتم ازدحام ذرات، الگوریتم خفاش و نیز الگوریتم باران داشته باشد. الگوریتم بهبود یافته باران برای حل یک مسأله شبیه‌سازی پیچیده در حوزه مهندسی معدن استفاده شد و قادر بود با دقتی بسیار بالا و با سرعت خوب جواب مسأله را پیدا کند.

مراجع

[1]. Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., & Cosar, A. (2019). A survey on new generation metaheuristic algorithms, Computers & Industrial Engineering, 137, 106040, 2019/11/01/ 2019, doi: https://doi.org/10.1016/j.cie.2019.106040.

[2]. Magnusson, J., & Nilsson, J. (2018). Project matching application framework using metaheuristic algorithms, 1st edition, Chalmers University of Technology, Gothenburg, Sweden, 1-83.

[۳]. شکیبا س. و دولتی ارده‌جانی ف. (۲۰۲۳). استفاده از روش‌های بهینه‌یابی فرابتکاری جستجوی گرانشی، ازدحام ذرات و ترکیب آن‌ها در مدل‌سازی شبکه شکستگی، پژوهش نفت، ۳۳(۱۴۰۲-۱): ۱۰۷-۱۰۰، doi: .10.22078/pr.2022.4960.3210

[4]. Goldberg, D. E. (1989). Genetic algorithms in search, Optimization, and MachineLearning, doi.org/10.11517/jjsai.7.1_168.

[5]. Storn, R., & Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimi zation over continuous spaces, journal of global optimization, 11(4): 341-359, 1997/12/01 1997, doi: 10.1023/A:1008202821328.
[6]. Banzhaf, W., Nordin, P., Keller, R. E., & Francone, F. D. (1998). Genetic programming: an introduction, Morgan Kaufmann Publishers San Francisco.
[7]. F. Glover and M. Laguna, (1998). Tabu Search, in Handbook of Combinatorial Optimization, 1–3, D.-Z. Du and P. M. Pardalos Eds. Boston, MA: Springer US, 2093-2229.
[8]. Marques-Silva, J. P., & Sakallah, K. A. (1999). A search algorithm for propositional satisfiability, IEEE Transactions on Computers, 48(5): 506-521, 1999, doi: 10.1109/12.769433.
[9]. Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated Local Search, in Handbook of Metaheuristics, F. Glover and G. A. Kochenberger Eds. Boston, MA: Springer US, 320-353.
[10]. Eusuff, M. M., & Lansey, K. E. (2003). Optimization of water distribution network design using the shuffled frog leaping algorithm, Journal of Water Resources planning and management, 129(3): 210-225, doi.org/10.1061/(ASCE)0733-9496(2003)129:3(21.
[11]. Wei, Y., & Qiqiang, L. (2004). Survey on particle swarm optimization algorithm, Engineering Science, 5(5): 87-94.
[12]. Martí, R., Laguna, M., & Glover, F. (2006). Principles of scatter search, European Journal of Operational Research, 169(2): 359-372, doi: https://doi.org/10.1016/j.ejor.2004.08.004.
[13]. Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization, IEEE computational intelligence magazine, 1(4): 28-39, doi: 10.1109/MCI.2006.329691.
[14]. Moazzeni, A. R., & Khamehchi, E. (2020). Rain optimization algorithm (ROA): A new metaheuristic method for drilling optimization solutions, Journal of Petroleum Science and Engineering, 195, 107512, doi.org/10.1016/j.petrol.2020.107512.
[15]. Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm, Advances in Engineering Software, 95, 51-67, doi: https://doi.org/10.1016/j.advengsoft.2016.01.008.
[16]. Cheng, M. Y., & Prayogo, D. (2014). Symbiotic organisms search: A new metaheuristic optimization algorithm, Computers & Structures, 139, 98-112, doi: https://doi.org/10.1016/j.compstruc.2014.03.007.
[17]. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey Wolf Optimizer, Advances in Engineering Software, 69, 46-61, doi: https://doi.org/10.1016/j.advengsoft.2013.12.007.
[18]. Cuevas, E., Cienfuegos, M., Zaldívar, D., & Pérez-Cisneros, M. (2013). A swarm optimization algorithm inspired in the behavior of the social-spider, Expert Systems with Applications, 40, (16): doi: https://doi.org/10.1016/j.eswa.2013.05.041.
[19]. Rao, R. V., Savsani, V. J., & Vakharia, D. P. (2011). Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems, Computer-Aided Design, 43(3): 303-315, doi: https://doi.org/10.1016/j.cad.2010.12.015.
[20]. Yang, X. S. (2010). Firefly algorithm, stochastic test functions and design optimisation, International journal of bio-inspired computation, 2, (2): 78-84, doi.org/10.1504/IJBIC.2010.032124.
[21]. Yang, X. S. (2010). A new metaheuristic bat-inspired algorithm, in nature inspired cooperative strategies for optimization (NICSO), Berlin, Heidelberg: Springer Berlin Heidelberg, 65-74.
[22]. Das, S., Biswas, A., Dasgupta, S., & Abraham, A. (2009). Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications, in foundations of computational intelligence, 3, Global Optimization, A. Abraham, A.-E. Hassanien, P. Siarry, and A. Engelbrecht Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 23-55.
[23]. Simon, D. (2008). Biogeography-based optimization, IEEE Transactions on Evolutionary Computation, 12, (6): 702-713, doi: 10.1109/TEVC.2008.919004.
[24]. Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization, Citeseer, Technical Report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department.
[25]. Ebrahimi, A., & Khamehchi, E. (2016). Sperm whale algorithm: an effective metaheuristic algorithm for production optimization problems, Journal of Natural Gas Science and Engineering, 29, 211-222, doi.org/10.1016/j.jngse.2016.01.001.
[26]. Ge, J. (2006). Development and prospect of chemical grouting techniques, Chinese Journal of Rock Mechanics and Engineering, 25(3): 384-3.
[27]. Li, S., Liu, R., Zhang, Q., & Zhang, X. (2016). Protection against water or mud inrush in tunnels by grouting: a review, Journal of Rock Mechanics and Geotechnical Engineering, 8(5): 753-766, doi.org/10.1016/j.jrmge.2016.05.002.

Petroleum Research
Petroleum Research, 2023(June-July), Vol. 33, No. 129, 5-7
DOI:10.22078/PR.2023.4953.3209

Using the Improved Rain Optimization Algorithm to Simulate the Movement of Two Dominant Fluids in the Fracture and the Matrix

Nouri Hojjat and Ali Alianvari
Department of Mining Engineering, Faculty of Engineering, University of Kashan, Iran
hnouri_ir@yahoo.com
DOI:10.22078/PR.2023.4953.3209

Received: October/23/2022 Accepted: February/12/2023

Introduction

Most of the meta-heuristic algorithms were presented before 2000 and are known as classical meta-heuristic algorithms, some of which are:

Genetic Algorithms (GA)[1], Differential Evolution (DE) [2], Genetic Programming (GP) [3], Tabu search (TS), Greedy Randomized Adaptive Search Procedure (GRASP), Iterated Local Search (ILS), Shuffled Frog-Leaping Algorithm (SFLA), Particle Swarm Optimization, Scatter Search (SS), Ant Colony Optimization (ACO).

Despite the achievements of classical meta-heuristic algorithms, new and innovative evolutionary approaches have emerged successfully in the last two decades. Research on meta-heuristic algorithms in this era introduces a large number of new meta-heuristic algorithms inspired by specific evolutionary or behavioral processes. In many cases, this new wave of meta-heuristic approaches provides the best solutions for some set of benchmark functions. Some of these algorithms, which are known as meta-heuristic algorithms of the new generation, are:

Rain Optimization algorithm (ROA), Whale Optimization Algorithm (WOA), Symbiotic Organisms Search (SOS), Grey Wolf Algorithm (GWA), Social Spider Optimization (SSO), Teaching Learning Based Optimization (TLBO), Firefly Algorithm (FA), Bat Algorithm (BA), Gravitational Search Algorithm (GSA), Bacterial Foraging (BFO), Biogeography-based optimization (BFO), Artificial Bee Colony (ABC).

Method

Principles of simulating fluid movement in porous and fractured media

Simulation of fluid movement in a porous medium and similar to a fracture can be done by combining three important equations, i.e. the equation of mass conservation, the equation of fluid movement in a porous medium, and the equation of state. The fluid used for injection into a fracture is a single-phase flow of an incompressible fluid, the viscosity of which varies with time.

Simulation of a fracture:

To simulate the movement of an incompressible fluid in a fracture, two different states can be considered:

1- A state where the walls on both sides of the fracture have zero permeability and the fluid cannot enter the matrix around the fracture.

2- A situation where the walls on both sides of the fracture are porous and have certain permeability and the injection fluid enters the walls.

The first simulation is actually a special mode of the second general simulation, and for the simulation to be more comprehensive, this mode will be discussed further. Figure 1 shows the top view of a fracture when fluid is injected into it from a borehole.

Table 1 The standard functions used in this research and the absolute minimum value of each of them.

No	Name	Range	D	Formulation	Min
1	Rastrigin	[-5.12,5.12]	n	$f_1(x)=10n+\sum_{i=1}^{n}[x_i^{\,2}-10\cos(2\pi x_i)]$	0
2	De Jong (Sphere)	[-5.12,5.12]	n	$f_2(x)=\sum_{i=1}^{n}x_i^{\,2}$	0
3	Griewank	[-600,600]	n	$f_3(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^{\,2}-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	0
4	Beale	[-4.5,4.5]	2	$f_4(x)=(1.5-x_1+x_1x_2)^2+(2.25-x_1+x_1x_2^2)^2+...$ $...(2.625-x_1+x_1x_2^3)^2$	0
5	Easom	[-100,100]	2	$f_5(x)=-\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2-(x_2-\pi)^2)$	-1
6	Matyas	[-10,10]	2	$f_6(x)=0.26(x_1^2+x_2^2)-0.48x_1x_2$	0
7	Bohachevsky1	[-100,100]	2	$f_7(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)-0.4\cos(4\pi x_2)+0.7$	0
8	Booth	[-10,10]	2	$f_8(x)=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2$	0
9	Michalewicz2	[0, π]	2	$f_9(x)=-\sum_{i=1}^{D}\sin(x_i)(\sin(ix_i^2/\pi))^{20}$	-1.8013
10	Schaffer	[-100,100]	2	$f_{10}(x)=0.5+\frac{\sin^2(\sqrt{x_1^2+x_2^2})-0.5}{(1+0.001(x_1^2+x_2^2))^2}$	0
11	Six Hump Camel Back	[-5,5]	2	$f_{10}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	-1.03163
12	Boachevsky2	[-100,100]	2	$f_{12}(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)(4\pi x_2)+0.3$	0
13	Boachevsky3	[-100,100]	2	$f_{13}(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1+4\pi x_2)+0.3$	0
14	Shubert	[-10,10]	2	$f_{14}(x)=(\sum_{i=1}^{5}i\cos(i+1)x_1+i)(\sum_{i=1}^{5}i\cos((i+1)x_2+i))$	-186.73
15	Colville	[-10,10]	4	$f_{15}(x)=100(x_1^2-x_2)^2+(x_1-1)^2+(x_3-1)^2+90(x_3^2-x_4)^2+..$ $..10.1(x_2-1)^2+(x_4-1)^2+19.8(x_2-1)(x_4-1)$	0
16	Michalewicz5	[0, π]	5	$f_{16}(x)=-\sum_{i=1}^{D}\sin(x_i)(\sin(ix_i^2/\pi))^{20}$	-4.6877
17	Zakharov	[-5,10]	10	$f_{17}(x)=\sum_{i=1}^{D}x_i^2+(\sum_{i=1}^{D}0.5ix_i^2)^2+(\sum_{i=1}^{D}0.5ix_i^2)^4$	0
18	Michalewicz10	[0, π]	10	$f_{18}(x)=-\sum_{i=1}^{D}\sin(x_i)(\sin(ix_i^2/\pi))^{20}$	-9.6602
19	Step	[-5.12,5.12]	30	$f_{19}(x)=\sum_{i=1}^{D}(x_i+0.5)^2$	0
20	SumSquares	[-10,10]	30	$f_{20}(x)=\sum_{i=1}^{D}ix_i^2$	0
21	Quartic	[-1.28,1.28]	30	$f_{21}(x)=\sum_{i=1}^{D}ix_i^4+\mathrm{Rand}$	0

Table 1

21	Quartic	[-1.28,1.28]	30	$f_{21}(x)=\sum_{i=1}^{D} ix_i^4 + \text{Rand}$	0
22	Schwefel 2.22	[-10,10]	30	$f_{22}(x)=\sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	0
23	Schwefel 1.2	[-100,100]	30	$f_{23}(x)=\sum_{i=1}^{D} (\sum_{j=1}^{D} x_j)^2$	0
24	Rosenbrock	[-30,30]	30	$f_{24}(x)=\sum_{i=1}^{D} 100(x_{i+1}-x_i^2)^2 + (x_i-1)^2$	0
25	Dixon-Price	[-10,10]	30	$f_{25}(x)=(x_1-1)^2 + \sum_{i=2}^{D} i(2x_i^2 - x_i - 1)^2$	0
26	Ackley	[-32,32]	30	$f_{26}(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$	0

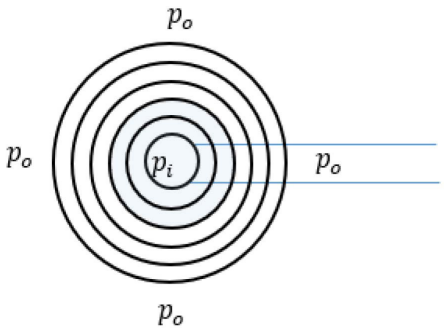


Fig. 1 The shape of a fracture when a borehole is drilled in the direction perpendicular to it and the radial movement of the injection fluid into the fracture.

Results and Discussion

In this research, a new meta-heuristic algorithm called improved rain optimization algorithm, which was inspired by the phenomenon of rainfall, was introduced. This algorithm is actually a new version of the rain algorithm and was created by making changes in this algorithm. This algorithm was used to solve several different criterion functions and a simulation problem in the field of mining engineering. Also, the performance of this algorithm was compared with other algorithms such as the genetic algorithm, particle swarm algorithm, bat algorithm, and rain algorithm, and the following results were obtained:

The improved rain optimization algorithm has a much higher speed than the rain optimization algorithm and is able to reach the solution of the problem with fewer iterations and fewer evaluations of the objective function. In this regard, in order to solve the Eggcrete standard function, the improved Baran algorithm was faster in solving this problem by 80% in terms of the number of iterations and 30% in terms of NFE.

Baran's improved algorithm was able to perform better than the genetic algorithm, particle swarm algorithm, bat algorithm, and Baran algorithm in solving complex criterion functions.

Baran's improved algorithm was used to solve a complex simulation problem in the field of mining engineering and was able to find the solution to the problem with high accuracy and good speed.

Nomenclatures

ROA: Rain Optimization algorithm

GA: Genetic Algorithms

BA: Bat Algorithm

References

1. Li, S., Liu, R., Zhang, Q., & Zhang X. (2016). Protection against water or mud inrush in tunnels by grouting: a review, Journal of Rock Mechanics and Geotechnical Engineering, 8(5): 753-766.
2. Ebrahimi, A., & Khamehchi, E., (2016). Sperm whale algorithm: an effective metaheuristic algorithm for production optimization problems, Journal of Natural Gas Science and Engineering, 29, 211-222.
3. Mirjalili, S., Mirjalili, S. M., & Lewis A. (2014). Grey wolf optimizer, Advances in Engineering Software, 69, 46-61, doi: https://doi.org/10.1016/j.advengsoft.2013.12.007.