الگوریتم سینوس کسینوس برای حل مسائل بهینه سازی

نویسنده: دکتر سید علی میرجلیلی

دانشکده صنایع اطلاعات و ارتباطات، دانشگاه گریفیث، استرالیا

ترجمه : اسماعیل فخیمی شریف ، دانشگاه علم و صنعت ایران

۰. چکیده

این مقاله یک الکوریتم بهینه‌سازی جدید مبتنی بر جمعیت به نام الگوریتم سینوس کسینوسی یا به اختصار SCA برای حل مسائل بهینه سازی را مطرح می‌کند. SCA چندین پاسخ تصادفی اولیه کاندید ایجاد میکند و آنها را ملزم می‌کند که با استفاده از یک مدل ریاضی مبتنی بر توابع سینوس و کسینوس به سمت بیرون یا به سمت بهترین پاسخ نوسان داشته باشند. همچنین به منظور تاکید بر کاوش و بهره‌برداری از فضای جستجو در نقاط عطف مختلف بهینه سازی، چندین متغیر تصادفی و تطبیقی نیز در این الگوریتم ادغام شده‌اند. عملکرد SCA در سه فاز آزمایشی محک زده می‌شود. در مرحله اول، یک مجموعه از موارد آزمایش شناخته شده شامل توابع تک وجهی، چند وجهی و ترکیبی به کار گرفته می‌شوند تا اکتشاف، بهره‌برداری، اجتناب از بهینگی محلی (اکسترمم نسبی) و همگرایی SCA سنجیده شود. در مرحله دوم، چندین معیار عملکرد (سابقه جستجو، خط سیر، میانگین تناسب پاسخ ها، و بهترین پاسخ در طول بهینه سازی) برای مشاهده کیفی و تایید عملکرد SCA در توابع تست دو بعدی انتقال یافته، استفاده می‌شود. در مرحله پایانی، سطح مقطع یک بال هواپیما به عنوان یک "موضوع مطالعه" چالش بر انگیز واقعی برای تایید و نشان دادن عملکرد این الگوریتم به وسیله SCA بهینه شده است. نتایج توابع تست و معیارهای عملکرد ثابت می‌کند که الگوریتم پیشنهادی قادر است مناطق مختلف فضای جستجو را کاوش کند، از بهینگی محلی اجتناب کند، به سمت بهینگی مطلق (اکسترمم مطلق) همگرا شود و از مناطق امیدوار کننده یک فضای جستجو به صورت موثر بهره‌برداری کند. الگوریتم SCA یک شکل صاف برای بال هواپیما با کشیدگی بسیار کم به دست می‌آورد که نشان می‌دهد این الگوریتم میتواند در حل مسائل واقعی با فضاهای جستجوی محدود و ناشناخته بسیار موثر باشد. دقت داشته باشید که کدهای منبع الگوریتم SCA به صورت همگانی در http://www.alimirjalili.com/SCA.html در دسترس است.

۱. مقدمه

بهینه‌سازی به فرآیند یافتن مقادیر بهینه برای پارامترهای یک سیستم معین از تمام مقادیر ممکن برای بیشینه یا کمینه کردن خروجی آن اشاره دارد. مسائل بهینه‌سازی را میتوان در همه زمینه‌های مطالعاتی یافت، به گونه‌ای که توسعه تکنیک‌های بهینه‌سازی را ضروری می‌سازد و یک جهت تحقیقاتی جالب برای محققان ایجاد می‌کند. با توجه به اشکالات پارادایم های بهینه‌سازی مرسوم، ایستایی بهینه محلی، و نیاز به استخراج فضای جستجو [۱]، علاقه فزاینده ای به رویکردهای بهینه‌سازی تصادفی [۲]، در طول دو دهه گذشته مشاهده شده است [۳-۵].

الگوریتم‌های بهینه‌سازی تصادفی مسائل بهینه‌سازی را به عنوان جعبه سیاه (جعبه در بسته) در نظر می‌گیرند [۶]. این بدان معنی است که استخراج مدل‌های ریاضی مورد نیاز نیست چراکه این دست پارادایم‌های بهینه‌سازی فقط ورودی‌ها را تغییر می‌دهند و خروجی‌های سیستم را برای بیشینه یا کمینه شدن خروجی آنها نظارت می‌کنند. یکی دیگر از مزایای در نظر گرفتن مسائل به عنوان جعبه سیاه، انعطاف‌پذیری بالا است، بدین معنی که الگوریتم‌های تصادفی به راحتی برای مسائل در زمینه های مختلف قابل استفاده هستند. همانطور که از نام تکنیک‌های بهینه‌سازی

تصادفی پیداست، آنها مسائل بهینه‌سازی را به صورت تصادفی بهینه می‌کنند [۷]. بنابراین، آنها ذاتا از "اجتناب از بهینگی محلی" بالاتری در مقایسه با الگوریتم‌های بهینه‌سازی مرسوم بهره می‌برند.

طبقه‌بندی‌های مختلفی برای الگوریتم‌های تصادفی در این حوزه وجود دارد. دو طبقه بندی اصلی بر اساس الهام از یک الگوریتم (مبتنی بر هوش جمعی [۸]، تکاملی [۹]، مبتنی بر فیزیک [۱۰] و غیره) و تعداد پاسخ‌های تصادفی که یک الگوریتم در هر مرحله از بهینه‌سازی ایجاد می‌کند، است. جدیدترین طبقه‌بندی الگوریتم‌ها را به دو دسته تقسیم می‌کند: الگوریتم‌های مبتنی بر فرد و الگوریتم‌های مبتنی بر جمعیت. در طبقه‌بندی اول (مبتنی بر فرد)، تنها یک پاسخ به طور تصادفی تولید می‌شود و در طول عملیات بهینه‌سازی بهبود می‌یابد. اما در طبقه‌بندی دوم (مبتنی بر جمعیت)، یک الگوریتم بهینه‌سازی بیش از یک پاسخ تصادفی (عمدتا تعداد زیادی) تولید میکند و آنها را در طول بهینه سازی بهبود می‌بخشد.

با توجه به مزایای ذکر شده در بالا، تکنیک‌های بهینه‌سازی تصادفی در این حوزه بسیار محبوب شده‌اند. این محبوبیت نه تنها در زمینه بهینه‌سازی بلکه در سایر زمینه های مطالعاتی نیز وجود دارد. کاربرد الگوریتم‌های تصادفی را میتوان در شاخه‌های مختلف علم و صنعت یافت. از آنجایی که تمرکز این مقاله بر روی جنبه نظری (تئوری) است، کاربردها زیاد مورد بحث قرار نگرفته و خوانندگان علاقه‌مند به [۱۱,۱۲] ارجاع داده می‌شوند.

تحقیقات نظری موجود در این حوزه را میتوان به سه جهت اصلی تقسیم کرد: بهبود تکنیک‌های فعلی، ترکیب الگوریتم‌های مختلف و پیشنهاد الگوریتم‌های جدید. در رویکرد اول، محققان سعی می‌کنند الگوریتم‌ها را با عملگرهای مختلف ریاضی یا تصادفی برای بهبود عملکرد تجهیز کنند. روش‌های رایج در این طبقه عبارتند از: نقشه های آشفته [۱۳-۱۷]، عملگرهای تکاملی [۱۸-۲۳]، و جستجوهای محلی [۲۴-۲۷]. دومین جهت تحقیقاتی محبوب با ترکیب الگوریتم‌های مختلف برای بهبود عملکرد یا حل مسائل خاص سر و کار دارد [۲۸-۳۵]. تعداد قابل توجهی از فرا ابتکاری ترکیبی وجود دارد، مانند: PSO-DE [۴۰], GA-DE [۳۹], ACO-GA [۳۸], PSO-ACO [۳۷], PSO-GA [۳۶], KH-CS [۴۲], ACO-DE [۴۱] و KH-BBO [۴۳].

آخرین اما نه کم اهمیت، پیشنهاد الگوریتم های جدید یک راه تحقیقاتی محبوب برای بسیاری از محققان است. یک الگوریتم جدید می تواند از پدیده های تکاملی، رفتار جمعی موجودات (تکنیک های هوش ازدحام)، قوانین فیزیکی و مفاهیم مربوط به انسان الهام گرفته شده باشد. برخی از الگوریتم‌های اخیر و محبوب در هر یک از طبقه‌بندی ها به شرح زیر است:

- تکنیک‌های تکاملی: الگوریتم‌های ژنتیک (GA) [۴۴]، تکامل افتراقی (Differential Evolution) (DE) [۴۵-۴۸]، الگوریتم‌های بهینه‌سازی مبتنی بر جغرافیای زیستی (BBO) [۴۹]، و استراتژی تکامل (ES) [۵۰].
- تکنیک‌های هوش ازدحام: بهینه‌سازی کلنی مورچه‌ها (ACO) [۵۱]، بهینه‌سازی ازدحام ذرات (PSO) [۵۲] و الگوریتم کلونی زنبورهای مصنوعی (ABC) [۵۳].
- تکنیک‌های مبتنی بر فیزیک: الگوریتم جستجوی گرانشی (GSA) [۵۴]، بهینه‌سازی اجسام برخوردی (CBO) [۵۵] و سیاهچاله (BH) [۵۶].
- تکنیک‌های مرتبط با انسان: الگوریتم قهرمانی لیگ (LCA) [۵۷]، الگوریتم انفجار معدن (MBA) [۵۸]، بهینه‌سازی مبتنی بر آموزش (TLBO) [۵۹].

با وجود تعداد قابل توجهی از الگوریتم‌های پیشنهاد شده اخیر در این زمینه، یک سوال اساسی در اینجا وجود دارد که آیا و چرا ما به تکنیک‌های بهینه‌سازی بیشتری نیاز داریم. این سوال را می توان با مراجعه به قضیه به اصطلاح "ناهار رایگان نداریم" (No Free Lunch) (NFL) [۶۰] پاسخ داد. این قضیه به طور منطقی ثابت می‌کند که هیچ‌کس

نمی‌تواند الگوریتمی برای حل تمام مسائل بهینه‌سازی پیشنهاد کند. این بدان معناست که موفقیت یک الگوریتم در حل یک سری از مسائل خاص، حل تمام مسائل بهینه‌سازی را با نوع و ماهیت مختلف را تضمین نمی‌کند. به عبارت دیگر تمام تکنیک‌های بهینه‌سازی با وجود عملکرد عالی در زیرمجموعه‌ای از مسائل بهینه‌سازی، هنگام در نظر گرفتن همه مسائل بهینه‌سازی، به طور میانگین یکسان عمل می‌کنند. قضیه NFL به محققان اجازه می‌دهد الگوریتم‌های جدیدی را پیشنهاد کننده یا الگوریتم‌های فعلی را برای حل زیرمجموعه‌های مسائل در زمینه‌های مختلف بهبود دهند یا اصلاح کنند.

همچنین انگیزه این کار این است که یک الگوریتم بهینه‌سازی ساده و در عین حال موثر برای بهینه‌سازی مسائل واقعی با فضاهای جستجوی ناشناخته پیشنهاد شده است. همچنین این مقاله نشان می‌دهد که می‌توان از توابع ساده ریاضی برای طراحی الگوریتم‌های بهینه‌سازی در این زمینه استفاده کرد. الگوریتم پیشنهادی از توابع سینوسی و کسینوسی برای کشف و بهره‌برداری از فضای بین دو پاسخ در فضای جستجو با اامید به یافتن پاسخ‌های بهتر استفاده می‌کند. در اینجا قابل ذکر است که نگارنده اخیراً الگوریتمی به نام الگوریتم Moth-Flame (MFO) [۶۱] ارائه کرده است. الگوریتم پیشنهادی در این کار از نظر الهام، فرمول بندی ریاضی و کاربرد در دنیای واقعی کاملاً متفاوت است. الگوریتم MFO از مسیریابی پروانه‌ها در طبیعت تقلید می‌کند، در حالی که الگوریتم SCA مبتنی بر توابع ریاضی سینوسی/کسینوس برای حل مسائل بهینه‌سازی است. MFO برای بهینه‌سازی شکل پروانه استفاده شده است، در حالی که SCA برای بهینه‌سازی شکل ایرفویل دو بعدی در بال‌های هواپیما استفاده می‌شود. بقیه مقاله به شرح زیر تنظیم شده است:

بخش ۲ شامل مقدمات و تعاریف اساسی، ارائه آثار مرتبط و مرور این حوزه می‌باشد. بخش ۳ مدل ریاضی را نشان می‌دهد و الگوریتم سینوس کسینوس (SCA) را بیان می‌کند. بسترهای آزمایشی به کار گرفته شده و نتایج به دست آمده در بخش ۴ ارائه و مورد بحث قرار گرفته است. شکل مقطع بال هواپیما توسط الگوریتم SCA در بخش ۵ بهینه شده است که مزایای این الگوریتم را در حل مسائل چالش بر انگیز واقعی با تعداد زیادی محدودیت و فضاهای جستجوی ناشناخته نشان می‌دهد. در نهایت، بخش ۶ دستاوردهای مقاله را فهرست می‌کند، کار را به پایان می‌رساند و چندین جهت را برای مطالعات آینده پیشنهاد می‌کند.

۲. اعمال مرتبط

این بخش ابتدا مقدمات و تعاریف بهینه‌سازی را پوشش می‌دهد. سپس مکانسیم‌ها و چالش‌های تکنیک‌های بهینه‌سازی تصادفی/ابتکاری را مورد بحث قرار می‌دهد. و در آخر انگیزه اینکار آورده شده است.

۱/۱. مقدمات و تعاریف

بهینه‌سازی تک معیاره تنها با بهینه‌سازی یک معیار (هدف) سروکار دارد. این عبارت قبل از بهینه‌سازی چند معیاره قرار میگیرد که در آن بیش از یک هدف برای بهینه‌سازی وجود دارد. رسیدگی به اهداف چندگانه نیازمند ملاحظات و مکانیسم‌های خاصی است، بنابراین خوانندگان علاقه‌مند به مقاله مروری اخیر از Zhou et al. [۵] ارجاع داده می‌شوند. چراکه تمرکز این اثر بر روی بهینه‌سازی تک معیاره است.

علاوه بر معیار، سایر عناصر دخیل در فرآیند بهینه‌سازی تک معیاره، پارامترها و محدودیت‌ها هستند. پارامترها متغیرها (مجهولات) مسائل بهینه‌سازی (سیستم‌ها) هستند که باید بهینه شوند. همانطور که شکل ۱ نشان می‌دهد، متغیرها را می‌توان به عنوان ورودی‌های اولیه در نظر گرفت و شروط محدودیت‌های اعمال شده برای سیستم هستند. در واقع، شروط امکان‌سنجی مقدار معیار (هدف) بدست آمده را تعریف می‌کنند. نمونه‌هایی از شروط، شروط تنش در هنگام طراحی سیستم‌های آیرودینامیکی یا محدوده متغیرها هستند.

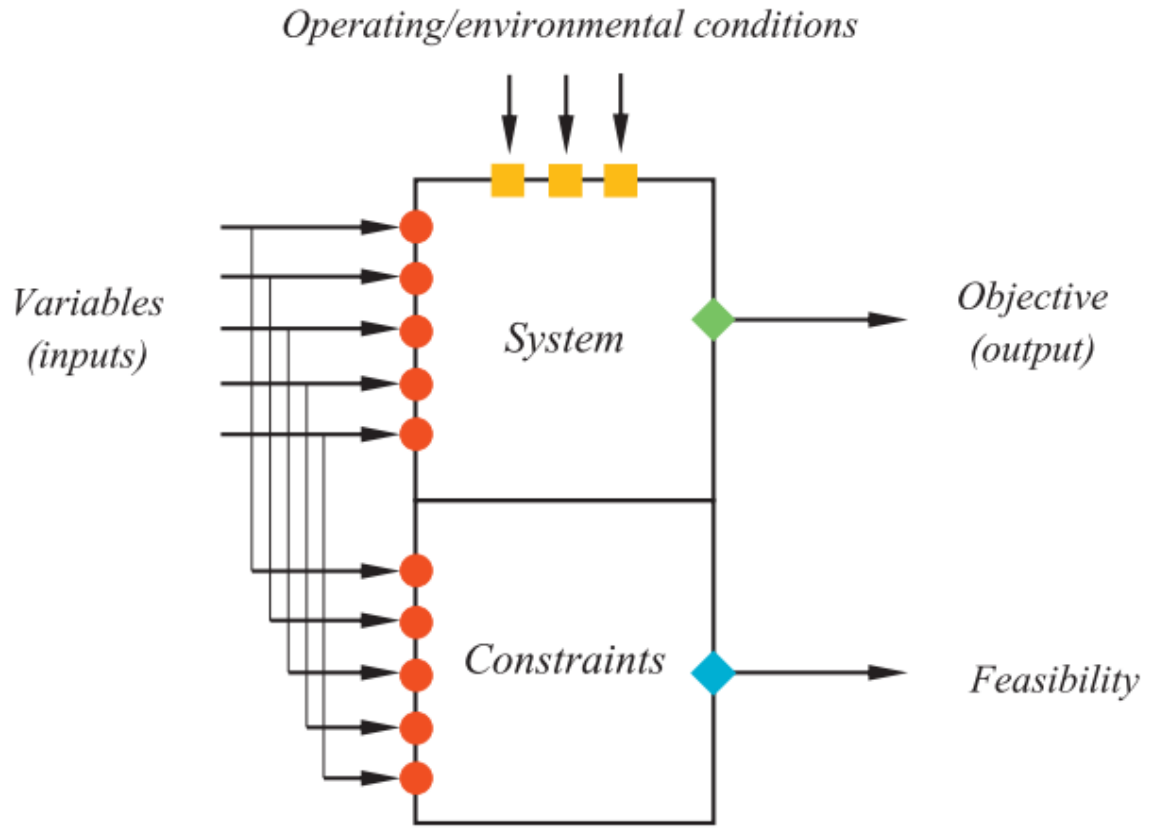


Fig. 1. Different components of an optimization system.

سایر ورودی‌های یک سیستم که ممکن است بر خروجی آن تاثیر بگذارد، شرایط عملیاتی/محیطی است. چنین ورودی‌هایی به عنوان ورودی‌های ثانویه در نظر گرفته می‌شوند که زمانی تعریف می‌شوند که یک سیستم در محیط شبیه‌سازی شده/نهایی کار می‌کند. نمونه‌هایی از این شرایط عبارتند از: دما/ضخامت سیال هنگام چرخش ملخ یا زاویه حمله هنگام پرواز هواپیما. این نوع ورودی‌ها توسط بهینه‌ساز ها، بهینه‌سازی نمی‌شوند، اما قطعا باید در طول بهینه‌سازی در نظر گرفته شوند، زیا ممکن است تاثیرات قابل توجهی بر خروجی‌ها داشته باشند.

بدون از دست رفتن کلیت، یک بهینه‌سازی تک معیاره را میتوان به عنوان یک مسئله کمینه‌سازی به صورت زیر فرموله کرد:

$$\text{Minimize}: \quad f(x_1, x_2, x_3, \ldots, x_{n-1}, x_n) \tag{2.1}$$

$$\text{Subject to}: \quad g_i(x_1, x_2, x_3, \ldots, x_{n-1}, x_n) \geq 0, \quad i = 1, 2, \ldots, m \tag{2.2}$$

$$h_i(x_1, x_2, x_3, \ldots, x_{n-1}, x_n) = 0, \quad i = 1, 2, \ldots, p \tag{2.3}$$

$$lb_i \leq x_i \leq ub_i \quad i = 1, 2, \ldots, n \tag{2.4}$$

که n تعداد متغیرها است، m تعداد نامساوی‌های محدودیت ها را مشخص می‌کند، p تعداد تساوی محدودیت ها را نشان می‌دهد، lb_i مرز پایینی متغیر iم و ub_i مرز بالایی متغیر iم است.

همانطور که در معادلات (۲.۲) و (۲.۳) می‌توان مشاهده نمود، دو نوع محدودیت وجود دارد : تساوی و نامساوی. مجموعه متغیرها، محدودیت ها و هدف، یک فضای جستجوی برای مسئله داده شده را می‌سازد. متاسفانه، ترسیم فضای جستجوی با توجه به ابعاد بالای متغیرها معمولا غیرممکن است. اگرچه، یک مثال از فضای جستجوی ساخته شده از ۲ متغیر و چندین محدودیت در تصویر شماره ۲ نمایش داده شده است.

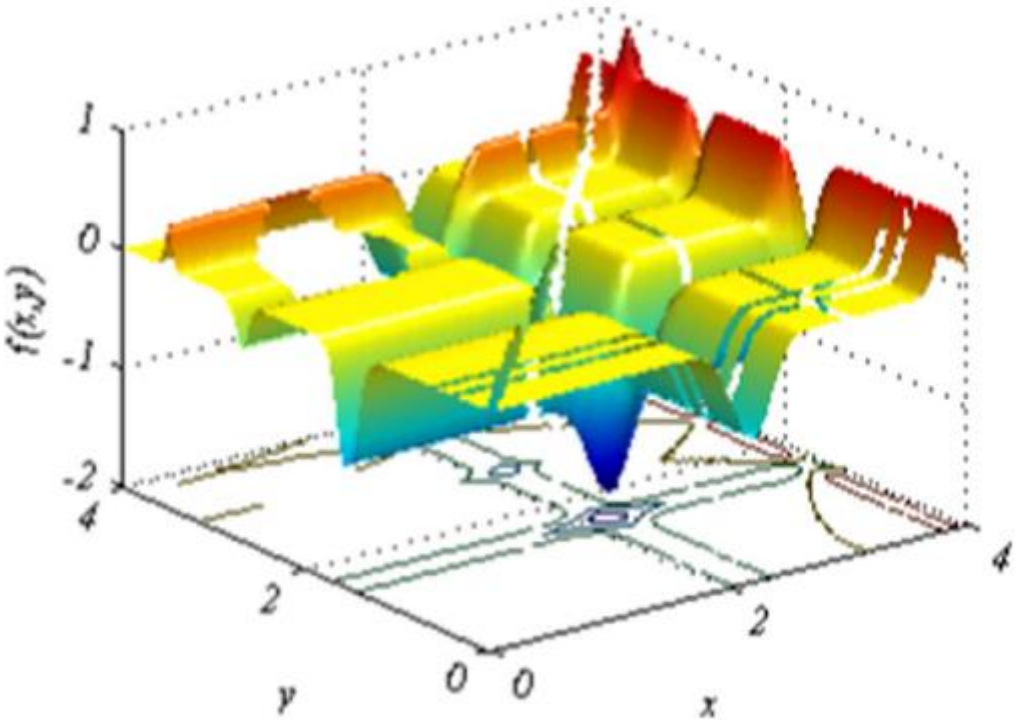


Fig. 2. Example of a search space with two variables and several constraints.

در تصویر ۲ ممکن است مشاهده شود که فضای جستجو می‌تواند چندین بهینگی محلی (اکسترمم نسبی) داشته باشد، اما یکی از آنها (یا بیش از یکی در مورد یک منظره مسطح) بهینه سراسری (اکسترمم مطلق) است. محدودیت‌ها شکاف‌هایی در فضای جستجو ایجاد می‌کنند و گه‌گاه آن را به مناطق مختلف جدا شده تقسیم می‌کنند. در متن، مناطق غیرممکن به مناطقی از فضای جستجو اشاره می‌کنند که محدودیت‌ها را نقض میکنند.

فضای جستجوی یک مسئله واقعی می‌تواند بسیار چالش بر انگیز باشد. برخی مشکلات فضاهای جستجوی واقعی عبارتند از: ناپیوستگی، تعداد زیاد بهینگی محلی، تعداد زیاد محدودیت ها، بهینه سراسری واقع در مرز محدودیت ها، دره‌های فریبنده به سمت بهینگی محلی و جداسازی بهینه سراسری. یک الگوریتم بهینه‌سازی باید مجهز به عملگرهای مناسب برای رسیدگی به همه این مشکلات برای یافتن بهینه سراسری باشد.

با فرموله کردن یک مسئله، یک بهینه‌ساز می‌تواند متغیرهای خود را بر اساس خروجی‌ها و محدودیت‌ها تنظیم کند.همانطور که در بخش ۱ ذکر شد، یکی از مزایای الگوریتم‌های تصادفی این است که یک سیستم را به عنوان جعبه سیاه در نظر می‌گیرند. شکل شماره ۳ نشان می‌دهد که بهینه‌ساز فقط متغیرها را در اختیار سیستم قرار می‌دهد و خروجی‌ها را مشاهده می‌کند. سپس بهینه‌ساز به صورت تکراری و تصادفی ورودی‌های سیستم را بر اساس بازخورد (خروجی) به دست آمده تا زمان برآورده شدن یک معیار تغییر می‌دهد. فرآیند تغییر متغیرها بر اساس تاریخچه خروجی‌ها با مکانیزم یک الگوریتم تعریف می‌شود. برای مثال، PSO بهترین پاسخ‌های به دست آمده را ذخیره می‌کند و پاسخ‌های جدید را تشویق می‌کند تا در اطراف آنها جابه‌جا شوند.

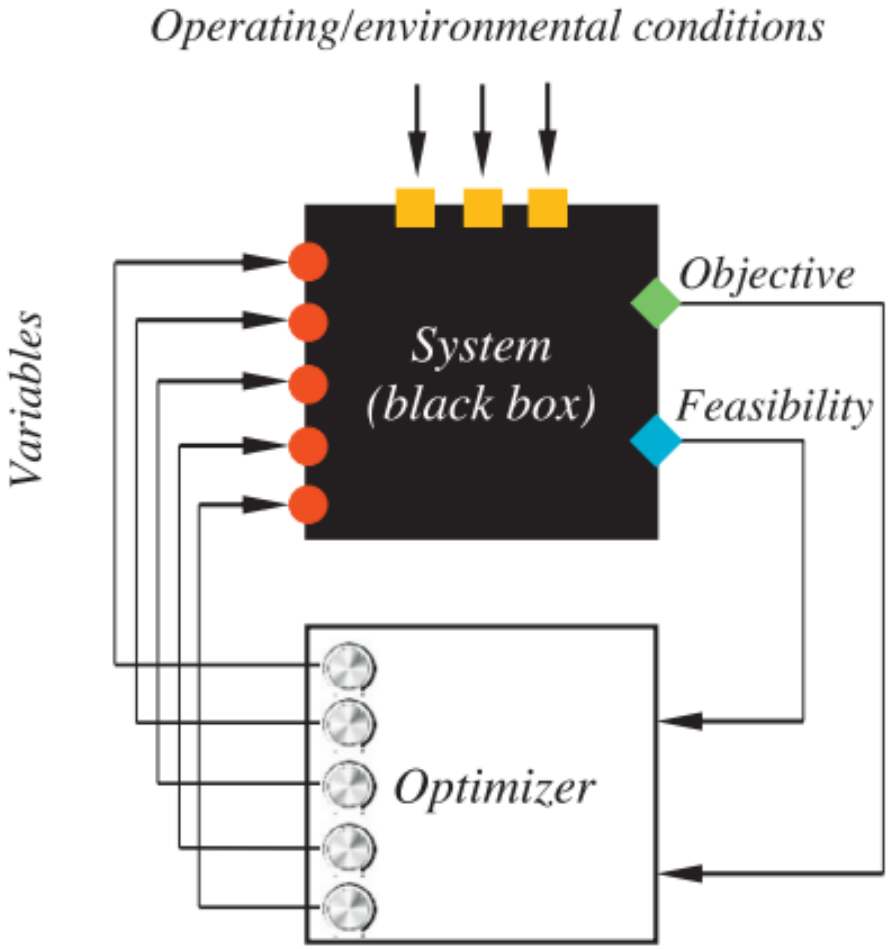


Fig. 3. Stochastic population-based optimizers consider the system as black box.

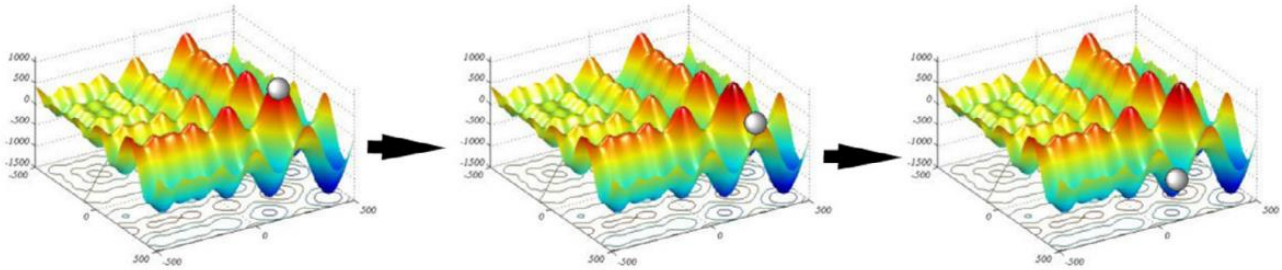
تاریخچه تکنیک‌های بهینه‌سازی تصادفی/اکتشافی و چالش‌های طراحی آنها در زیربخش زیر بررسی و به تفصیل مورد بحث قرار می‌گیرد.

۱٫۲. مرور تاریخچه

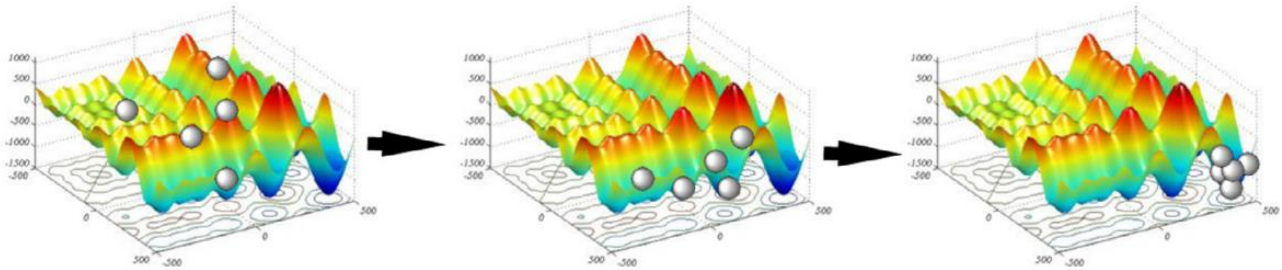
در زمینه بهینه‌سازی، در سال ۱۹۷۷، یک ایده انقلابی توسط هالند ارائه شد که در آن مفاهیم تکاملی در طبیعت در کامپیوتر برای حل مسائل بهینه‌سازی شبیه‌سازی شد [۴۴]. الگوریتم GA به وجود آمد و راه جدیدی برای مقابله با مشکلات چالش بر انگیر در زمینه‌های مختلف مطالعاتی باز کرد. ایده کلی الگوریتم GA بسیار ساده بود. انتخاب، نوترکیبی و جهش ژن‌ها در طبیعت را تقلید می‌کرد. در واقع، نظریه تکامل داروین الهام‌بخش اصلی این الگوریتم بود. در GA، فرآیند بهینه‌سازی با ایجاد مجموعه‌ای از پاسخ‌های کاندید (انفرادی) برای یک مسئله بهینه‌سازی مشخص آغاز می‌شود. هر متغیر مسئله به عنوان یک ژن در نظر گرفته می‌شود و مجموعه متغیرها مشابه کروموزوم‌ها هستند. مشابه طبیعت، تابع هزینه تناسب هر کروموزوم را مشخص می‌کند. کل مجموعه پاسخ‌ها به عنوان یک جمعیت در نظر گرفته می‌شود. هنگامی که تناسب کروموزوم‌ها محاسبه می‌شود، بهترین کروموزوم‌ها به طور تصادفی برای ایجاد جمعیت بعدی انتخاب می‌شوند. الهام‌بخش اصلی الگوریتم GA اینجاست که در آن افراد مناسب‌تر احتمال بیشتری دارند که انتخاب شوند و در ایجاد جمعیت بعدی مشابه آنچه در طبیعت اتفاق می‌افتد شرکت کنند. مرحله بعدی ترکیب افراد انتخاب شده است. در این مرحله ژن‌های جفت افراد به طور تصادفی با هم ادغام می‌شوند تا افراد جدید تولید شوند. در نهایت، برخی ژن‌های افراد در جمعیت به طور تصادفی برای تقلید جهش تغییر می‌کنند.

الگوریتم GA ثابت کرد که پارادایم‌های الهام گرفته شده از طبیعت می‌توانند در مسائل بهینه‌سازی، بسیار ساده و در عین حال قدرتمند باشند. پس از پیشنهاد الگوریتم GA، زمینه تکنیک‌های بهینه‌سازی تصادفی بسیار مورد توجه قرار گرفت. الگوریتم PSO [۵۲] حاصل این محبوبیت چندین ساله پس از اختراع الگوریتم GA است. الگوریتم PSO رفتار اجتماعی و فردی گله حیوانات، ماهی‌ها یا دسته‌های پرندگان را در جستجوی غذا تقلید می‌کند. مشابه الگوریتم GA، فرآیند بهینه‌سازی با مجموعه ای از پاسخ‌های به طور تصادفی ایجاد شده آغاز می‌شود. علاوه بر مجموعه پاسخ‌ها، مجموعه دیگری به نام سرعت وجود دارد که وظیفه ذخیره و تعیین میزان حرکت ذرات را بر عهده دارد. در طول بهینه‌سازی، سرعت یک ذره بر اساس بهترین پاسخی که تا کنون به دست آورده و همچنین بهترین پاسخی که ازدحام را یافته است، به روز می‌شود. سه مولفه تصادفی در تعریف وجود دارد، گرایش به سرعت قبلی، بهترین تاثیر فردی و بهترین تاثیر سراسری. از آنجایی که بهترین پاسخ‌ها در الگوریتم PSO ذخیره می‌شوند، همیشه هنگام جستجو در اطراف آنها، امکان یافتن پاسخ‌های بهتر وجود دارد. این دلیل اصلی موفقیت الگوریتم PSO است.

پس از توسعه الگوریتم‌های GA و PSO، چندین الگوریتم توسعه و مطرح شد. همانطور که در مقدمه ذکر شد، آنها را می‌توان به دو دسته اصلی تقسیم کرد: الگوریتم‌های مبتنی بر فرد در مقابل الگوریتم‌های مبتنی بر جمعیت. الگوریتم های مبتنی بر فرد تنها یک پاسخ ایجاد می‌کنند و آن را در طول تکرارها تکامل/بهبود می‌بخشند. در صورتی که یک الگوریتم مبتنی بر جمعیت، فرآیند بهینه‌سازی را با بیش از یک پاسخ آغاز می‌شوند. سپس پاسخ‌های این مجموعه در طول تکرارها بهبود می‌یابند. روشی که این دو خانواده بهینه‌سازی را انجام می دهند در شکل ۴ نشان داده شده است. مزیت الگوریتم های مبتنی بر فردی نیاز به تعداد کم ارزیابی تابع است زیرا یک پاسخ تنها به یک ارزیابی تابع نیاز دارد. بنابراین، چنین تکنیک‌های بهینه‌سازی به T × ۱ تعداد ارزیابی تابع نیاز دارند که در آن T حداکثر تعداد تکرار است. با این حال، احتمال بالای ایستایی بهینگی محلی و فقدان اشتراک‌گذاری اطلاعات از ایرادات اصلی این الگوریتم‌ها است که به دلیل تعداد کم پاسخ‌ها می‌باشد. شکل ۴(a) نشان می‌دهد که پاسخ کاندید واحد در بهینگی محلی که بسیار نزدیک به بهینه سراسری است به دام می‌افتد.



(a) Individual-based stochastic optimization



(b) Population-based stochastic optimization

Fig. 4. Individual-based versus population-based stochastic optimization algorithms.

در مقابل الگوریتم‌های مبتنی بر جمعیت از اجتناب از بهینه محلی بالایی بهره می‌برند زیرا از پاسخ‌های متعدد استفاده می‌کنند. شکل ۴(b) نشان می‌دهد که چگونه مجموعه پاسخ‌های کاندید منجر به یافتن بهینه سراسری

می‌شود. پاسخ‌های متعدد همچنین به یک الگوریتم مبتنی بر جمعیت کمک می‌کند تا اطلاعات را از مناطق مختلف فضای جستجو به راحتی جمع‌آوری کند. این کار با تبادل اطلاعات بین عوامل جستجو در طول فرآیند بهینه سازی انجام می‌شود. بنابراین، عوامل جستجو می‌توانند فضاهای جستجو را بهتر و سریعتر کاوش کرده و از آنها بهره‌برداری کنند. با این حال، اشکال اصلی این روش‌ها، تعداد زیاد توابع ارزیابی است. روش های بهینه‌سازی این‌چنینی به n × T عدد تابع ارزیابی نیاز دارند که در آن n تعداد پاسخ‌ها (عامل‌های جستجو) و Tحداکثر تعداد تکرار است.

۱/۳.انگیزه کار

با وجود نیاز به به توابع ارزیابی بیشتر، تاریخچه نشان می‌دهد که الگوریتم‌های مبتنی بر جمعیت برای حل مسائل چالش‌برانگیز واقعی بسیار مناسب هستند، زیرا می‌وانند از بهینه محلی اجتناب کنند، فضای جستجو را کاوش کنند و از بهینه سراسری با اطمینان بیشتری در مقایسه با الگوریتم‌های مبتنی بر فرد بهره‌برداری کنند. علاوه بر این، قضیه NFL می‌گوید که همه الگوریتم‌ها در تمام مسائل بهینه‌سازی یکسان عمل می‌کنند. بنابراین هنوز مسائلی وجود دارد که هنوز حل نشده‌اند یا با الگوریتم‌های جدید بهتر می‌توان آنها را حل کرد. این دو دلیل انگیزه‌های اصلی این کار هستند که در آن یک الگوریتم جدید بهینه سازی مبتنی بر جمعیت پیشنهاد شده و با الگوریتم‌های شناخته شده فعلی در این زمینه مقایسه شده است.

۳. الگوریتم سینوس کسینوس SCA

به طور کلی، روش‌های بهینه‌سازی مبتنی بر جمعیت، فرآیند بهینه‌سازی را با مجموعه‌ای از پاسخ‌های تصادفی آغاز می‌کند. این مجموعه تصادفی مکررا به وسیله تابع هدف ارزیابی می‌شود و با مجموعه‌ای از قوانین که هسته اصلی یک روش بهینه‌سازی است، بهبود می‌یابد. از آنجایی که روش‌های بهینه‌سازی مبتی بر جمعیت به طور تصادفی به دنبال مقدار بهینه مسائل بهینه‌سازی هستند، هیچ تضمینی برای یافتن پاسخ در یک اجرا وجود ندارد. با این حال، با تعداد کافی پاسخ تصادفی و مراحل بهینه‌سازی (تکرار)، احتمال یافتن بهینه سراسری افزایش می‌یابد.

صرف نظر از تفاوت‌های بین الگوریتم ها در زمینه بهینه‌سازی مبتنی بر جمعیت تصادفی، تفاوت مرسوم تقسیم فرآیند بهینه‌سازی به دو فاز است : اکتشاف در مقابل بهره‌برداری[۶۲]. در فاز اول، یک الگوریتم بهینه‌سازی پاسخ‌های تصادفی موجود در مجموعه پاسخ‌ها را به طور ناگهانی با نرخ تصادفی بالایی ترکیب می‌کند تا مناطق امیدوار کننده فضای جستجو را پیدا کند. در فاز بهره‌برداری، با این حال که تغییرات تدریجی در پاسخ‌های تصادفی وجود دارد، و تغییرات تصادفی به طور قابل توجهی کمتر از تغییرات در فاز اکتشاف است.

در این مقاله، معادله به‌روزرسانی زیر برای هر دو فاز مطرح می‌شود:

$$X_i^{t+1} = X_i^t + r_1 \times \sin(r_2) \times \left| r_3 P_i^t - X_i^t \right| \quad (3.1)$$

$$X_i^{t+1} = X_i^t + r_1 \times cos(r_2) \times \left| r_3 P_i^t - X_{i\,i}^t \right| \quad (3.2)$$

که X^t_i موقعیت پاسخ فعلی در بعد iام در تکرار tام است، r۱/r۲/r۳ اعداد تصافی هستند، P_i موقعیت نقطه مقصد در بعد iام است، و | | مشخص کننده قدر مطلق است.

این دو معادله برای استفاده به صورت زیر ترکیب شده اند:

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 \times \sin(r_2) \times \left| r_3 P_i^t - X_i^t \right|, & r_4 < 0.5 \\ X_i^t + r_1 \times cos(r_2) \times \left| r_3 P_i^t - X_{ii}^t \right|, & r_4 \geq 0.5 \end{cases} \quad (3.3)$$

که r۴ یک عدد تصادفی در بازه [۰,۱] است.

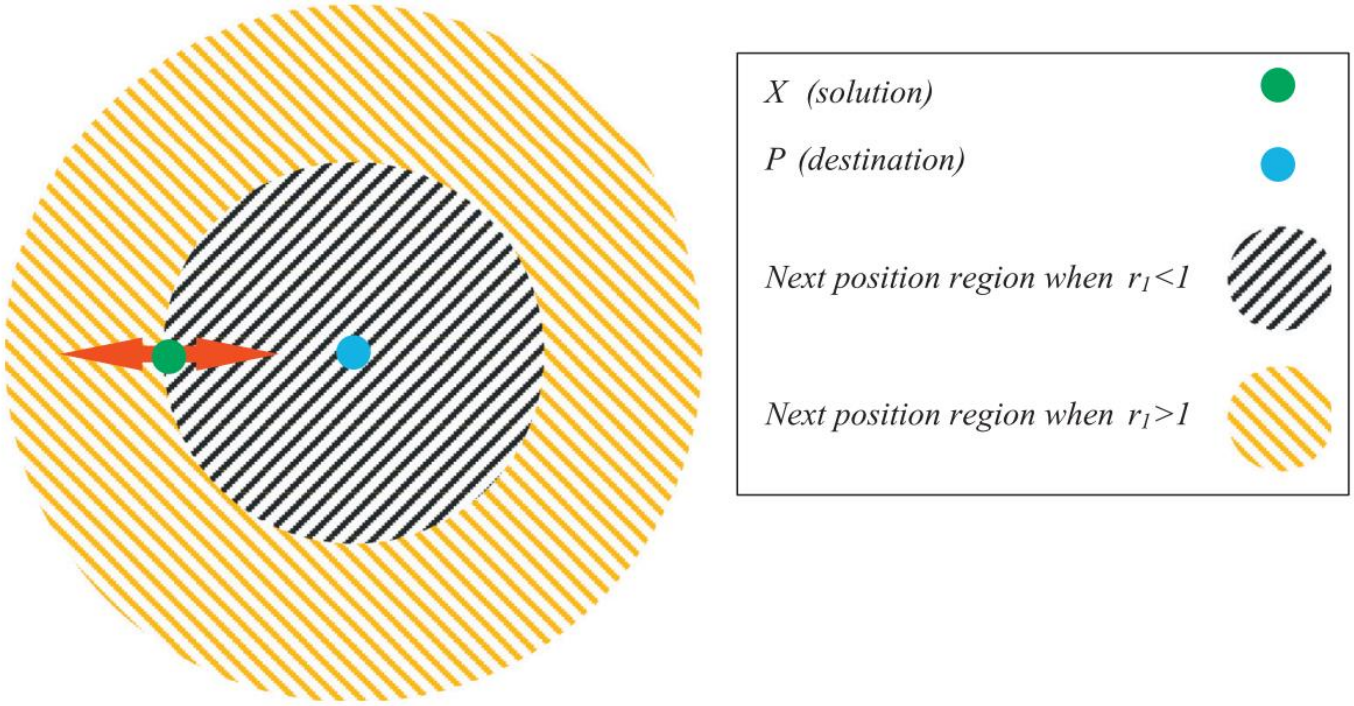


Fig. 5. Effects of Sine and Cosine inn Eqs. (3.1) and (3.2) on the next position.

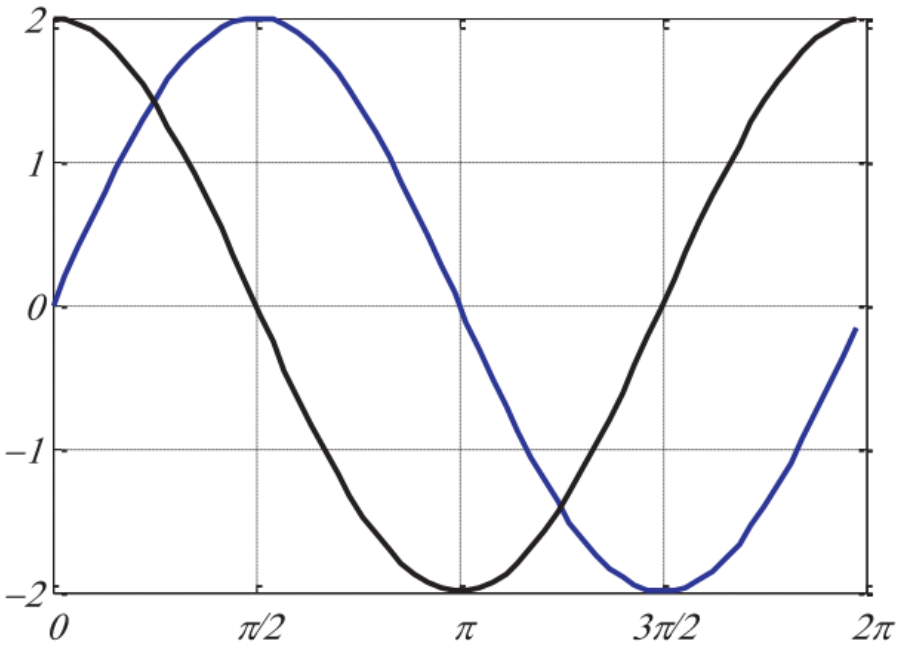


Fig. 6. Sine and cosine with range of [−2,2].

همانطور که معادلات بالا نشان می‌دهند، در SCA چهار پارامتر اصلی وجود دارد: r۱,r۲,r۳ و r۴. پارامتر r۱ ناحیه موقعیت بعدی (یا جهت حرکت) را مشخص می‌کند که می‌تواند در فضای بین پاسخ و مقصد یا خارج از آن باشد. پارامتر r۲ تعیین می‌کند که حرکت چقدر باید به سمت یا خارج از مقصد باشد. پارامتر r۳ یک وزن تصادفی را برای

مقصد تعیین می‌کند تا به صورت تصادفی تاثیر مقصد در تعریف فاصله را تاکیید کند (r۳ > ۱) یا تاکیید نکند (r۳ < ۱). در آخر، پارامتر r۴ به طور مساوی بین اجزای سینوسی و کسینوس در معادله (۳.۳) سوئیچ می‌کند.

با توجه به استفاده از سینوس و کسینوس در این فرمولاسیون، این الگوریتم به نام الگوریتم سینوس کسینوسی (SCA) نامگذاری شده است. اثرات سینوس و کسینوس بر معادلات (۳.۱) و (۳.۲) در شکل ۵ نشان داده شده است. این شکل نشان می‌دهد که چگونه معادلات پیشنهادی یک فضای بین دو پاسخ را در فضای جستجو تعریف می‌کنند. لازم به ذکر است که این معادله را می‌توان به ابعاد بالاتر تعمیم داد اگرچه یک مدل دو بعدی در شکل ۵ نشان داده شده است. الگوی چرخه‌ای تابع سینوس و کسینوس اجازه می‌دهد تا یک پاسخ دوباره در اطراف پاسخ دیگری قرار گیرد. این می‌تواند بهره‌برداری از فضای تعریف شده بین دو پاسخ را تضمین کند. برای کاوش در فضای جستجو، پاسخ‌ها باید بتوانند خارج از فضای بین مقصدهای مربوطه خود را نیز جستجو کنند. این را می‌توان با تغییر دامنه توابع سینوس و کسینوس همانطور که در شکل ۶ نشان داده شده است بدست آورد.

یک مدل مفهومی از اثرات توابع سینوس و کسینوس با دامنه [-۲,۲] در شکل ۷ نشان داده شده است. این شکل نشان می‌دهد که چگونه تغییر دامنه توالع سینوس و کسینوس نیازمند یک پاسخ برای به‌روز رسانی موقعیت آن در خارج یا داخل فضای بین خود و پاسخ دیگر است. مکان تصادفی در داخل یا خارج با تعریف یک عدد تصادفی برای r۲ در [۰,۲π] در معادله (۳.۳) به دست می‌آید. بنابر این، این مکانیسم به ترتیب کاوش و بهره‌برداری از فضای جستجو را تضمین می‌کند.

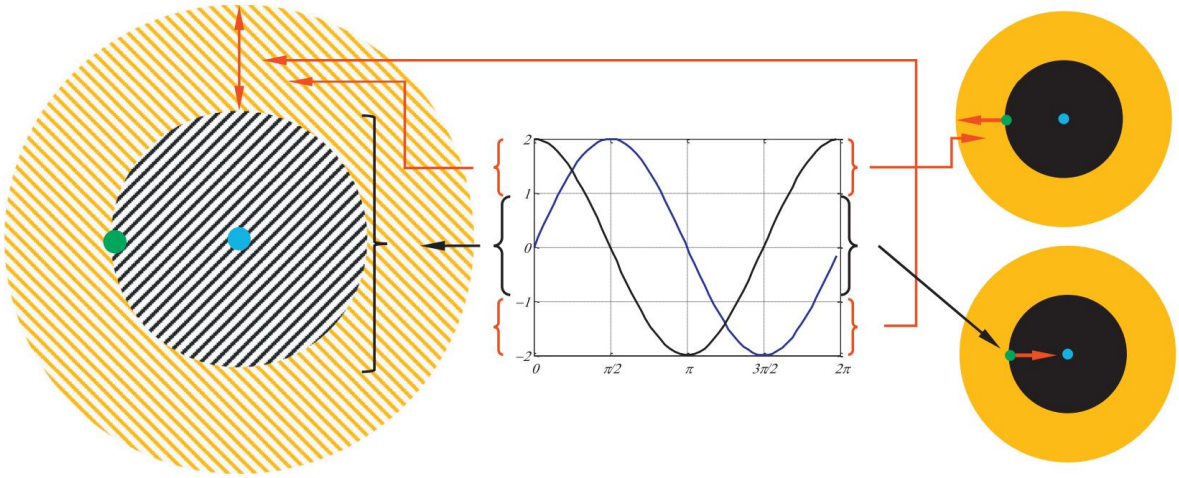


Fig. 7. Sine and cosine with the range in [−2,2] allow a solution to go around (inside the space between them) or beyond (outside the space between them) the destination.

یک الگوریتم باید بتواند اکتشاف و بهره‌برداری را متعادل کند تا مناطق امیدوارکننده فضای جستجو را پیدا کند و در نهایت به بهینه سراسری همگرا شود. به منظور ایجاد تعادل بین اکتشاف و بهره‌برداری، محدوده سینوس و کسینوس در معادلات (۳.۱) تا (۳.۳) به صورت تطبیقی با استفاده از معادله زیر تغییر می‌کند:

$$r_1 = a - t\frac{a}{T} \qquad (3.4)$$

که t تکرار فعلی است، T حداکثر تعداد تکرارها است، و a یک ثابت است.

شکل ۸ نشان می‌دهد که چگونه این معادلات محدوده توابع سینوس و کسینوس را در طول دوره تکرارها کاهش می‌دهد. از شکل‌های ۷ و ۹ می‌شود استنباط کرد که الگوریتم SCA فضای جستجو را زمانی که محدوده توابع

سیسنوس و کسینوس در بازه [۱,۲] و [-۲,-۱] است کاوش می‌کند. ضمنا، این الگوریتم از فضای جستجو زمانی که محدوده در بازه [-۱,۱] بهره‌برداری می‌کند.

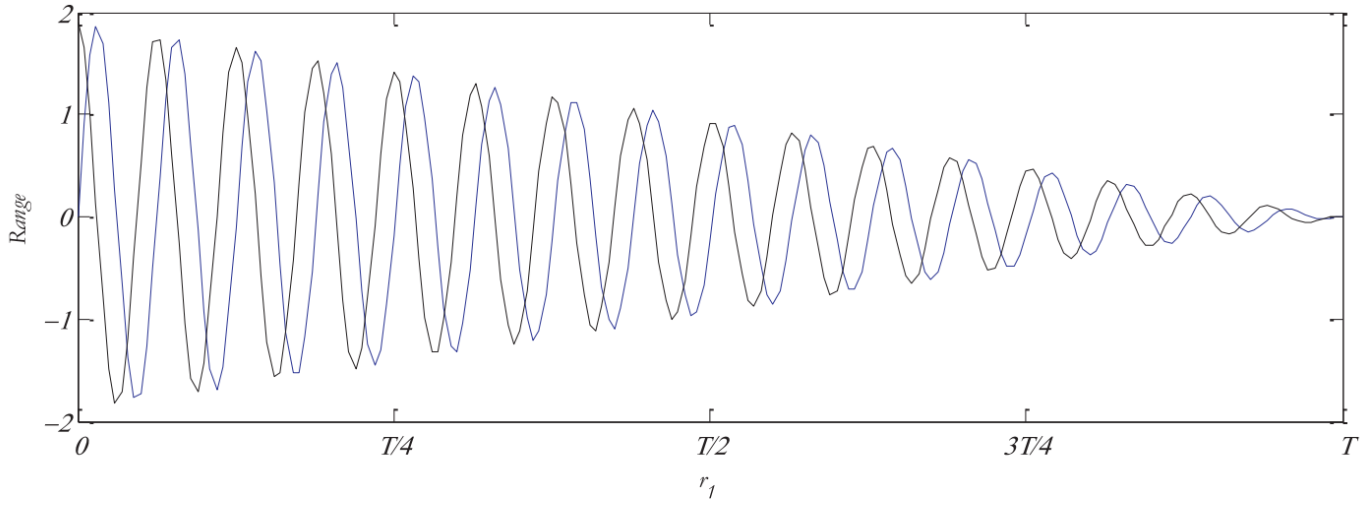


Fig. 8. Decreasing pattern for range of sine and cosine ($a = 3$).

با همه این‌ها، شبه کد الگوریتم SCA در شکل ۹ نمایش داده شده است. این شکل نشان می‌دهد که الگوریتم SCA فرآیند بهینه‌سازی را با مجموعه‌ای از پاسخ‌های تصادفی آغاز می‌کند. سپس الگوریتم بهترین پاسخ‌های به‌دست آمده را ذخیره می‌کند، آن را به عنوان نقطه مقصد اختصاص می‌دهد و پاسخ‌های دیگر را با توجه به آن به‌روز رسانی می‌کند. در همین حال، محدوده توابع سینوس و کسینوس برای تاکید بر بهره‌برداری از فضای جستجو با افزایش شمارنده تکرار، به روز می‌شوند. الگوریتم SCA زمانی فرآیند بهینه‌سازی را خاتمه می‌دهد که شمارنده تکرار به طور پیشفرض بالاتر از حداکثر تعداد تکرارها باشد. با این حل، هر شرط خاتمه دیگری مانند حداکثر تعداد توابع ارزیابی یا دقت بهینه سراسری به‌دست آمده را می‌توان در نظر گرفت.

```
Initialize a set of search agents (solutions) (X)
Do
   Evaluate each of the search agents by the objective function
   Update the best solution obtained so far (P=X*)
   Update r1, r2, r3, and r4
   Update the position of search agents using Eq. (3.3)
While(t< maximum number of iterations)
Return the best solution obtained so far as the global optimum
```

Fig. 9. General steps of the SCA Algorithm.

با عملگرهای بالا، با توجه به دلایل زیر الگوریتم مطرح شده از نظر تئوری قادر به تشخیص بهینه سراسری مسائل بهینه‌سازی می‌باشد:

- الگوریتم SCA مجموعه ای از پاسخ‌های تصادفی را برای مسئله مفروض میسازد و بهبود می‌بخشد، بنابر این ذاتا از کاوش بالا و اجتناب از بهینه محلی در مقایسه با الگوریتم‌های مبتنی بر فرد سود می‌برد
- هنگامی که توابع سیتوس و کسینوس مقداری بزرگتر از ۱ یا کمتر از -۱ را برمی‌گرداند، مناطق مختلف فضای جستجو کاوش می‌شوند.
- مناطق امیوار کننده فضای جستجو زمانی مورد سوء استفاده قرار می‌گیرند که سینوس و کسینوس مقدار بین -۱ و ۱ را برمی‌گرداند.

- الگوریتم SCA با استفاده از محدوده تطبیقی در توابع سیسنوس و کسینوس به آرامی از اکتشاف به بهره‌برداری می‌گذرد.
- بهترین تقریب از بهینه سراسری در یک متغیر به عنوان نقطه مقصد ذخیره می‌شود و هرگز در طول بهینه‌سازی گم نمی‌شود.
- از آنجایی که پاسخ‌ها همیشه موقعیت‌های خود را حول بهترین پاسخ به‌دست آمده تا کنون به‌روز می‌کنند، در طول بهینه‌سازی، تمایل به سمت بهترین مناطق فضاهای جستجو وجود دارد.
- چون الگوریتم پیشنهادی میئله بهینه‌سازی را به عنوان جعبه سیاه در نظر می‌گیرد، به آسانی با مسائل در زمینه های مختلف با توجه به فرمول‌بندی صحیح مسئله قابل تلفیق است.

بخش‌های بعدی طیف گسترده‌ای از مسائل ازمون و یک مطالعه موردی واقعی را برای بررسی، تجزیه و تحلیل و تایید اثربخشی الگوریتم SCA به کار می‌گیرند.

۴. نتایج و بررسی

در زمینه بهینه‌سازی با استفاده از الگوریتم‌های فراابتکاری و تکاملی، باید از چندین مورد آزمایش برای تایید عملکرد یک الگوریتم استفاده کرد. این به دلیل ماهیت تصادفی این الگوریتم‌ها است که در آن باید مجموعه‌ای مناسب و کافی از توابع تست و موضوعات مطالعه (case study) به کار گرفته شود تا به طور قطع اطمینان حاصل شود که بهترین نتایج حاصله شانسی اتفاق نمی‌افتد. با این حال، هیچ تعریف روشنی از مناسب بودن برای مجموعه‌ای از موضوعات مطالعه معیار وجود ندارد. لذا، محققان سعی می‌کنند الگوریتم‌های خود را بر روی بسیاری از موارد تست آزمایش کنند. ایم مقاله همچنین چندین تابع تست با ویژگی‌های مختلف را به کار می‌گیرد. بعدا، یک مسئله واقعی چالش بر انگیز دینامیک سیالات محاسباتی (CFD) توسط الگوریتم SCA نیز حل می‌شود.

مجموعه موضوعات مطالعاتی به کار گرفته شده شامل سه خانواده از توابع تست است: توابع آزمون تک‌وجهی، چندوجهی و ترکیبی [۶۳-۶۶]. فرمول ریاضی این توابع تست در پیوست موجود است. اولین خانواده از توابع تست هیچ بهینه محلی ندارند و تنها یک بهینه سراسری وجود دارد. این باعث می‌شود آنها برای آزمایش سرعت همگرایی و بهره‌برداری از الگوریتم‌ها بسیار مناسب باشد. با این حال، گروه دوم توابع تست، علاوه بر بهینه سراسری، چندین پاسخ محلی نیز دارد. این ویژگی‌ها برای آزمایش اجتناب از بهینه محلی و توانایی اکتشافی یک الگوریتم مفید هستند. در نهایت، توابع تست ترکیبی، چرخش داده شده، جابه‌جا شده، جانبدارانه، و نسخه ترکیبی از چندین تابع تست تک‌وجهی و چندوجهی هستند.

برای حل توابع تست فوق الذکر، در مجموع ۳ عامل جستجو مجاز به تعیین بهینه سراسری در بیش از ۵۰۰ تکرار هستند. الگوریتم SCA با الگوریتم کرم شب تاب (FA) [۶۷] ، الگوریتم خفاش (BA) [۶۸]، الگوریتم گرده افشانی گل (FPA) [۶۹]، الگوریتم جستجوی گرانشی (GSA) [۵۴]، PSO و GA برای صحت نتایج، مقایسه شده است. از آنجایی که نتایج یک اجرا ممکن است به دلیل ماهیت تصادفی فراابتکاری غیرقابل اعتماد باشد، همه الگوریتم‌ها ۳۰ بار اجرا می‌شوند و نتایج آماری (میانگین و انحراف استاندارد) جمع‌آوری و در جدول ۱ گزارش شده است. توجه داشته باشید که نتایج در [۰,۱] برای مقایسه نتایج همه توابع تست نرمال می‌شوند. برای تصمیم‌گیری در مورد اهمیت نتایج، یک آزمون آماری غیر پارامتری به نام آزمون رتبه بندی ویلکاکسون نیز انجام می‌شود. مقادیر p بدست آمده از این آزمون آماری در جدول ۲ گزارش شده است.

Table 1
Results on benchmark functions.

F	SCA		PSO		GA		BA		FPA		FA		GSA
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*	*ave*
F1	0.0000	0.0000	0.0003	0.0011	0.8078	0.4393	1.0000	1.0000	0.2111	0.0717	0.0004	0.0002	0.0000
F2	0.0000	0.0001	0.0693	0.2164	0.5406	0.2363	1.0000	1.0000	0.9190	0.7804	0.0177	0.0179	0.0100
F3	0.0371	0.1372	0.0157	0.0158	0.5323	0.2423	1.0000	1.0000	0.2016	0.1225	0.0000	0.0004	0.0016
F4	0.0965	0.5823	0.0936	0.4282	0.8837	0.7528	1.0000	1.0000	0.8160	0.5618	0.0000	0.0107	0.1177
F5	0.0005	0.0017	0.0000	0.0000	0.6677	0.4334	1.0000	1.0000	0.0813	0.0426	0.0000	0.0000	0.0000
F6	0.0002	0.0001	0.0004	0.0033	0.7618	0.7443	1.0000	1.0000	0.2168	0.1742	0.0004	0.0002	0.0000
F7	0.0000	0.0014	0.0398	0.0634	0.5080	0.1125	1.0000	1.0000	0.3587	0.2104	0.0009	0.0022	0.0021
F8	1.0000	0.0036	1.0000	0.0036	1.0000	0.0055	0.0000	1.0000	1.0000	0.0029	1.0000	0.0168	1.0000
F9	0.0000	0.7303	0.3582	0.8795	1.0000	0.6881	0.4248	1.0000	0.8714	0.8665	0.0190	0.3298	0.0222
F10	0.3804	1.0000	0.1045	0.0541	0.8323	0.0686	0.8205	0.0796	1.0000	0.0162	0.0000	0.0079	0.1569
F11	0.0000	0.0051	0.0521	0.0448	0.7679	0.2776	1.0000	1.0000	0.2678	0.0706	0.0074	0.0001	0.4011
F12	0.0000	0.0000	0.0000	0.0000	0.4573	0.4222	1.0000	1.0000	0.0008	0.0015	0.0000	0.0000	0.0000
F13	0.0000	0.0000	0.0000	0.0000	0.6554	0.8209	1.0000	1.0000	0.0187	0.0375	0.0000	0.0000	0.0000
F14	0.3908	0.1924	0.1816	1.0000	0.4201	0.1610	1.0000	0.6977	0.3786	0.1716	0.0000	0.9571	0.0961
F15	0.0230	0.0676	0.3016	1.0000	0.0000	0.0779	1.0000	0.7614	0.2235	0.4252	0.4395	0.9135	0.2926
F16	0.0497	0.4921	0.0427	0.7228	0.0000	0.2422	0.3572	0.7629	0.2652	0.6012	0.5298	1.0000	1.0000
F17	0.0000	0.1105	0.0249	1.0000	0.1093	0.1873	0.8189	0.7754	0.5197	0.4847	0.7093	0.8842	0.7887
F18	0.0129	0.0134	0.1772	0.4289	0.0000	0.0538	1.0000	0.2855	0.1310	0.0429	0.0723	0.2069	0.8018
F19	0.0000	0.2001	0.7727	1.0000	0.0192	0.0312	1.0000	0.2142	0.3192	0.4635	0.8176	0.7924	0.9950
Sum	1.9911	3.5379	3.2346	6.8619	9.9634	5.9972	16.4214	15.5767	7.8004	5.1479	3.6143	5.1403	5.6858

Table 2
p-Values of the Wilcoxon ranksum test over all runs ($p >= 0.05$ have been underlined).

F	SCA	PSO	GA	BA	FPA	FA	GSA
F1	N/A	0.002165	0.002165	0.002165	0.002165	0.002165	0.002165
F2	N/A	0.002165	0.002165	0.002165	0.002165	0.002165	0.002165
F3	0.004329	0.002165	0.002165	0.002165	0.002165	N/A	0.008658
F4	0.002165	0.002165	0.002165	0.002165	0.002165	N/A	0.002165
F5	N/A	0.002165	0.002165	0.002165	0.002165	0.002165	0.681818
F6	0.002165	0.002165	0.002165	0.002165	0.002165	0.002165	N/A
F7	N/A	0.002165	0.002165	0.002165	0.002165	0.24026	0.002165
F8	0.002165	0.002165	0.002165	N/A	0.002165	0.002165	0.002165
F9	N/A	0.002165	0.002165	0.002165	0.002165	0.484848	0.818182
F10	1.000000	0.002165	0.002165	0.002165	0.002165	N/A	0.093074
F11	N/A	0.002165	0.002165	0.002165	0.002165	0.002165	0.002165
F12	N/A	0.015152	0.002165	0.002165	0.002165	0.064935	0.064935
F13	0.002165	0.002165	0.002165	0.002165	0.002165	N/A	0.393939
F14	0.064935	0.588745	0.064935	0.041126	0.064935	N/A	0.132035
F15	0.179654	0.064935	N/A	0.002165	0.008658	0.008658	0.002165
F16	0.818182	0.937229	N/A	0.002165	0.002165	0.002165	0.002165
F17	N/A	1.000000	0.015152	0.002165	0.002165	0.002165	0.002165
F18	0.818182	0.393939	N/A	0.002165	0.002165	0.699134	0.025974
F19	N/A	0.064935	0.699134	0.002165	0.041126	0.041126	0.002165

نتایج جدول ۱ نشان می‌دهد که الگوریتم SCA در اکثر موارد تست بهتر از سایرین عمل می‌کند. اولا، الگوریتم SCA در ۳ مورد از ۶ مورد توابع تست تک‌وجهی نتایج برتری را نشان می‌دهد. مقادیر p در جدول ۲ نشان می‌دهد که این برتری از نظر آماری معنی‌دار است. با توجه به ویژگی‌های توابع تست تک‌وجهی، این نتایج قویا نشان می‌دهد که الگوریتم SCA بهره‌برداری و همگرایی بالایی دارد. ثانیا، جدول ۱ نشان می‌دهد که الگوریتم SCA از همه الگوریتم‌های به کار رفته در اکثر توابع تست چندوجهی (F۱۱, F۹, F۷ و F۱۲) بهتر عمل می‌کند. مقادیر p در جدول ۲ نیز از نظر آماری نتایج بهتر SCA را پشتیبانی می‌کند. با بررسی نتایج این جدول، الگوریتم SCA مقادیر p بیشتر از ۰/۰۵ را برای

بقیه توابع تست ارائه می‌دهد که نشان می‌دهد این الگوریتم بسیار رقابتی است. این نتایج ثابت می‌کند که الگوریتم SCA از اکتشاف بالا و اجتناب از بهینه محلی سود می‌برد. در نهایت، نتایج الگوریتم پیشنهادی بر روی توابع تست ترکیبی در جدول ۱ و ۲، شایستگی SCA را در حل مسائل با فضاهای جستجوی چالش برانگیز نشان می‌دهد. با توجه به نرمال بودن نتایج، می‌توان عملکرد کلی الگوریتم‌ها را نیز مقایسه کرد. ردیف آخر جدول ۱ جمع میانگین و انحراف معیار الگوریتم‌ها را در تمام توابع تست ارائه می‌دهد. بدیهی است که SCA حداقل مقادیر را برای هر دو ave و std نشان، می‌دهد، و ثابت می‌کند که این الگوریتم به طور قابل اعتمادی در کل عملکرد بهتری از سایر الگوریتم‌ها دارد.

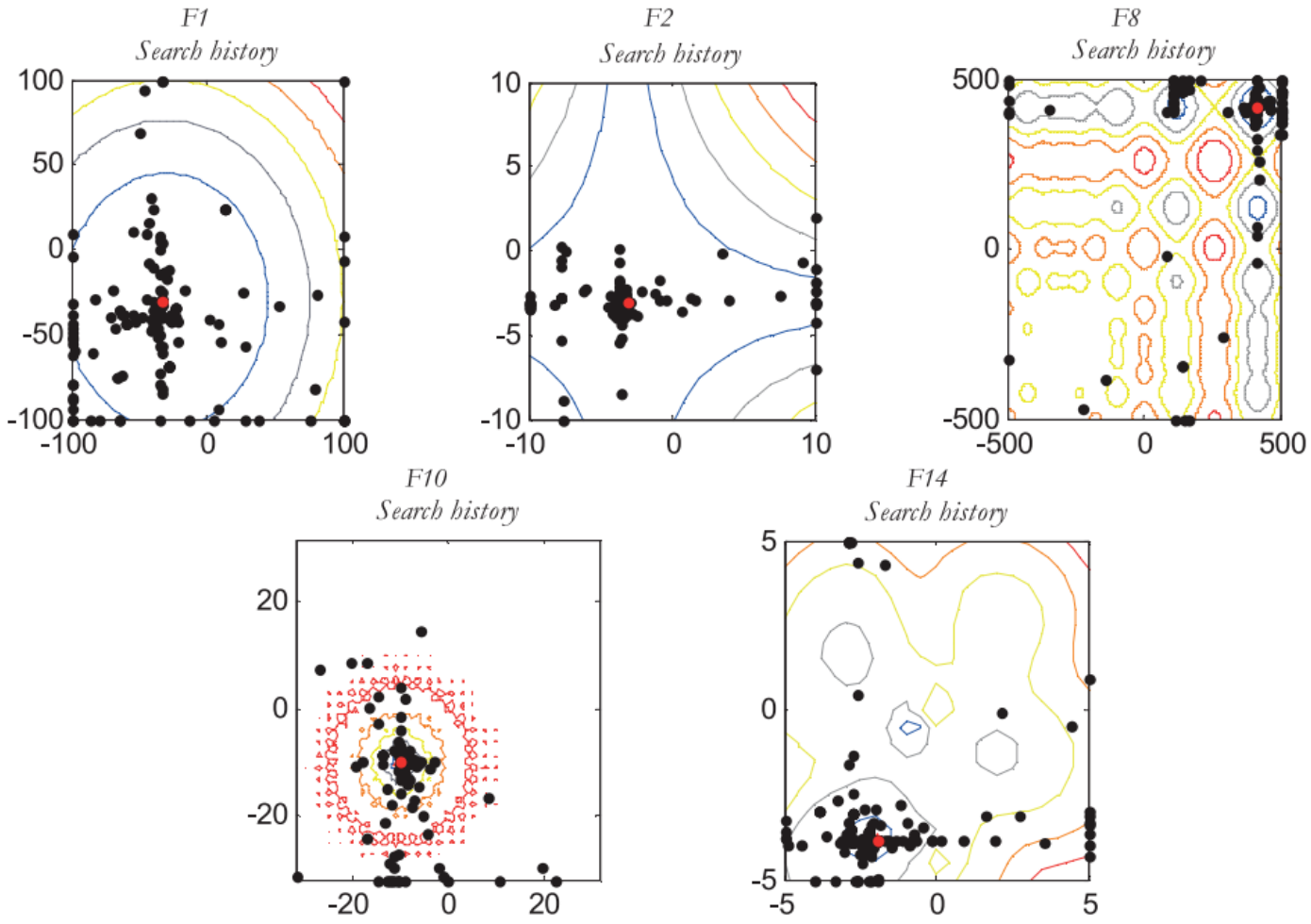


Fig. 10. Search history of search agents when solving the test problems.

اگرچه نتایج بخث شده در بالا کارایی بالای الگوریتم SCA را اثبات و تایید می‌کند، آزمایش‌های متعدد دیگری نیز وجود دارد که باید انجام شود تا با اطمینان عملکرد این الگوریتم در حل مسائل واقعی تایید شود. به عبارت دیگر، رفتار عوامل جستجو در طول بهینه‌سازی باید تحت نظارت قرار گیرد تا مشاهده شود: چگونه آنها در فضای جستجوی حرکت می‌کنند، اگر در مراحل اولیه بهینه‌سازی برای کاوش در فضای جستجو با تغییرات ناگهانی مواجه شوند، اگر تغییرات کوچکی در مراحل نهایی تکرار برای بهره‌برداری از فضای جستجو داشته باشند، چگونه به سمت مناطق امیدوارکننده فضای جستجو همگرا می‌شوند، چگونه پاسخ‌های تصادفی اولیه خود را بهبود می‌بخشد، و چگونه مقادیر تناسب خود را در طول تکرارها بهبود می‌بخشد. به منظور مشاهده رفتار عوامل جستجو، نسخه دو بعدی توابع تست توسط ۴ عامل جستجو حل می‌شود. توجه داشته باشید که بهینه برخی از توابع تست به مکان‌هایی غیر از مبدا منتقل می‌شود تا بسترهای آزمون چالش‌برانگیزتری ارائه شود. تاریخچه جستجوی عوامل جستجو در شکل ۱۰ نشان داده شده است. این شکل نشان می‌دهد که الگوریتم SCA در اطراف مناطق امیدوار کننده فضای جستجو، جتسجو می‌کند. توزیع نقاط نمونه‌برداری شده در اطراف بهینه سراسری به طور قابل ملاحظه‌ای بالاست، که نشان می‌دهد

که الگوریتم SCA علاوه بر اکتشاف، از امیدوارکننده‌ترین منطقه فضای جستجو نیز بهره‌برداری می‌کند. با این حال، از این شکل مشخص نیست که آیا عوامل جستجو ابتدا شروع به اکتشاف یا بهره‌برداری می‌کنند. برای مشاهده این موضوع، شکل ۱۱ در این رابطه ارائه شده است که نوسانات بعد اول را در اولین عامل جستجو نشان می‌دهد.

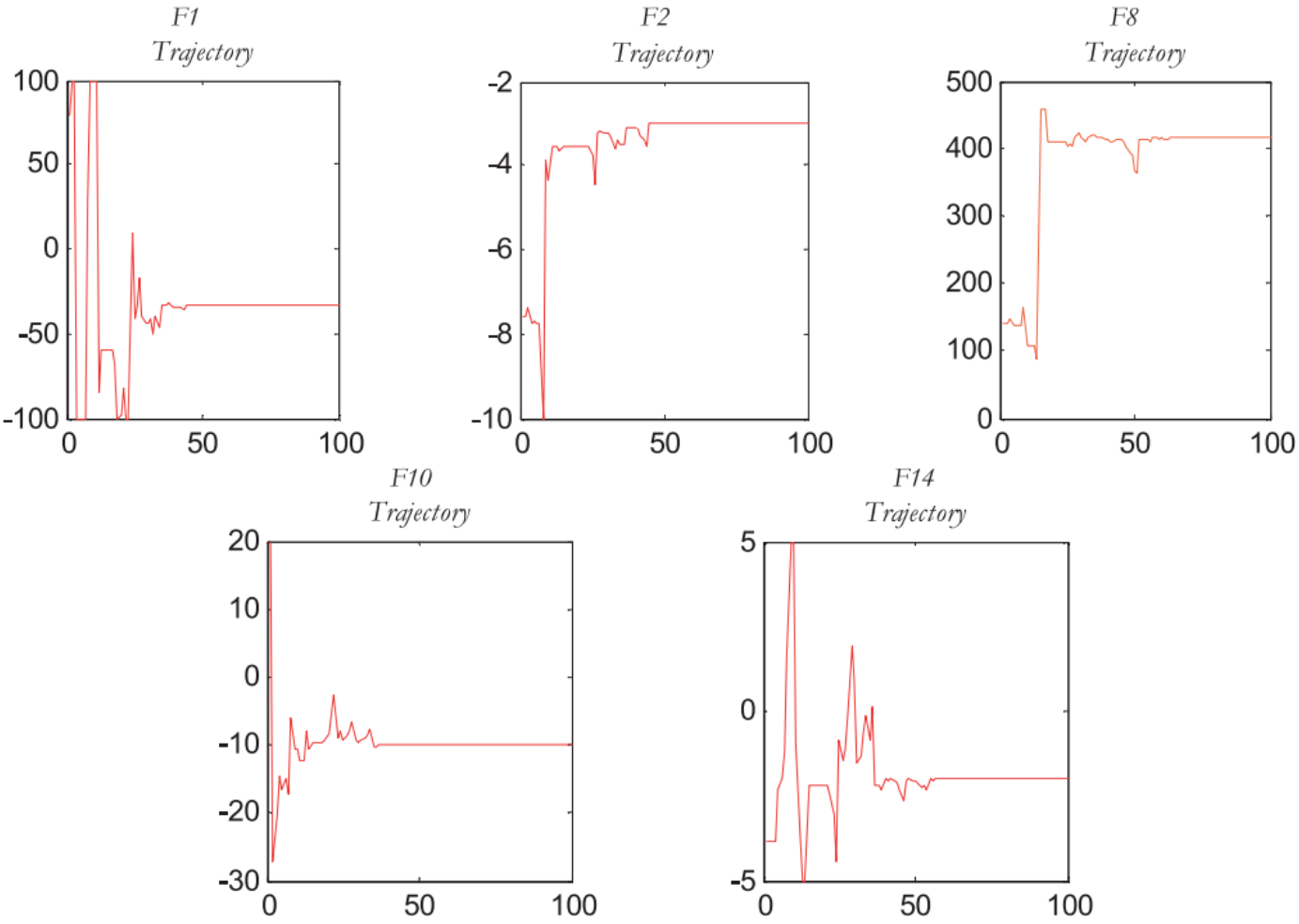


Fig. 11. Trajectory of the first variable of the first search agent when solving the test problems.

شکل ۱۱ نشان می‌دهد که عوامل جستجو در مراحل اولیه بهینه‌سازی با نوسانات ناگهانی روبرو هستند. با این حال، تغییرات ناگهانی به تدریج در طول تکرار کاهش می‌یابد. این تایید می‌کند که عوامل جستجو ابتدا فضای جستجو را کاوش می‌کنند و سپس حول بهترین پاسخ به‌دست آمده در مرحله اکتشاف همگرا می‌شوند. در اینجا یک سوال وجود دارد که چگونه می‌توان مطئن شد که همه عوامل جستجو در طول بهینه‌سازی با وجود تغییرات سریع و مداوم در شکل ۱۱ بهبود یافته اند. به منظور تایید بهبود همه پاسخ‌ها، میانگین تناسب همه عوامل جستجو در طول بهینه‌سازی در شکل ۱۲ نشان داده شده است.

این شکل نشان می‌دهد که میانگین برازش همه عوامل جستجو در طول تکرارها میل به کاهش یافتن دارند. الگوی جالبی که می‌توان در این شکل مشاهده نمود، نوسان زیاد میانگین برازش در مرحله اکتشاف (تا نزدیک به ۵۰ تکرار) و تغییرات کم در میانگین برازش در مرحله بره‌برداری (پس از تکرار ۵۰) است. بدتر شدن برازش برخی از عوامل جستجو در مرحله اکتشاف که در آن الگوریتم SCA باید مناطق امیدوارکننده فضای جستجو را کشف کند، اجتناب ناپذیر است. با این حال، الگوهای مشاهده شده در شکل ۱۲ نشان می‌دهد که برازش عوامل جستجو در طول تکرارها

رفتار نزولی دارد. این ثابت می‌کند که الگوریتم SCA می‌تواند در نهایت برازش پاسخ‌های تصادفی اولیه را برای یک مسئله بهینه‌سازی معین بهبود بخشد.

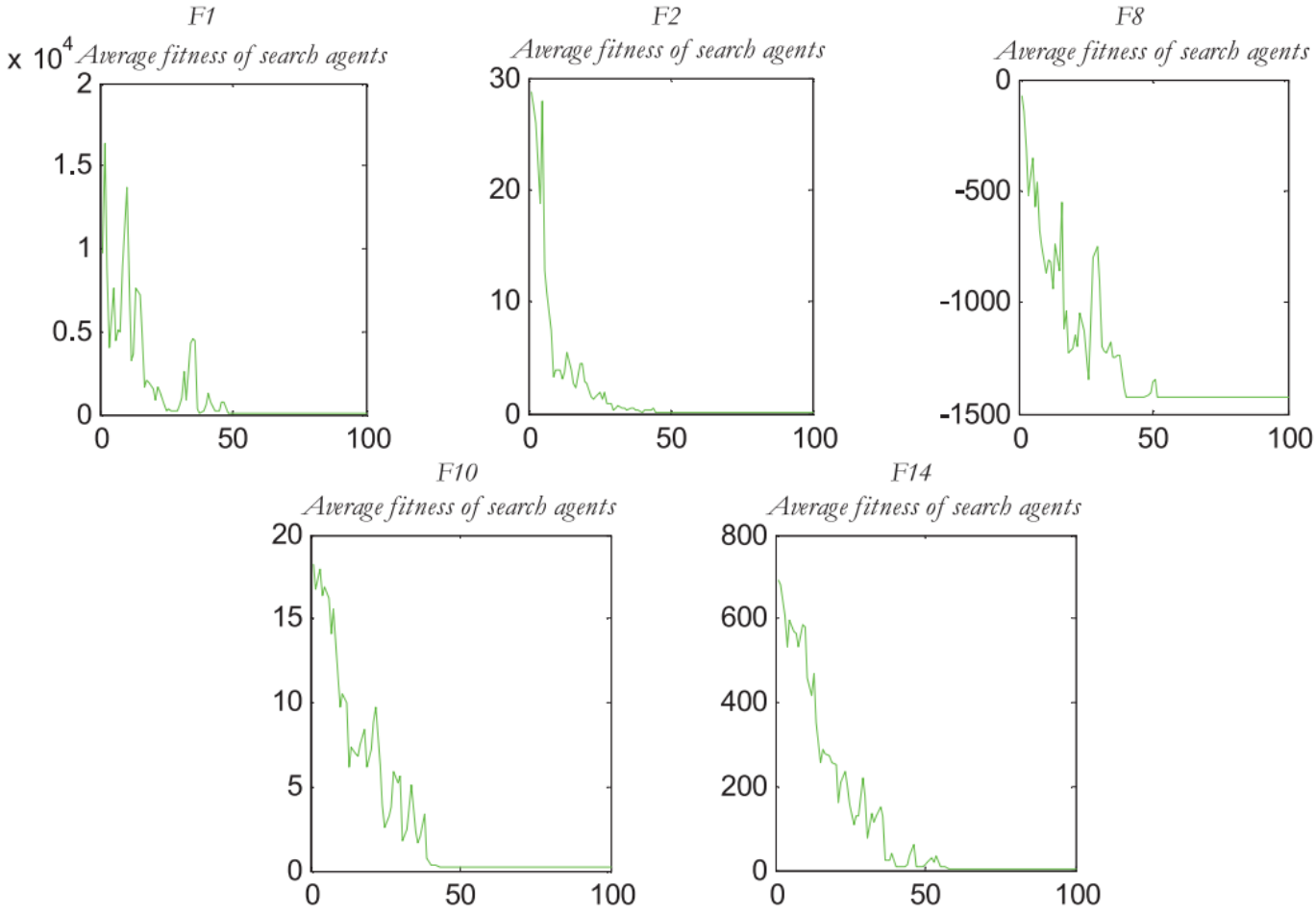


Fig. 12. Average fitness of search agents during optimization.

در پاراگراف‌های قبلی، ادعا شد که عوامل جستجوی الگوریتم SCA تمایل دارند مناطق امیدوارکننده فضای جستجو را کاوش کنند و در نهایت از بهترین آنها بهره‌برداری کنند. با این حال، رفتار همگرایی الگوریتم مشاهده و تایید نشد. اگرچه این را میتوان به طور غیر مستقیم از مسیر و برازش متوسط استنباط کرد، منحنی‌های همگرایی SCA در شکل ۱۳ نشان داده شده است.

این شکل بهترین پاسخی را که تا کنون در طول بهینه‌سازی به‌دست آمده را نشان می‌دهد. روند نزولی در منحنی‌های همگرایی SCA در تمام توابع تست بررسی شده کاملا مشهود است. این قویا نشان دهنده توانایی الگوریتم SCA در به‌دست آوردن تقریب بهتر از بهینه سراسری در طول دوره تکرار است.

نتایج و بحث‌های این بخش ثابت می‌کند که الگوریتم پیشنهادی SCA قادر به تعیین بهینه سراسری توابع تست است. اگرچه در اینجا می‌توان ادعا کرد که این الگوریتم قادر به تقریب بهینه سراسری مسائل واقعی است، تفاوت اصلی بین مسائل واقعی و توابع معیار وجود دارد. شکل فضای جستجو و مکان بهینه سراسری توابع تست مشخص است، در حالی که مسائل واقعی عمدتا ناشناخته هستند. علاوه بر این، مسائل واقعی با تعداد زادی از قیود معادله‌ای و نامعادله‌ای همراه است. بنابراین، نیاز به بررسی عملکرد الگوریتم SCA در حل حداقل یک مسئله مقید چالش‌برانگیز واقعی با بهینه سراسری و فضای جستجوی ناشناخته وجود دارد. این انگیزه بخش بعدی است که در آن مقطع دو بعدی بال هواپیما توسط SCA بهینه‌سازی می‌شود تا عملکرد آن در عمل تایید شود.

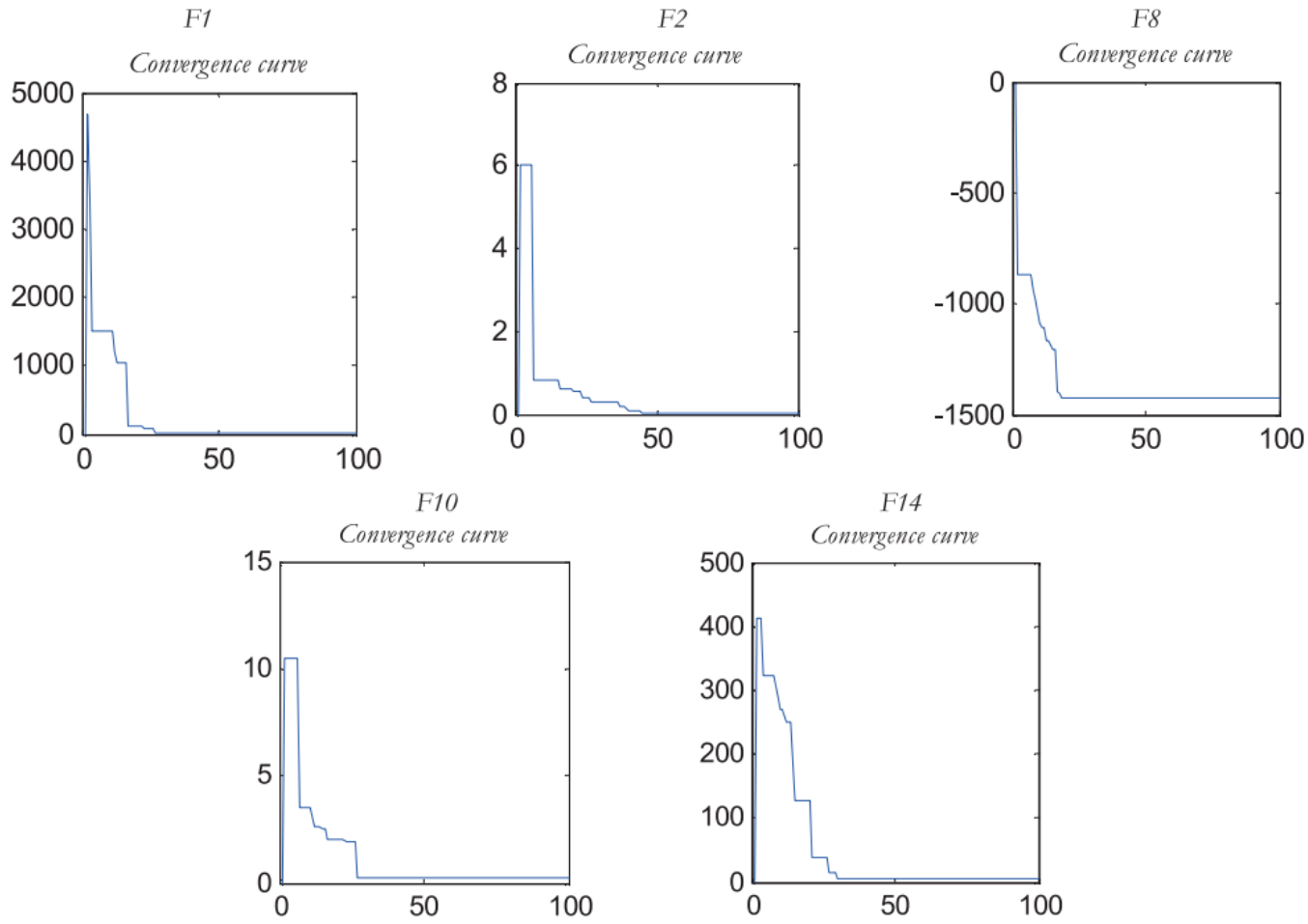


Fig. 13. Convergence curve (best solution in each iteration) of the SCA algorithm.

۵. طراحی بال هواپیما با استفاده از SCA

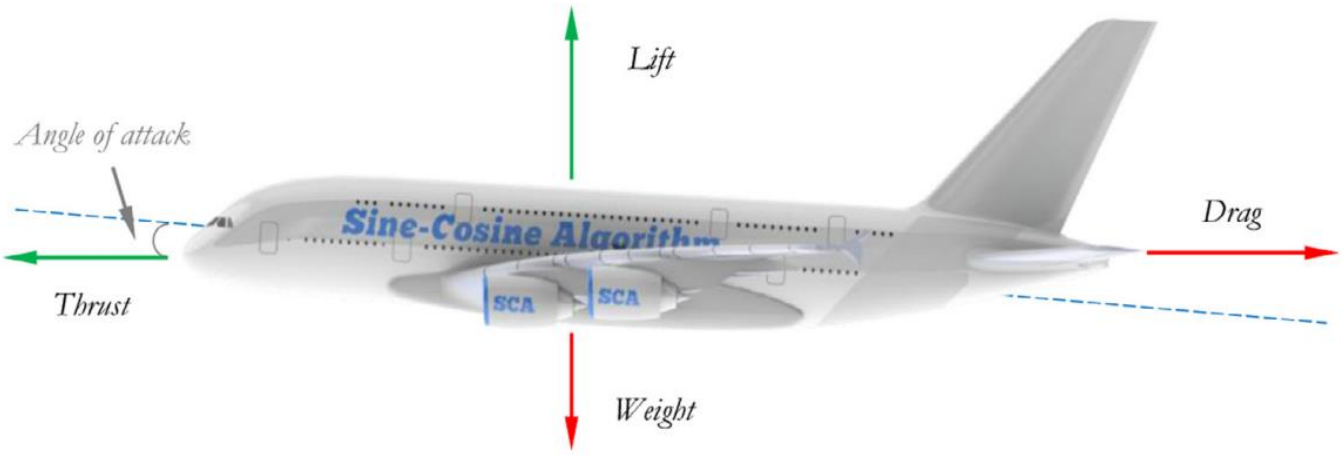


Fig. 14. Different forces that apply to an airplane.

مسئله‌ای که در این بخش مورد بررسی قرار می‌گیرد طراحی ایرفویل است. در این مسئله دو هدف وجود دارد: لیفت (بالا رفتن و برخاستن) در مقابل درگ (به عقب کشیده شدن). دو نیرو در شکل ۱۴ نشان داده شده است. مشاهده می‌شود که لیفت زمانی است که نیروی رانش (Thrust) به نیروی عمودی تبدیل می‌شود که باعث پرواز هواپیما می‌شود. اما درگ نیروی مخالفی است که به بال وارد می‌شود و باعث کاهش سرعت می‌شود. لیفت و درگ در تضاد هستند، به این معنی که افزایش یکی منجر به کاهش دیگری می‌شود. در یک هواپیمای واقعی هر دوی این نیروها در موقعیت‌های مختلف مطلوب هستند. هنگامی که هواپیما در حال برخاستن، صعود و کروز (پیمایش مسیر

مستقیم) است، حداکثر لیفت و حداقل درگ مثمر ثمر است. هنگام کاهش ارتفاع، هنگام فرود و نشستن روی زمین نیروی درگ برای کاهش سرعت وسیله نقلیه مهم می‌شود. در این بخش فقط درگ در نظر گرفته شده است، لذا هدف اصلی به حداقل رساندن این نیرو است. به عبارت دیگر، این بخش از الگوریتم SCA برای تعریف بهترین شکل برای بال استفاده می‌کند تا درگ را به حداقل برساند.

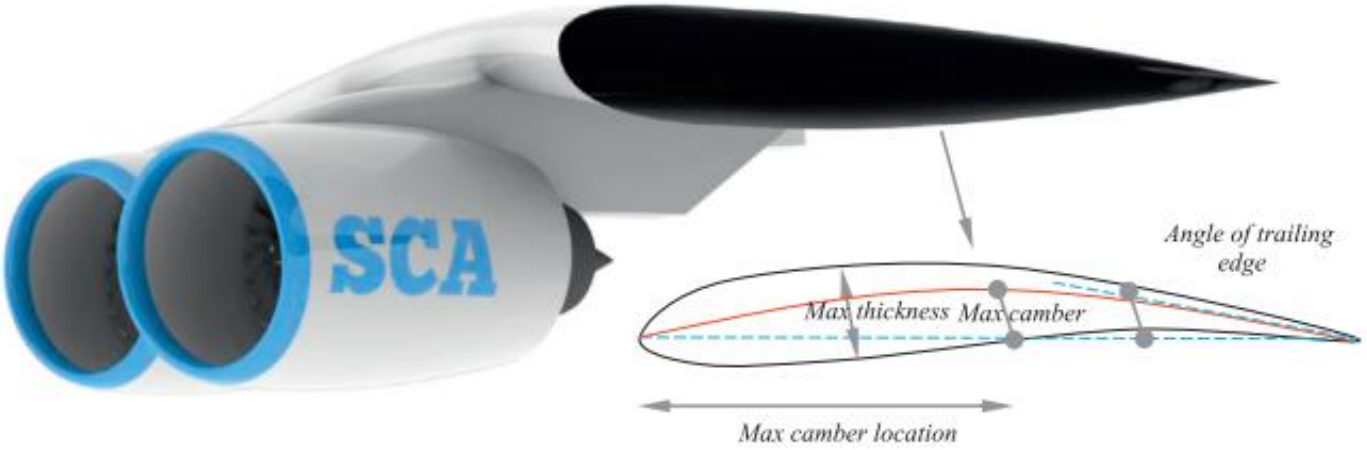


Fig. 15. Cross section of a real with a 2D airfoil.

برای طراحی یک بال هواپیما باید چندین جزء را در نظر گرفت: شکل سطح مقطع بال (ایرفویل)، شکل کلی بال، فلپ ها، چارچوب‌های داخلی و موقعیت موتورها. این مقاله روی طراحی یک ایرفویل دو بعدی متمرکز شده است که جزء اصلی و ضروری یک بال است. شکل یک ایرفویل دو بعدی در شکل ۱۵ نشان داده شده است.

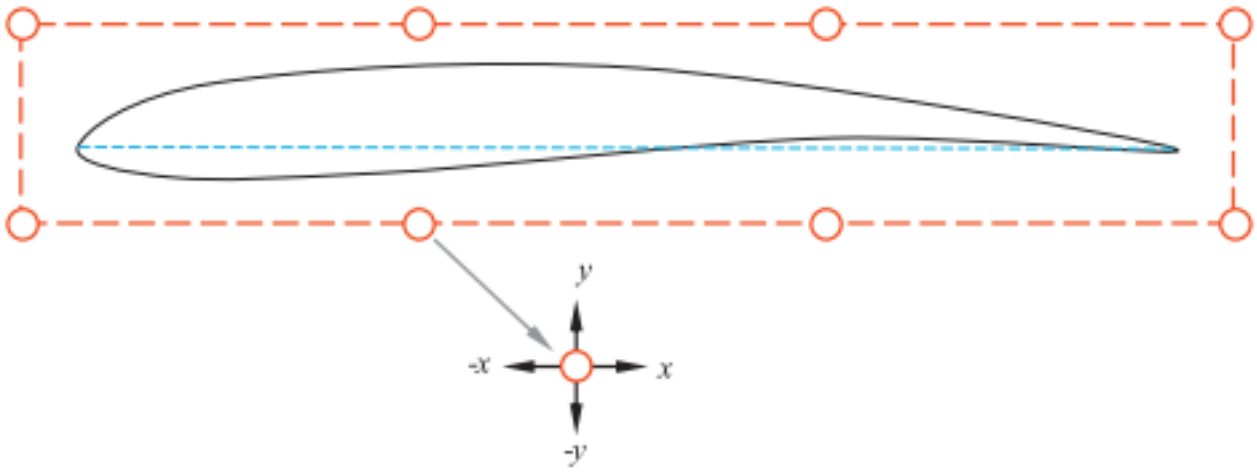


Fig. 16. B-spline for the problem of Airfoil design.

نسخه‌های مختلفی از این مسئله در مقالات تحقیقاتی از نظر پارامترهای طراحی وجود دارد. در این مقاله از B-Spline برای تعریف شکل ایرفویل استفاده شده است. هماهنطور که در شکل ۱۶ نشان داده شده است، هشت پارامتر کنترلی وجود دارد که یکی از نقاط اصلی ثابت است. با این حا، بقیه پارامترهای کنترل کننده اجازه دارند در هر دو جهت محور x و y حرکت کنند. لذا، در مجموع ۱۴ پارامتر (۲ × ۷) وجود دارد که موقعیت‌های x و y هفت نقطه کنترلی هستند. مسئله طراحی ایرفویل برای الگوریتم SCA به صورت زیر فرموله شده است:

$$\begin{aligned} &Minimize: && F(\vec{x},\ \vec{y}) = C_d(\vec{x},\ \vec{y}) \\ &Subject\ to: && -1 \le \vec{x},\ \vec{y} \le 1,\ satisfaction\ of\ CO\ set \end{aligned} \tag{5.1}$$

که در آن $\vec{x}= \{x_1, x_2, \dots, x_7\}$ و $\vec{y}= \{y_1, y_2, \dots, y_7\}$ و مجموعه CO شامل قیود زیادی می‌شود از قبیل : حداقل نازک بودن، حداکثر ضخامت و غیره.

یک نرم‌افزار رایگان به نام XFoil برای محاسبه درگ استفاده می‌شود [۷۵]. در معادله (۵.۱) قابل مشاهده است که مسئله چندین قید دارد. به طور کلی، مسئله محاسباتی دینامیک سیالات (CFD) بسیار محدود و مقید است، که آنها را بسیار چالش برانگیز می‌کند. برای حل چنین مسائلی، یک الگوریتم بهینه‌سازی باید مجهز به روش مدیریت قیود مناسبی باشد. در این حوزه تحقیقاتی رویکردهای مختلفی برای مقابله با قیود وجود دارد که توابع جریمه ساده‌ترین آنها هستند. در چنین روشهایی تابع هدف اصلی با توجه به سطح نقض محدودیت‌ها توسط تابع جریمه، جریمه می‌شود. سایر روشهای قدرتمند مدیریت قیود را می‌توان در [۷۰-۷۳] یافت. خوانندگان علاقه‌مند به بررسی تاریخچه جامع توسط Coello Coello [۷۴] ارجاع داده می‌شوند. در این مقاله از تابع جریمه زیر استفاده شده است که F را متناسب با سطح تخلف جریمه می‌کند:

$$F(\vec{x},\ \vec{y}) = F(\vec{x},\ \vec{y}) + p\sum_{i=1}^{3} P_i \qquad (5.2)$$

که در آن p یک ثابت است و P_i اندازه تخلف در قید iام در مجموعه قیود CO در معادله (۵.۱) است.

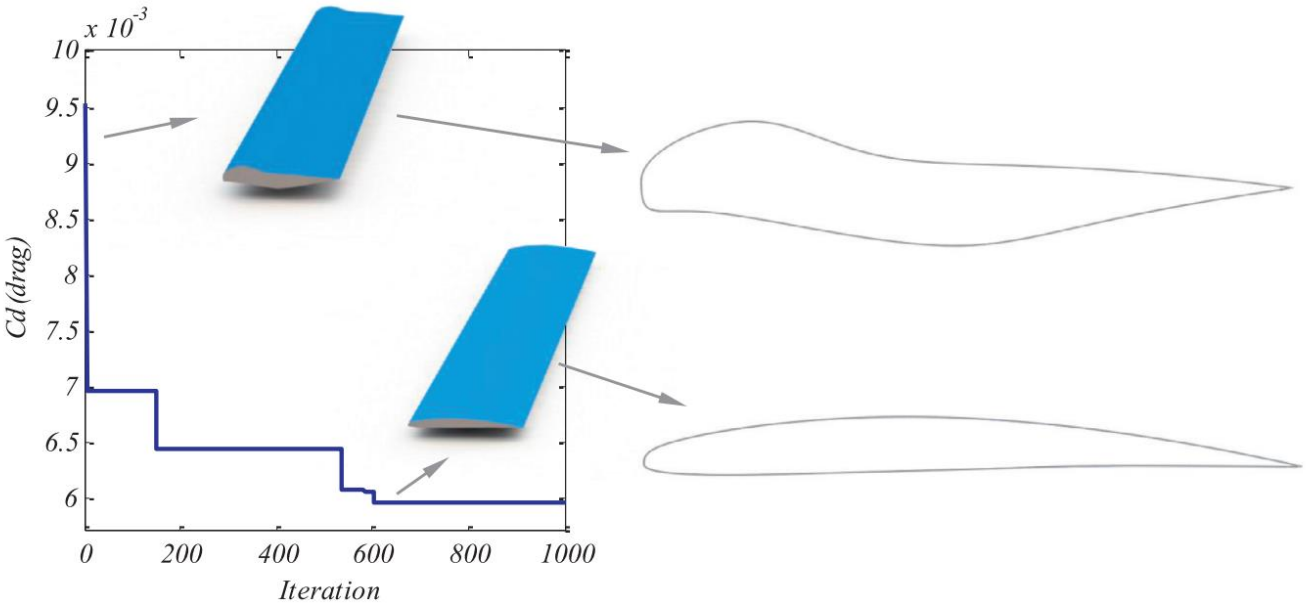


Fig. 17. Convergence curve of the SCA on the airfoil design problem, initial aifoil, and optimized airfoil.

برای حل این مسئله، ۳۰ عامل جستجو به کار گرفته شده است و به منظور تشخیص شکل بهینه برای ایرفویل در طول ۱۰۰۰ تکرار اجازه داده شده است. این الگوریتم ۴ بار اجرا شده است و بهترین نتیجه در شکل ۱۷ به نمایش درآمده است.

این شکل به وضوح نشان می‌دهد که الگوریتم SCA شکل تصادفی اولیه ایرفویل را بهبود می‌بخشد تا درگ را به حداقل برساند. این بهبود بسیار قابل توجه است، که در آن درگ از ۰/۰۰۹ به ۰/۰۰۶۱ کاهش یافت. این نتایج به خوبی نشان می‌دهد که الگوریتم SCA قادر به حل مسایل واقعی با فضاهای جستجوی ناشناخته، چالش برانگیز و مقید است. این بنا به دلایل مختلفی است. اولا، SCA یک الگوریتم مبتنی بر جمعیت است، بنابراین ذاتا از کاوش بالا و اجتناب از بهینه محلی سود می‌برد. این به الگوریتم SCA کمک میکند تا از تعداد زیادی پاسخ محلی در یک فضای

جستجوی واقعی اجتناب کند و مناطق مختلف را به طور گسترده بررسی کند. ثانیا، SCA با استفاده از مکانیسم تطبیقی برای برد توابع سینوس و کسینوس، به ارامی از اکتشاف به بهره‌برداری می‌گذرد. این امر سبب اجتناب از بهینه محلی در ابتدای بهینه‌سازی و همگرایی سریع به سمت امیدوارکننده‌ترین منطقه فضای جستجو در مراحل نهایی بهینه‌سازی می‌شود. ثالثا، SCA پاسخ‌ها را موظف می‌کند تا موقعیت‌های خود را در اطراف بهترین پاسخ به‌دست آمده تا نقطه مقصد به‌روز کنند. بنابراین، همواره در حین بهینه‌سازی، تمایل به سمت بهترین مناطق فضاهای جستجو وجود دارد و شانس بهبود پاسخ‌ها به طور قابل توجهی بالاست. در نهایت، الگوریتم SCA مسائل بهینه‌سازی را به عنوان جعبه‌های سیاه در نظر می‌گیرد، بنابر این با توجه به فرمول‌بندی مناسب مسئله، به آسانی با مسائل در زمینه‌های مختلف ترکیب نمی‌شود. علاوه بر این، استقلال مسئله به این الگوریتم اجازه می‌دهد تا به اطلاعات متحرک فضای جستجو نیاز نداشته باشد و با هر نوع توابع جریمه برای حل مسائل محدود کار کند.

۶. جمع بندی

در این مقاله یک الگوریتم بهینه‌سازی مبتنی بر جمعیت جدید به عنوان جایگزینی برای حل مسائل بهینه‌سازی در میان روشهای فعلی در این زمینه پیشنهاد شده است. در الگوریتم SCA پیشنهادی، پاسخ‌ها باید موقعیت‌های خود را با توجه به بهترین پاسخ به‌دست آمده تا نقطه مقصد به‌روز رسانی کنند. مدل ریاضی به‌روز رسانی موقعیت، پاسخ‌ها را به سمت خارج یا به سمت نقطه مقصد نوسان می‌کند تا اکتشاف و بهره‌برداری از فضای جستجو را تضمین کند. چندین متغیر تصادفی و تطبیقی نیز واگرایی و همگرایی عوامل جستجو را در الگوریتم SCA تسهیل کردند. برای محک زدن عملکرد SCA، چندین آزمایش انجام شد. در مرحله اول، مجموعه‌ای از موارد تست شناخته شده شامل توابع تک‌وجهی، چندوجهی و ترکیبی برای آزمایش اکتشاف، بهره‌برداری، اجتناب از بهینه محلی، و همگرایی الگوریتم پیشنهادی استفاده شد. ثانیا، نسخه‌های دو بعدی برخی از توابع تست توسط SCA انتخاب و دوباره حل شدند. چندین معیار عملکرد (سابقه جستجو، مسیر، میانگین برازش پاسخ‌ها و بهترین پاسخ در طول بهینه‌سازی) برای مشاهده کیفی و تایید عملکرد SCA استفاده شد. در نهایت، شکل یک ایرفویل دو بعدی (سطح مقطع بال هواپیما) توسط SCA به عنوان یک مورد مطالعه چالش برانگیز واقعی برای تایید و نشان دادن عملکرد این الگوریتم در حل مسائل واقعی با فضاهای جستجوی محدود و ناشناخته بهینه‌سازی شد.

نتایج توابع تست تک‌وجهی نشان داد که الگوریتم SCA به صورت قابل ملاحظه‌ای سریع‌تر از الگوریتم‌های PSO، GSA، FPA، BA، FA و GA همگرا می‌شود. رفتار مشابهی در توابع تست چندوجهی مشاهده شد که اکتشاف بالا و اجتناب از بهینه محلی الگوریتم پیشنهادی را ثابت کرد. با توجه به نتایج توابع تست ترکیبی، SCA گاهی اوقات از سایر الگوریتم‌ها بهتر عمل می‌کند، که نشان می‌دهد این الگوریتم همچنین قادر است اکتشاف و بهره‌برداری را با موفقیت متعادل کند تا بهینه سراسری توابع تست چالش برانگیز را تعیین کند. نتایج معیارهای عملکرد ثابت کرد که SCA نیاز به تغییر ناگهانی عامل جستجوی خود در مرحله اولیه بهینه‌سازی و به تدریج در مراحل نهایی بهینه‌سازی دارد. نتایج نشان داد که این رفتار باعث اکتشاف گسترده فضای جستجو و بهره‌برداری از امیدوار کننده‌ترین منطقه شده است. میانگین برازش پاسخ‌ها و منحنی‌های همگرایی نیز بهبود جمعیت تصادفی اولیه و بهترین پاسخ به‌دست آمده تا کنون توسط SCA را اثبات و تایید می‌کند. نتایج دو فاز اول آزمایشی ثابت کرد که SCA قادر به حل موفقیت‌آمیز مسائل آزمایشی است که شکل شناخته شده فضای جستجو را دارند. نتایج SCA روی مسئله طراحی آیروفویل نیز نشان داد که الگوریتم پتانسیل حل مسائل چالش برانگیز واقعی را نیز دارد. مشکل طراحی ایرفویل یک موضوع مطالعاتی بسیار مقید با فضای جستجوی کاملا ناشناخته بود. بنابر این، نتایج موضوع مطالعاتی واقعی، شایستگی SCA را در حل مسائل واقعی به خوبی نشان داد و تایید کرد.

با توجه به یافته‌های این مقاله و با مراجعه به قضیه NFL، می‌توان نتیجه گرفت که SCA می‌تواند جایگزین بسیار مناسبی در مقایسه با الگوریتم‌های موجود در این حوزه برای حل مسائل مختلف بهینه‌سازی باشد. از سوی دیگر، این الگوریتم ممکن است نتواند در مجموعه‌ای از مسائل خاص از الگوریتم‌های دیگر بهتر عمل کند، اما قطعا ارزش آزمایش و اعمال در مسائل در زمینه‌های مختلف را دارد. از این رو الگوریتم SCA به پژوهشگران حوزه‌های مختلف ارائه می‌شود. کدهای منبع این الگوریتم به صورت عمومی در آدرس http://www.alimirjalili.com/SCA.html در دسترس است.

این مقاله چندین جهت تحقیقاتی برای مطالعات آینده باز می‌کند. در مرحله اول، نسخه باینری و چند هدفه این الگوریتم را می‌توان برای حل مسائل به ترتیب با اهداف دودویی و چند هدفه پیشنهاد کرد. ثانیا، پرواز لِوی، جهش و سایر عملگرهای تکاملی را می‌توان برای بهبود عملکرد آن به این الگوریتم ادغام کرد. ثالثا، الگوریتم SCA را میتوان با سایر الگوریتم‌ها در زمینه بهینه‌سازی تصادفی ترکیب کرد تا عملکرد آن را بهبود بخشد. در نهایت، بررسی کاربرد SCA در زمینه‌های مختلف کمک ارزشمندی خواهد بود.

۷. ضمیمه ۱

جدول A۱ و A۲ و A۳ را مشاهده کنید.

Table A.1
Unimodal benchmark functions.

Function	Dim	Range	Shift position	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	20	[−100,100]	[−30, −30,.., −30]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	20	[−10,10]	[−3, −3,..., −3]	0
$f_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	20	[−100,100]	[−30, −30,.., −30]	0
$f_4(x) = \max_i \{\lvert x_i \rvert,\ 1 \le i \le n\}$	20	[−100,100]	[−30, −30,.., −30]	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	20	[−30,30]	[−15, −15,.., −15]	0
$f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	20	[−100,100]	[−750,.., −750]	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1)$	20	[−1.28,1.28]	[−0.25,.., −0.25]	0

Table A.2
Multimodal benchmark functions.

Function	Dim	Range	Shift position	f_{min}
$F_8(x)=\sum_{i=1}^{n}-x_i\sin(\sqrt{\lvert x_i\rvert})$	20	[−500,500]	[−300,..., −300]	-418.9829×5
$F_9(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	20	[−5.12,5.12]	[−2, −2,..., −2]	0
$F_{10}(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$	20	[−32,32]		0
$F_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	20	[−600,600]	[−400,..., −400]	0
$F_{12}(x)=\frac{\pi}{n}\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$	20			
$y_i=1+\frac{x_i+1}{4}$		[−50,50]	[−30, −30,..., −30]	
$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a<x_i<a\\ k(-x_i-a)^m & x_i<-a\end{cases}$	20			0
$F_{13}(x)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_i+1)]+(x_n-1)^2[1+\sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$		[−50,50]	[−100,..., −100]	0

Table A.3
Composite benchmark functions.

Function	Dim	Range	f_{min}
F_{14} *(CF1):*			
$f_1, f_2, f_3, \ldots, f_{10}$ = Sphere Function			
$[б_1, б_2, б_3, \ldots, б_{10}] = [1, 1, 1, .., 1]$	10	[−5,5]	0
$[\lambda_1, \lambda_2, \lambda_3 \ldots, \lambda_{10}] = [5/100, 5/100, 5/100, .., 5/100]$			
F_{15} *(CF2):*			
$f_1, f_2, f_3, \ldots, f_{10}$ = *Griewank′sFunction*			
$[б_1, б_2, б_3, \ldots, б_{10}] = [1, 1, 1, .., 1]$	10	[−5,5]	0
$[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/100, 5/100, 5/100, .., 5/100]$			
F_{16} *(CF3):*			
$f_1, f_2, f_3, \ldots, f_{10}$ = *Griewank′sFunction*			
$[б_1, б_2, б_3, \ldots, б_{10}] = [1, 1, 1, .., 1]$	10	[−5,5]	0
$[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [1, 1, 1, .., 1]$			
f_{17} *(CF4):*			
f_1, f_2 = Ackley′sFunction			
f_3, f_4 = Rastrigin′s Function			
f_5, f_6 = Weierstrass Function	10	[−5,5]	0
f_7, f_8 = Griewank′s Function			
f_9, f_{10} = Sphere Function			
$[б_1, б_2, б_3, \ldots, б_{10}] = [1, 1, 1, .., 1]$			
$[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/32,\ 5/32,\ 1,\ 1,\ 5/0.5,\ 5/0.5,\ 5/100,\ 5/100, 5/100,\ 5/100]$			
f_{18} *(CF5):*			
f_1, f_2 = Rastrigin′s Function			
f_3, f_4 = Weierstrass Function			
f_5, f_6 = Griewank′s Function	10	[−5,5]	0
f_7, f_8 = Ackley′sFunction			
f_9, f_{10} = Sphere Function			
$[б_1, б_2, б_3, \ldots, б_{10}] = [1, 1, 1, .., 1]$			
$[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [1/5,\ 1/5,\ 5/0.5,\ 5/0.5,\ 5/100,\ 5/100,\ 5/32,\ 5/32, 5/100,\ 5/100]$			
f_{19} *(CF6):*			
f_1, f_2 = Rastrigin′s Function			
f_3, f_4 = Weierstrass Function			
f_5, f_6 = Griewank′s Function	10	[−5,5]	0
f_7, f_8 = Ackley′sFunction			
f_9, f_{10} = Sphere Function			
$[б_1, б_2, б_3, \ldots, б_{10}] = [0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.9,\ 1]$			
$[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [0.1*1/5,\ 0.2*1/5,\ 0.3*5/0.5,\ 0.4*5/0.5,\ 0.5*5/100,$			
$0.6*5/100, 0.7*5/32,\ 0.8*5/32, 0.9*5/100, 1*5/100]$			

۸. ارجاعات

[1] A.R. Simpson, G.C. Dandy, L.J. Murphy, Genetic algorithms compared to other techniques for pipe optimization, J. Water Resour. Plan. Manag. 120 (1994) 423–443.

[2] C. James, Introduction to Stochastics Search and Optimization, WileyInterscience, New Jersey, 2003.

[3] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Inf. Sci. 237 (2013) 82–117.

[4] J.A. Parejo, A. Ruiz-Cortés, S. Lozano, P. Fernandez, Metaheuristic optimization frameworks: a survey and benchmarking, Soft Comput. 16 (2012) 527–561.

[5] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P.N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: a survey of the state of the art, Swarm Evol. Comput. 1 (2011) 32–49.

[6] S. Droste, T. Jansen, I. Wegener, Upper and lower bounds for randomized search heuristics in black-box optimization, Theory of Comput. Syst. 39 (2006) 525–544.

[7] H.H. Hoos, T. Stützle, Stochastic Local Search: Foundations & Applications, Elsevier, 2004.

[8] R.S. Parpinelli, H.S. Lopes, New inspirations in swarm intelligence: a survey, Int. J. Bio-Inspired Comput. 3 (2011) 1–16.

[9] C.M. Fonseca, P.J. Fleming, An overview of evolutionary algorithms in multiobjective optimization, Evol. Comput. 3 (1995) 1–16.

[10] A. Biswas, K. Mishra, S. Tiwari, A. Misra, Physics-inspired optimization algorithms: a survey, J. Optim. 2013 (2013).

[11] A. Gogna, A. Tayal, Metaheuristics: review and application, J. Exp. Theor. Artif. Intell. 25 (2013) 503–526.

[12] X.-S. Yang, Z. Cui, R. Xiao, A.H. Gandomi, M. Karamanoglu, Swarm intelligence and bio-inspired computation: theory and applications, Newnes (2013).

[13] S. Saremi, S. Mirjalili, A. Lewis, Biogeography-based optimisation with chaos, Neural Comput. Appl. 25 (2014) 1077–1097.

[14] G.-G. Wang, L. Guo, A.H. Gandomi, G.-S. Hao, H. Wang, Chaotic krill herd algorithm, Inf. Sci. 274 (2014) 17–34.

[15] G.-G. Wang, A. Hossein Gandomi, A. Hossein Alavi, A chaotic particle-swarm krill herd algorithm for global numerical optimization, Kybernetes 42 (2013) 962–978.

[16] G.G. Wang, S. Deb, A.H. Gandomi, Z. Zhang, A.H. Alavi, A novel cuckoo search with chaos theory and elitism scheme, in: Proceedings of 2014 International Conference on Soft Computing and Machine Intelligence (ISCMI), 2014, pp. 64– 69.

[17] G.-G. Wang, S. Deb, A.H. Gandomi, Z. Zhang, A.H. Alavi, Chaotic cuckoo search, Soft Comput. (1726) 1–14.

[18] G. Wang, L. Guo, H. Wang, H. Duan, L. Liu, J. Li, Incorporating mutation scheme into krill herd algorithm for global numerical optimization, Neural Comput. Appl. 24 (2014) 853–871.

[19] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, A bat algorithm with mutation for UCAV path planning, Sci. World J. 2012 (2012).

[20] J.W. Zhang, G.G. Wang, Image matching using a bat algorithm with mutation, Appl. Mech. Mater. 203 (2012) 88–93.

[21] H.-R. Li, Y.-L. Gao, Particle swarm optimization algorithm with exponent decreasing inertia weight and stochastic mutation, in: Proceedings of Second International Conference on Information and Computing Science, 2009 (ICIC'09), 2009, pp. 66–69.

[22] S. Chen, Particle swarm optimization with pbest crossover, in: Proceedings of 2012 IEEE Congress on Evolutionary Computation (CEC), 2012, pp. 1–6.

[23] Q. Zhu, Z. Yang, An ant colony optimization algorithm based on mutation and dynamic pheromone updating, J. Softw. 15 (2004) 185–192.

[24] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer with local search, in: Proceedings of 2005 IEEE Congress on Evolutionary Computation, 2005, pp. 522–528.

[25] K. Premalatha, A. Natarajan, A new approach for data clustering based on PSO with local search, Comput. Inf. Sci. 1 (2008) 139.

[26] N. Noman, H. Iba, Accelerating differential evolution using an adaptive local search, IEEE Trans. Evol. Comput. 12 (2008) 107–125.

[27] J. Levine, F. Ducatelle, Ant colony optimization and local search for bin packing and cutting stock problems, J. Oper. Res. Soc. 55 (2004) 705–716.

[28] C. Blum, A. Roli, Hybrid metaheuristics: an introduction, Hybrid Metaheuristics, Springer, 2008, pp. 1–30.

[29] M. Ehrgott, X. Gandibleux, Hybrid metaheuristics for multi-objective combinatorial optimization, Hybrid metaheuristics, Springer, 2008, pp. 221–259.

[30] G. Wang, L. Guo, A novel hybrid bat algorithm with harmony search for global numerical optimization, J. Appl. Math. 2013 (2013).

[31] G.-G. Wang, A.H. Gandomi, A.H. Alavi, G.-S. Hao, Hybrid krill herd algorithm with differential evolution for global numerical optimization, Neural Comput. Appl. 25 (2014) 297–308.

[32] G. Wang, L. Guo, H. Duan, H. Wang, L. Liu, M. Shao, A hybrid metaheuristic DE/CS algorithm for UCAV three-dimension path planning, Sci. World J. 2012 (2012).

[33] G.-g. Wang, L. Guo, H. Duan, H. Wang, L. Liu, M. Shao, Hybridizing harmony search with biogeography based optimization for global numerical optimization, J. Comput. Theor. Nanosci. 10 (2013) 2312–2322.

[34] H. Duan, W. Zhao, G. Wang, X. Feng, Test-sheet composition using analytic hierarchy process and hybrid metaheuristic algorithm TS/BBO, Math. Probl. Eng. 2012 (2012).

[35] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, B. Wang, A hybrid meta-heuristic DE/CS algorithm for UCAV path planning, J. Inf. Comput. Sci. 5 (2012) 4811– 4818.

[36] X. Shi, Y. Liang, H. Lee, C. Lu, L. Wang, An improved GA and a novel PSO-GAbased hybrid algorithm, Inf. Process. Lett. 93 (2005) 255–261.

[37] N. Holden, A.A. Freitas, A hybrid PSO/ACO algorithm for discovering classification rules in data mining, J. Artif. Evol. Appl. 2008 (2008) 2.

[38] S. Nemati, M.E. Basiri, N. Ghasem-Aghaee, M.H. Aghdam, A novel ACO–GA hybrid algorithm for feature selection in protein function prediction, Expert Syst. Appl. 36 (2009) 12086–12094.

[39] W.-Y. Lin, A GA–DE hybrid evolutionary algorithm for path synthesis of fourbar linkage, Mech. Mach. Theory 45 (2010) 1096–1107.

[40] B. Niu, L. Li, A novel PSO-DE-based hybrid algorithm for global optimization, Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence, Springer, 2008, pp. 156–163.

[41] H. Duan, Y. Yu, X. Zhang, S. Shao, Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm, Simul. Model. Pract. Theory 18 (2010) 1104–1115.

[42] G.-G. Wang, A.H. Gandomi, X.-S. Yang, A.H. Alavi, A new hybrid method based on krill herd and cuckoo search for global optimization tasks, Int. J. BioInspired Comput. (2013).

[43] G.-G. Wang, A.H. Gandomi, A.H. Alavi, An effective krill herd algorithm with migration operator in biogeography-based optimization, Appl. Math. Model. 38 (2014) 2454–2462.

[44] J.H. Holland, J.S. Reitman, Cognitive systems based on adaptive algorithms, ACM SIGART Bull. (63) (1977) 49–49.

[45] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341– 359.

[46] Y. Wang, H.-X. Li, T. Huang, L. Li, Differential evolution based on covariance matrix learning and bimodal distribution parameter setting, Appl. Soft Comput. 18 (2014) 232–247.

[47] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (2011) 55–66.

[48] Y. Wang, Z. Cai, Q. Zhang, Enhancing the search ability of differential evolution through orthogonal crossover, Inf. Sci. 185 (2012) 153–177.

[49] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (2008) 702–713.

[50] I. Rechenberg, Evolutionsstrategien, in: B. Schneider, U. Ranft (Eds.), Simulationsmethoden in der Medizin und Biologie, 8, Springer, Berlin Heidelberg, 1978, pp. 83–114.

[51] M. Dorigo, M. Birattari, Ant colony optimization, Encyclopedia of Machine Learning, Springer, 2010, pp. 36–39.

[52] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.

[53] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Global Optim. 39 (2007) 459–471.

[54] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. 179 (2009) 2232–2248.

[55] A. Kaveh, V. Mahdavi, Colliding Bodies Optimization method for optimum discrete design of truss structures, Comput. Struct. 139 (2014) 43–53.

[56] A. Hatamlou, Black hole: a new heuristic optimization approach for data clustering, Inf. Sci. 222 (2013) 175–184.

[57] A.H. Kashan, League Championship Algorithm (LCA): an algorithm for global optimization inspired by sport championships, Appl. Soft Comput. 16 (2014) 171–200.

[58] A. Sadollah, A. Bahreininejad, H. Eskandar, M. Hamdi, Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems, Appl. Soft Comput. 13 (2013) 2592–2612.

[59] R.V. Rao, V.J. Savsani, D. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Comput.-Aided Des. 43 (2011) 303–315.

[60] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1997) 67–82.

[61] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, Knowl.-Based Syst. 89 (2015) 228–249.

[62] M. Crepinšek, S.-H. Liu, M. Mernik, Exploration and exploitation in evolution- ˇ ary algorithms: a survey, ACM Comput. Surv. 45 (2013) 35.

[63] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (1999) 82–102.

[64] J. Digalakis, K. Margaritis, On benchmarking functions for genetic algorithms, Int. J. Comput. Math. 77 (2001) 481–506.

[65] M. Molga, C. Smutnicki, Test functions for optimization needs, Test Funct. Optim. Needs (2005).

[66] X.-S. Yang, Test problems in optimization, 2010. Available from arXiv:1008. 0549.

[67] X.-S. Yang, Firefly algorithm, stochastic test functions and design optimisation, Int. J. Bio-Inspired Comput. 2 (2010) 78–84.

[68] X.-S. Yang, A new metaheuristic bat-inspired algorithm, Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Springer, 2010, pp. 65–74.

[69] X.-S. Yang, M. Karamanoglu, X. He, Flower pollination algorithm: a novel approach for multiobjective optimization, Eng. Optim. 46 (2014) 1222–1237.

[70] Y. Wang, Z. Cai, Combining multiobjective optimization with differential evolution to solve constrained optimization problems, IEEE Trans. Evol. Comput. 16 (2012) 117–134.

[71] S.H.R. Pasandideh, S.T.A. Niaki, A. Gharaei, Optimization of a multiproduct economic production quantity problem with stochastic constraints using sequential quadratic programming, Knowl.-Based Syst. 84 (2015) 98–107.

[72] S. Jalali, M. Seifbarghy, J. Sadeghi, S. Ahmadi, Optimizing a bi-objective reliable facility location problem with adapted stochastic measures using tunedparameter multi-objective algorithms, Knowl.-Based Syst. (2015) in press.

[73] H. Salimi, Stochastic fractal search: a powerful metaheuristic algorithm, Knowl.-Based Syst. 75 (2015) 1–18.

[74] C.A. Coello Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Comput. Methods Appl. Mech. Eng. 191 (2002) 1245–1287.

[75] M. Drela, XFOIL: An analysis and design system for low Reynolds number airfoils, in: Low Reynolds number aerodynamics, Springer, 1989, pp. 1–12.