ارائه روشی جدید برای بهینه‌سازی سراسری توابع با ابعاد بالا

محمد علی بزرگ زاده*، آرش رحیمی†، سعید شیری قیداری‡

چکیده

در این مقاله، روش جدیدی برای بهینه‌سازی سراسری توابع با متغیرهای پیوسته معرفی شده است. در این روش، جستجو در فضای ورودی از طریق انجام جستجوهای متوالی و متناوب بر روی صفحات حاصل از انتخاب دوبعد از فضای ورودی انجام می‌شود. در هر صفحه جستجو، تعدادی نقطه بر روی شبکه‌ای منظم قرار داده می‌شود و طی رویه‌ای تکراری، هر نقطه با توجه به مقادیر تابع در یکی از نقاط همسایه‌اش تحت تأثیر مجموعه‌ای از قوانین جذب و دفع قرار می‌گیرد. در پایان الگوریتم نیز، با انجام جستجوی محلی با استفاده از روش نلدر-مید نقطه بهینه مشخص می‌گردد. روش ارائه‌شده با بیش از ۲۰ نوع تابع آزمون استاندارد، بصورت موفقیت‌آمیز مورد آزمایش قرار گرفته است. نتایج آزمایشات انجام‌شده نشان می‌دهند که، از مزایای این‌کار نسبت به کارهای مشابه، قابلیت روش ارائه‌شده در یافتن نقاط بهینه سراسری در توابع با ابعاد بالا (بیش از ۳۰ بعد) است.

کلمات کلیدی

بهینه‌سازی سراسری ، جستجوی تصادفی، متغیر پیوسته، بهینه‌سازی بدون قید.

A Novel Approach for Global Optimization in High Dimensions

Mohammad Ali Bozorgzadeh, Arash Rahimi, Saeed Shiry Ghidary
Department of Computer Engineering and Information Technology
Amirkabir University of Technology

Abstract

In this paper, a new method is presented for global optimization of functions defined on real-valued variables. In the presented method, the search in the input space is divided into several successive searches on 2-dimensional planes which are defined by selecting pairs of variables from the input space. The optimization process within each plane divides the plane into a grid of points and then iteratively applies a set of absorption and emission rules in order to find a near-optimum point in the plane. The proposed method was successfully tested with more than 20 standard test functions. The results show that the proposed method can be effectively used for optimization of functions with high dimensions.

Keywords

Global optimization, Stochastic Search, Real-Valued Variable, Unconstrained Optimization.

* دانشجوی کارشناسی ارشد مهندسی کامپیوتر – گرایش هوش ماشین و رباتیک، دانشگاه صنعتی امیرکبیر، bozorgzadeh@ce.aut.ac.ir
† دانشجوی کارشناسی ارشد مهندسی کامپیوتر – گرایش هوش ماشین و رباتیک، دانشگاه صنعتی امیرکبیر، rahimi@ce.aut.ac.ir
‡ استادیار و عضو هیأت علمی، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر، shiry@ce.aut.ac.ir

۱- مقدمه

مسائل زیادی در علوم ریاضی، مهندسی، و کاربردی، از جمله مسائل مختلف مطرح در زمینه علوم و مهندسی کامپیوتر، منجر به بهینه‌سازی سراسری تابعی خاص می‌گردند. مسأله بهینه‌سازی سراسری عبارت است از یافتن نقطه مینیمم سراسری $\mathbf{x}^*$ برای تابع (معلوم یا مجهول) چندبعدی و با دامنه تعریف محدود $f(\mathbf{x})$:

$$\begin{aligned} &\min(f(\mathbf{x})) \\ &subject\ to:\quad a_1 < x_1 < b_1 \\ &\qquad\qquad a_2 < x_2 < b_2 \\ &\qquad\qquad \vdots \\ &\qquad\qquad a_n < x_n < b_n \end{aligned} \tag{۱}$$

برای حل این مسأله، روش‌های متعددی از سالها پیش تاکنون مطرح شده است که می‌توان آنها را در دو دسته روشهای قطعی و تصادفی قرارداد. عمده روشهای قطعی محدودیتهایی برروی تابع دارند. برای مثال، در یکی از این روش‌ها، ورودی می‌بایست یک تابع پیوسته لیپشیتز، با ثابت لیپشیتز معلوم یا قابل تخمین باشد. در روش دیگری، تابع می‌بایست مقعر و تعریف‌شده برروی مجموعه‌های مقعر باشد. حداقل محدودیتی که در این روش‌ها دیده می‌شود دیفرانسیل‌پذیر بودن تابع است. از جمله این روش‌ها می‌توان به روش شاخه و حد اشاره نمود [۸].

روش‌های شاخه و حد یا بازه‌ای ابتدا در سال ۱۹۶۶ توسط مور معرفی شدند. از آن زمان تاکنون بهبودهایی بر روی آن ایجاد شده و از آن جمله تکنیک‌هایی برای انجام موازی این نوع جستجو است [۴]. در حیطه روشهای قطعی، غالباً تضمین‌هایی در مورد جواب مسأله وجود دارد. برای مثال، در روشهای شاخه و حد، می‌توان تضمینی در مورد حد بالای فاصله تا مقدار مینیمم سراسری ارائه نمود و با داشتن زمان کافی، می‌توان این بازه را تا حد دلخواه کوچک نموده و نواحی‌ای را یافت که می‌توانند مقادیری در این فاصله تولید کنند. همچنین می‌توان به تکنیک‌هایی اشاره نمود که به یافتن کلیه مینیمم‌های محلی می‌پردازند تا از بین آنها مینیمم سراسری یافت شود [۱۶,۱۳].

با وجود تضمین‌های روشهای قطعی، به‌دلیل زمان‌گیر بودن و محدودیت‌هایی که برروی تابع دارند، همچنان روشهای تصادفی مورد توجه است. از الگوریتم‌های کلاسیک تصادفی، می‌توان به جستجوی تصادفی صرف و روش چندآغازی [۲] اشاره نمود که در آن یک بهینه‌سازی محلی برروی کلیه نقاط آغازین انجام می‌گیرد. در ادامه، می‌توان به روش تابکاری شبیه‌سازی‌شده[۱] (SA) و دیگر رویه‌های مبتنی بر تئوری‌های نمونه‌برداری متروپولیس و هستینگز اشاره کرد [۱۲,۶].

از تکنیک‌هایی که اخیراً مورد توجه قرار گرفته می‌توان به جستجوی تابو (TS) [۹]، راهبردهای تکاملی [۱۵,۷] و الگوریتم ژنتیک پیوسته [۳] اشاره نمود. غیر از این موارد، روش‌های توپوگرافیکی [۱۸] و روش‌های مبتنی بر تئوری گاویانو [۵] هستند که در هر مرحله یک جهت تصادفی برای حرکت انتخاب می‌کنند.

در زمینه مقایسه الگوریتم‌های فوق، مشکلاتی وجود دارد. از آن جمله اینکه همگی آنها با توابع یکسان و هم‌سطحی مورد آزمایش قرار نگرفته‌اند. برخی از روش‌ها، احتیاج به تنظیمات تعداد زیادی پارامتر دارند و اظهار نظر در مورد بهترین روش در بسیاری موارد به ویژگیهای تابع ورودی وابسته است. مثلاً برای توابع دارای نویز روشهای خاصی توسعه داده شده‌اند [۱۰,۱۷,۱۴].

در نگاه اول، برخی از روش‌ها در مدت زمان اندک نتایج بسیار نزدیک به مینیمم سراسری ارائه می‌کنند، اما برای بسیاری از آنها، در مورد توابع با ابعاد بالا (خصوصاً بالاتر از ۱۲ بعد) گزارشی ارائه نشده است [۱۱].

در این مقاله، به ارائه روشی جدید جهت بهینه‌سازی سراسری توابع پیوسته پرداخته شده است که خاصه برای توابع با ابعاد بالا می‌تواند کارآ باشد. با وجودی که در هیچ یک از نتایج ارائه‌شده در مقاله‌های مرجع ما به توابع بالاتر از ۳۰ بعد پرداخته نشده است، در این مقاله، نتایج بر روی توابع آزمون استاندارد با تعداد ابعاد ۷۰ نیز ارائه شده است.

روند این مقاله بدین شرح است که، ابتدا در بخش ۲، الگوریتمی جهت بهینه‌سازی سراسری تابعی دوبعدی تشریح می‌شود و سپس تعمیم آن به ابعاد بالاتر در بخش ۳ ارائه می‌گردد. در پایان، در بخش ۴، نتایج آزمایشات انجام‌شده و در بخش ۵، نتیجه‌گیری نهایی ارائه خواهد شد.

۲- الگوریتم پایه برای حالت دوبعدی

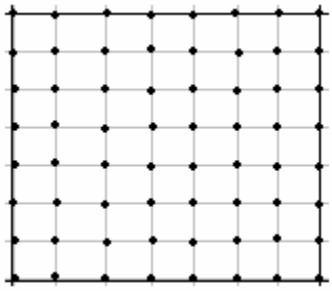
در این بخش، الگوریتمی جهت بهینه‌سازی سراسری توابع دوبعدی پیشنهاد خواهد شد. جهت انجام بهینه‌سازی فرض می‌کنیم که تابعی دوبعدی بر روی صفحه x-y داریم. جهت انجام بهینه‌سازی به‌صورت زیر عمل می‌کنیم:

هر یک از محورهای x و y در محدوده‌ای که عمل جستجو در آن انجام می‌گیرد، به‌طور منظم به N نقطه تقسیم می‌گردد که N پارامتر دقت می‌باشد. حال با امتداد دادن هر یک از نقاط تقسیم بر روی محور x با خطوطی موازی محور y ، همچنین امتداد دادن هر یک از نقاط تقسیم بر روی محور y با خطوطی موازی محور x ، شبکه‌ای منظم حاصل می‌گردد که نقاط تقاطع این خطوط محل قرارگیری نقاط اولیه است (شکل ۱).

در هر تکرار الگوریتم، کلیه نقاط بررسی شده و مکان آنها طبق قوانینی مشخص دستخوش تغییر می‌شوند. تغییر انجام‌شده بر روی مکان هر نقطه تنها به نزدیکترین همسایه آن نقطه وابسته است. بدین ترتیب که چنانچه مقدار تابع در نزدیکترین نقطه به آن از مقدار تابع در نقطه کوچکتر باشد، نقطه مورد بررسی در امتداد خط واصلش به

نزدیکترین نقطه به میزان مشخصی به آن جذب می‌گردد، و در غیر اینصورت، در همان امتداد دفع می‌گردد. البته از خروج نقاط از مرزهای تعریف‌شده برای هرکدام از محورها، جلوگیری می‌شود.

توجیه این قوانین ساده است. انتظار داریم نقاط مینیمم در نزدیکی نقاطی باشد که در آن مقدار تابع کمتر است و نیز از نقاطی که مقدار تابع در آن بیشتر است، دور باشد. چنانچه در طول گامهای جذب و دفع خطا کرده باشیم، جذب و دفع‌های بعدی، خطا را اصلاح خواهند کرد. در ادامه و در بخش ۲-۱ به یک مشکل که ممکن است با آن مواجه شویم اشاره کرده و راه حلی برای آن ارائه می‌کنیم.



شکل (۱): محل قرارگیری نقاط اولیه جستجو

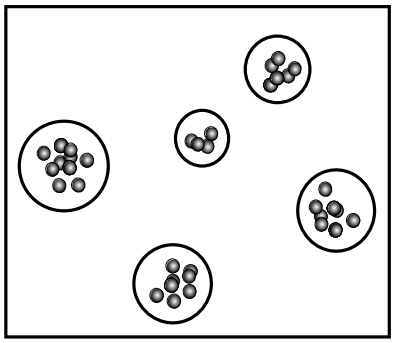
بنابراین، در هر تکرار، ابتدا مقدار تابع در کلیه نقاط محاسبه می‌شود، سپس کلیه نقاط موردبررسی قرار گرفته و نزدیکترین همسایه به آنها و مکان جدیدشان بعد از جذب یا دفع مشخص می‌گردد. این روند تا جایی ادامه می‌یابد که پس از چندین تکرار متوالی، مقدار مینیمم یافت‌شده از میان کلیه نقاط ارزیابی‌شده، بهبود پیدا نکند.

بنابر موارد ذکرشده، پیچیدگی زمانی الگوریتم در هر تکرار از $O(N^2 F_{eval} + N^4)$ خواهد بود، که در آن F_{eval} پیچیدگی زمانی یک‌بار فراخوانی تابع است.

۲-۱- کلونی‌های بسته و مقابله با آن

در جریان تکرارها در یک سطح، ممکن است با پدیده ایجاد کلونی مواجه شویم. کلونی زیرمجموعه‌ای از نقاط جستجو است که به ازای هر نقطه داخل مجموعه، نزدیکترین همسایه آن نقطه نیز در مجموعه موجود باشد. حال فرض می‌کنیم که در جریان جستجو داخل یک سطح، با حالتی مواجه شویم که در آن:

- نقاط جستجو به چند کلونی مجزا تفکیک شده باشند (شکل ۲).
- جذب و دفع‌های داخل هر کلونی موجب خروج یک نقطه از کلونی یا ورود نقطه جدیدی به کلونی نشود و تنها باعث جابجایی اندک نقاط در همان کلونی گردد.



شکل (۲): تشکیل کلونی‌های بسته

واضح است که از وقوع چنین حالتی بایستی جلوگیری شود؛ زیرا به محض قرار گرفتن در آن، امکان جستجو در فضای حالت محدود به فضای اطراف هر کلونی شده و بدین ترتیب، امکان گرفتار شدن در نقطه مینیمم محلی بیشتر می‌شود. برای مثال، یک کلونی می‌تواند شامل یک مینیمم محلی و محدوده اطراف آن باشد. در این حالت، جذب و دفع‌ها باعث می‌شود نقاط کلونی در اطراف مینیمم محلی تجمع یابند و کلونی دائماً کوچکتر شود و بدین ترتیب، نقاط نمی‌توانند از فضای داخل کلونی آزاد شوند و در مسیر دیگری به حرکت ادامه دهند.

یک راه حل برای جلوگیری از وقوع چنین حالتی، جلوگیری از ایجاد کلونی است. برای این منظور، از مجموعه‌های مجزا[۲] [۱] استفاده می‌نماییم. روند کار بدین ترتیب است که ابتدا به‌ازای هر نقطه، یک مجموعه در نظر گرفته می‌شود. در طی اجرای الگوریتم، هر نقطه بدنبال یافتن نزدیکترین همسایه‌ای است که در مجموعه‌اش قرار نگرفته باشد. سپس مجموعه آن نقطه با مجموعه همسایه یافت‌شده ادغام شده و بروزسازی انجام می‌گیرد. می‌توان ثابت نمود که با انجام این کار عدم وجود کلونی تضمین می‌شود.

اثبات: فرض می‌کنیم k نقطه با یکدیگر تشکیل کلونی داده باشند. با انتخاب یک نقطه از این k نقطه و اضافه کردن مجموعه نزدیکترین همسایه آن، تعداد اعضای مجموعه شامل نقاط کلونی، لااقل، به اندازه یک واحد افزایش می‌یابد. بنابراین، با اضافه کردن مجموعه نقطه k-1ام این مجموعه که از ابتدا یک عضو داشته k عضوی خواهد بود. از آنجایی که این مجموعه تنها می‌تواند اعضای همان کلونی را در بر داشته باشد، حتما نقطه kام نیز داخل همین مجموعه قرار می‌گیرد. بنابراین طبق الگوریتم ذکرشده، این نقطه می‌بایستی نزدیکترین همسایه خود را از خارج از مجموعه کلونی انتخاب نماید. بدین ترتیب، کلونی شکسته می‌شود و ارتباط بین اعضای آن و اعضای بیرون از کلونی ایجاد می‌شود.

بایستی به این نکته توجه نمود که ترتیب انتخاب نقاط در یافتن نزدیکترین همسایه‌ها مؤثر است. به عبارت دیگر، هر چه یک نقطه زودتر انتخاب شود، شانس انتخاب همسایه نزدیکتری را دارد. بنابراین، بهتر است ترتیب نقاط از پیش ثابت نباشد. برای این منظور، قبل از یافتن نزدیکترین همسایه نقاط، ترتیب بررسی و بروزسازی نقاط به‌طور تصادفی بهم ریخته می‌شود.

همانطور که می‌دانیم، همواره یک کلونی سراسری شامل کلیه نقاط وجود دارد. بنابراین، می‌توان استدلال نمود که طبق الگوریتم ذکرشده، همواره نقطه آخر در ترتیب بررسی، هیچ نزدیکترین همسایه مجازی نخواهد داشت. جهت رفع این مسأله، می‌توان برای این نقطه راهکارهای دیگری را درنظر گرفت، اما با توجه به متغیر بودن ترتیب بررسی نقاط در تکرارهای متوالی و ثابت نبودن نقطه آخر در آن، این موضوع مشکلی در رویه اجرای الگوریتم ایجاد نمی‌کند.

لازم به‌ذکر است که، الگوریتم فوق را می‌توان به‌سادگی و با تغییرات اندکی جهت توابع با فضای ورودی یک‌بعدی نیز مورد استفاده قرار داد.

۲-۲- پارامترها

به منظور کنترل نحوه عملکرد الگوریتم، رویه جذب و دفع، و شرط پایان جستجو پارامترهای زیر درنظر گرفته شده است:

دقت (N): تعداد نقاط تقسیم بر روی هر محور.

حداکثر تعداد تکرارهای ایستا (S): برای شرط پایان جستجو به‌کار می‌رود.

وجود یا عدم وجود دفع: مشخص می‌کند که آیا فقط از جذب استفاده شود یا هم از جذب هم از دفع.

تطبیقی بودن: چنانچه الگوریتم غیرتطبیقی باشد به معنای آن است که میزان جذب و دفع نقاط کسر ثابتی از فاصله بین دو نقطه و مطابق با رابطه ۲، خواهد بود:

$$\begin{cases} x_{point} = x_{point} + alpha * (x_{point} - x_{neighbor}) \\ \quad \mathrm{sgn}(f(neighbor) - f(point)) \\ y_{point} = y_{point} + alpha * (y_{point} - y_{neighbor}) \\ \quad \mathrm{sgn}(f(neighbor) - f(point)) \end{cases} \qquad (۲)$$

در غیر اینصورت، مقدار جذب و دفع متناسب با اختلاف مقدار تابع در نقطه با نزدیکترین همسایه‌اش و براساس رابطه ۳ خواهد بود:

$$\begin{cases} x_{point} = x_{point} + alpha * \dfrac{f(neighbor) - f(point)}{|f(neighbor)| + |f(point)| + \varepsilon} \\ \quad (x_{point} - x_{neighbor}) \\ y_{point} = y_{point} + alpha * \dfrac{f(neighbor) - f(point)}{|f(neighbor)| + |f(point)| + \varepsilon} \\ \quad (y_{point} - y_{neighbor}) \end{cases} \qquad (۳)$$

در این روابط، *f(point)* مقدار تابع در نقطه و *f(neighbor)* مقدار تابع در همسایگی نقطه می‌باشد. کاربرد ε برای جلوگیری از صفر شدن مخرج کسر می‌باشد.

ضریب جذب و دفع ($alpha$): میزان جذب و دفع یک نقطه به سمت نقطه همسایه.

پارامتر به‌روزرسانی همسایگان: تعیین می‌کند که در هر چند تکرار یکبار، برای هر نقطه، نزدیکترین همسایه‌اش، بروزرسانی شود (مقدار پیش‌فرض: ۱). این پارامتر جهت افزایش سرعت الگوریتم و برقراری توازن میان سرعت و دقت عملکرد آن می‌تواند مورد استفاده قرار گیرد.

۳- تعمیم الگوریتم به بیش از دو بعد

برای تعمیم الگوریتم به ابعاد بالاتر، یک ایده ساده اما ناکارآمد قرار دادن نقاط در فواصل یکسان از یکدیگر در فضای nبعدی و تعمیم مستقیم الگوریتم ذکرشده در بخش قبلی جهت این حالت است. در اینصورت، با افزایش ابعاد، تعداد نقاط و در نتیجه آن، پیچیدگی محاسباتی (در این حالت از $O(N^n F_{eval} + N^{2n})$) بصورت نمایی افزایش می‌یابد، و واضح است که، در ابعاد بالا الگوریتم از لحاظ حجم محاسبات و حافظه مصرفی غیرقابل انجام است. جهت حل این مشکل، می‌توان از ایده بهینه‌سازی تناوبی[۳] [۱۹]، یعنی بهینه‌سازی با استفاده از جستجوهای متوالی در زیرفضاهای با ابعاد کمتر از ورودی، بهره برد. به این ترتیب که در هر مرحله، چند بعد انتخاب شده و تشکیل ابرصفحه جستجو را خواهند داد، و مقادیر مربوط به ابعاد دیگر همگی نقاط ثابت و مطابق با بهترین مکان‌های یافت‌شده در جستجوهای ابرصفحه‌ای قبلی مقداردهی می‌شود. حال جستجو بدنبال نقطه بهینه در ابرصفحه موردنظر انجام‌شده و مقدار بهینه سراسری یافت‌شده به‌روز می‌گردد، و این عمل جهت سایر زیرفضاهای ورودی به‌طور متناوب ادامه خواهد یافت.

اما مسأله مهم جهت استفاده از ایده فوق چگونگی تقسیم فضای ورودی به زیرفضاهای مناسب است. در روش پیشنهادی، جهت جلوگیری از افزایش پیچیدگی محاسباتی رویه بهینه‌سازی در هر ابرصفحه و همچنین ایجاد امکان گسترش‌های آتی الگوریتم پیشنهادی، از زیرفضاها و صفحات دوبعدی استفاده شده است. الگوریتم انجام این‌کار به‌شرح زیر است:

۰. شروع
۱. مقداردهی اولیه تصادفی به بردار ابعاد
۲. انتخاب صفحه جستجو
۳. جستجو در داخل صفحه تا رسیدن به وضعیت ایستا
۴. بروزرسانی بهترین مینیمم سراسری یافت‌شده تاکنون
۵. مقداردهی ابعاد صفحه جاری در بردار ابعاد با مینیمم یافت‌شده داخل صفحه جستجوی جاری
۶. خاتمه الگوریتم در صورت ارضای شرط توقف و در غیر اینصورت تکرار کلیه مراحل از مرحله ۲
۷. پایان

با استفاده از توالی جستجوهای دوبعدی، وقتی بر روی یک نقطه مینیمم قرار گرفته باشیم، به یک حالت ایستا می‌رسیم. در اینجا، منظورمان از حالت ایستا، حالتی است که هیچ صفحه جستجویی پیدا نشود که پس از جستجو در آن، مقادیر بهینه‌سراسری یافت‌شده به‌روز شود.

در پایان جهت استقرار بهینه سراسری یافت‌شده در نزدیکترین دره یا بهینه محلی از روش جستجوی محلی نلدر-مید [۲۰] که یکی از معروفترین و کارآمدترین و در عین حال، ساده‌ترین روشهای جستجوی محلی است استفاده می‌گردد.

۳-۱- انتخاب صفحه جستجو

جهت انتخاب صفحه جستجو در هر مرحله اجرای الگوریتم، می‌بایستی دو بعد از ابعاد ورودی انتخاب شده و صفحه جستجو بر روی این دو بعد ساخته شود. انتخاب ابعاد می‌تواند به روشهای مختلف از جمله روشهای تصادفی، احتمالی، و روشهای حریصانه قطعی انجام شود. راه‌حل انتخاب‌شده در این مقاله، از ترکیبی از روشهای قطعی و تصادفی استفاده می‌کند.

روش پیشنهادی، در مرحله اول، از بین تمامی ابعادی که تابحال کمترین تعداد دفعات تشکیل صفحه را داشته‌اند، یکی را به صورت تصادفی انتخاب می‌کند. سپس، جهت انتخاب بعد دوم، از بین کلیه ابعاد باقیمانده مجدداً بعدی را که کمترین تعداد دفعات تشکیل صفحه را داشته است، به صورت تصادفی، جهت تشکیل صفحه انتخاب می‌کند.

یکی از گسترش‌های قابل اعمال برروی الگوریتم پیشنهادی، تخمین وابستگی تابعی دو بعد انتخاب‌شده جهت ایجاد صفحه به یکدیگر و جلوگیری از ایجاد صفحه بر روی این دوبعد در صورت عدم وابستگی آنها به یکدیگر است. به عبارت دیگر، در صورت استقلال تابعی دو متغیر از یکدیگر، بهینه‌سازی برروی صفحه حاصل از آنها معادل انجام دو بهینه‌سازی مستقل یک-بعدی برروی دو متغیر انتخاب‌شده است. به‌عنوان مثال، در تابع $f(x,y)=x+y$ ، متغیرهای x و y از یکدیگر استقلال تابعی داشته در صورتی‌که، در $f(x,y)=xy$ این دو متغیر به یکدیگر وابسته خواهند بود.

۳-۲- جستجو در داخل صفحه جاری

همانطور که می‌دانیم، هر نقطه جستجو یک بردار nبعدی است. هنگامی که داخل یک صفحه جستجو می‌کنیم، تنها دوبعد از این n بعد طبق قواعد جذب و دفع توضیح داده شده، بروز می‌شوند و ابعاد دیگر کلیه نقاط مقادیر ثابتی دارند که از بردار ابعاد خوانده می‌شوند. پس از تکمیل جستجوی صفحه‌ای، ابعاد صفحه جستجو در بردار ابعاد مطابق با بهترین مینیمم یافت‌شده در آن صفحه بروزرسانی می‌شود. به این ترتیب جستجوی nبعدی مبدل به تکرار جستجوهای دوبعدی خواهد شد؛ که مقادیر بهینه یافت‌شده در جستجوهای صفحات قبل، در صفحات بعدی استفاده خواهند شد.

۳-۳- شرط خاتمه

الگوریتم در صورتی خاتمه می‌یابد که پس از انجام c بار انتخاب صفحه و انجام عمل جستجو، مینیمم سراسری جدیدی یافت نشود. مقدار c مناسب در حالت کلی، به نوع تابع و تعداد ابعاد ورودی بستگی دارد. در هر صورت، واضح است که انتخاب مقادیر کوچک جهت c باعث افزایش سرعت الگوریتم شده، در حالیکه انتخاب مقادیر بزرگتر جهت c دقت مینیمم یافت‌شده را افزایش خواهد داد.

۴- نتایج تجربی

جهت بررسی کارآیی الگوریتم پیشنهادی، پیاده‌سازی آن با استفاده از زبان #C انجام شد[۴]. همچنین، جهت سنجش کارآیی آن، تعدادی از توابع کلاسیک مورد استفاده جهت ارزیابی مسائل بهینه‌سازی سراسری [۱ و ۱۸] پیاده‌سازی شده و مورد استفاده قرار گرفت. پارامترهای مورد استفاده در نسخه تحت بررسی الگوریتم در جدول ۱ ذکر شده است. کلیه آزمایش‌ها برروی یک کامپیوتر کیفی با پردازنده Pentium-M و سرعت 1.6GHz انجام شده است.

نسخه‌های ۱۰، ۲۰، ۳۰ ، ۴۰، ۵۰، ۶۰، و ۷۰ بعدی کلیه توابع دارای ابعاد متغیر و همچنین تمامی توابع با ابعاد ثابت (مجموعاً ۷۳ تابع) به عنوان ورودی به الگوریتم پیاده‌سازی‌شده داده شدند. جهت هر تابع، الگوریتم موردنظر ۳۰ بار اجرا شده و بهترین، بدترین، میانگین، و واریانس اختلاف با بهینه سراسری اصلی تابع، تعداد دفعات محاسبه تابع، زمان اجرای الگوریتم، و زمان یافتن نقطه بهینه یافت‌شده ثبت شد[۵]. با توجه به حجم بالای نتایج ثبت‌شده، تنها گزیده‌ای از آنها در جدول ۲ ثبت شده است. نتایج بررسی نشان می‌دهد که الگوریتم ارائه‌شده توانایی بسیار بالایی در یافتن نقطه مینیمم در بسیاری از حالات و در ابعاد بالا از خود ارائه داده است. همچنین، می‌توان متوجه این نکته قابل توجه شد که با افزایش تعداد ابعاد، زمان اجرای الگوریتم پیشنهادی افزایشی خطی داشته است، در حالیکه، با کمی دقت در نتایج ارائه‌شده در اکثر روش‌های پیشنهادی مانند [۱۲] می‌توان دریافت که افزایش ابعاد تأثیری نمایی بر افزایش زمان اجرا دارند.

۵- نتیجه‌گیری

در این مقاله به ارائه الگوریتمی جدید جهت حل مسائل بهینه‌سازی سراسری پرداخته شد. همانطور که ملاحظه شد، الگوریتم ارائه‌شده برپایه روشی جهت بهینه‌سازی همزمان مقادیر دوبعد تابع طراحی شده است. نتایج بررسی‌های انجام‌شده توانایی الگوریتم ذکرشده جهت یافتن نقطه مینیمم سراسری به‌خصوص در ابعاد بالا را نشان می‌دهد.

با وجود نتایج قابل قبول به‌دست آمده، کماکان راهکارهایی جهت بهبود الگوریتم وجود دارد که از جمله آنها می‌توان به ترکیب مناسب روش پیشنهادی با روشهای تخمین وابستگی تابعی ابعاد به یکدیگر جهت کاهش تعداد فراخوانی‌های تابع اشاره کرد. همچنین، با توجه به ساختار الگوریتم ارائه‌شده، بر خلاف بسیاری از الگوریتم‌های موجود در این حوزه، بسادگی می‌توان نسخه‌ای موازی از آن در کاربردهای بزرگ و دارای ابعاد بالا ارائه نمود که از جمله مزایای دیگر آن و یکی از موارد کار بر روی روش پیشنهادی در آینده محسوب می‌شود.

جدول (۱): مقادیر پارامترهای مورداستفاده جهت آزمایشات

پارامتر	مقدار
R	۱۰
S	۱۰۰
وجود دفع	بلی
تطبیقی	خیر
alpha	از ۰،۸ تا ۰،۰۵ با نرخ کاهش ۰،۹ در هر ۱۰ تکرار
شرط پایان	۱۰ تغییر صفحه بدون تغییر بهترین جواب

جدول (۲): نتایج آزمایشات انجام‌شده

نام تابع	کمترین اختلاف	بیشترین اختلاف	میانگین اختلاف	واریانس اختلاف	میانگین فراخوانی تابع	میانگین زمان اجرا	میانگین زمان یافتن نقطه بهینه
Sphere(d = 30)	3.53E-10	2.97E-07	5.19593E-06	1.26038E-12	398112	6.918548384	3.608388608
Sphere(d = 40)	2.09E-08	7.72E-07	7.18871E-06	1.56899E-12	456094	8.524858144	4.962736064
Sphere(d = 50)	2.35E-08	3.20E-06	8.91069E-06	1.68845E-12	522142	10.7138057	6.809792
Sphere(d = 60)	5.08E-07	4.19E-06	1.0816E-05	2.64739E-12	550930	12.50157638	8.220019808
Sphere(d = 70)	2.46E-07	6.49E-06	1.241E-05	2.49563E-12	625360	16.93154637	11.74989552
Schwefel1(d = 10)	2.10E-08	6.70E-07	1.88666E-06	3.07575E-13	361492	7.074973312	2.387633248
Schwefel1(d = 20)	2.36E-07	1.76E-06	3.8128E-06	5.8612E-13	456830	10.03923571	4.617038976
Schwefel1(d = 30)	2.99E-07	2.66E-06	5.87541E-06	1.1675E-12	545902	11.9806273	6.56544064
Schwefel1(d = 40)	3.77E-07	3.77E-06	8.28079E-06	1.61761E-12	669906	16.43983933	10.26936662
Schwefel1(d = 50)	1.02E-06	4.40E-06	3.89342E-05	3.82028E-08	748300	21.63010256	14.3356136
Schwefel1(d = 60)	1.56E-06	5.19E-06	0.003458242	0.000592929	818026	24.91823066	17.19592653
Schwefel1(d = 70)	2.11E-06	9.600000847	0.00085516	3.37574E-05	893944	28.19253888	20.20024653
Schwefel2(d = 10)	8.28E-90	2.78E-75	1.52545E-06	1.93106E-13	295324	5.213096064	1.763936416
Schwefel2(d = 20)	2.27E-17	1.07E-08	3.30743E-06	6.08855E-13	354448	7.02810592	3.033562048
Schwefel2(d = 30)	2.69E-10	3.87E-07	4.98896E-06	6.53541E-13	429844	10.15199786	5.305428832
Schwefel2(d = 40)	5.73E-12	4.92E-08	6.73903E-06	1.51022E-12	481568	14.00273494	7.972263552
Schwefel2(d = 50)	8.36E-11	1.61E-08	9.16033E-06	1.60312E-12	557538	23.63939178	14.58817677
Schwefel2(d = 60)	5.29E-11	5.25E-08	0.00001071	2.155E-12	633012	18.46895706	12.29988637
Schwefel2(d = 70)	1.12E-09	1.81E-07	1.26164E-05	2.53654E-12	691604	27.58867056	19.11468557
Schwefel4(d = 10)	0.000127276	0.000127276	0.000128809	2.68946E-13	385048	5.758680576	2.329750016
Schwefel4(d = 20)	0.000254551	0.000254552	0.012861962	0.00726848	514696	8.931242496	4.859187168
Schwefel4(d = 30)	0.000381827	0.009015306	1.450142839	19.74066232	610928	12.18371933	7.467938368
Schwefel4(d = 40)	0.000509103	0.534868996	5.602628519	128.756422	637700	14.38889021	9.025778432
Schwefel4(d = 50)	0.000636379	0.393598132	14.21570403	579.9312381	750640	19.03036432	13.02552979
Schwefel4(d = 60)	0.000763849	0.567679878	12.2785133	339.8017513	814104	23.21438064	16.3084504
Schwefel4(d = 70)	0.000892965	118.6794584	30.76427989	1972.26372	896662	28.46713373	20.88362918
Step(d = 10)	0	0	0	0	161342	2.428492	0.556600352
Step(d = 20)	0	0	0	0	196270	3.469388736	1.285248096
Step(d = 30)	0	0	0	0	229374	4.728999968	2.228804864
Step(d = 40)	0	0	0	0	262702	6.23496544	3.3998888
Step(d = 50)	0	0	0	0	296194	7.99749984	4.790087808
Step(d = 60)	0	0	0	0	329082	10.00919251	6.454881664
Step(d = 70)	0	0	0	0	361446	12.29708234	8.356816512
Ackley(d = 10)	6.66E-15	6.25E-13	3.2256E-06	9.68409E-13	685546	9.671707232	6.233763712
Ackley(d = 20)	1.69E-07	5.29E-06	0.000007438	1.74819E-12	1101278	17.75773437	13.9484569
Ackley(d = 30)	1.09E-06	5.77E-06	1.17034E-05	2.83875E-12	1483884	27.15484675	22.95941402
Ackley(d = 40)	1.87E-06	9.98E-06	0.000016108	5.69177E-12	1843658	37.87886714	33.18071152
Ackley(d = 50)	2.92E-06	8.96E-06	0.000020586	6.07837E-12	2253190	43.81440202	39.38763664
Ackley(d = 60)	8.00E-06	1.63E-05	0.000026022	1.66887E-11	2531432	59.54261808	54.29387075
Ackley(d = 70)	8.25E-06	2.18E-05	0.000031122	2.10977E-11	2818196	73.35728259	67.60000403
Griewank(d = 10)	0.009857285	0.068799791	0.035562407	0.000229892	510726	7.552259616	4.008964608
Griewank(d = 20)	6.55E-15	0.159502608	0.08470942	0.002804822	436254	7.31151344	3.2246368
Griewank(d = 30)	1.83E-11	0.198522162	0.14772973	0.012893086	467052	8.938452864	4.419755296
Griewank(d = 40)	1.10E-10	0.399386947	0.211378192	0.017300307	477716	10.32765043	5.349492192
Griewank(d = 50)	2.85E-07	0.279946852	0.245814096	0.037614519	547082	13.23643306	7.832662816
Griewank(d = 60)	1.92E-07	0.289511006	0.27657568	0.05563026	543250	14.62963638	8.889582592
Griewank(d = 70)	6.62E-08	0.497873675	0.280295936	0.056379389	604096	17.95842294	11.51035107
Levy2(d = 10)	5.76E-32	1.48E-27	1.45764E-06	2.9393E-13	261440	4.842162688	1.607110912
Levy2(d = 20)	6.91E-18	1.82E-07	3.2686E-06	5.0398E-13	328806	8.255070208	3.824899936
Levy2(d = 30)	6.14E-10	2.49E-07	5.0088E-06	1.18517E-12	398454	12.82163661	7.106819104
Levy2(d = 40)	8.78E-11	1.17E-06	7.2448E-06	1.8922E-12	424246	16.8642496	9.90223872
Levy2(d = 50)	6.17E-09	1.69E-06	9.3128E-06	2.14178E-12	492876	23.21638352	14.9230583
Levy2(d = 60)	7.21E-09	7.89E-07	1.07476E-05	2.19694E-12	571400	31.19064995	21.66735613
Levy2(d = 70)	6.02E-09	9.01E-07	1.35518E-05	3.35953E-12	606904	39.1947593	28.05574218
Kowalik	1.40E-08	0.00128655	0.000294519	1.83862E-07	2332676	38.23898496	35.6879167
Hump	4.65E-08	4.65E-08	5.04444E-07	8.52581E-14	184032	2.549866528	0.165838464
Branin	0.00000036	0.00000135	7.6854E-07	1.0003E-13	178872	2.416875296	0.153821184
Goldstein	2.35E-09	0.000000979	3.56837E-07	1.05565E-13	199726	2.735533504	0.171446528
Hartmann3D	0.0000025	0.00000355	3.0142E-06	9.6033E-14	203816	2.937624096	0.764899872
Hartmann6D	0.00000217	0.119209862	0.045301109	0.003416224	249410	3.763211232	1.528798304

مراجع

[1] Cormen, T., Rivest, R., Leiserson, C., Stein, C., *Introduction To Algorithms*, MIT Press, Cambridge, 2001.

[2] Huyer, W., Neumaier, A., *Global Optimization by Multilevel Coordianate Search*, Kluwer Academic Publishers, 1998.

[3] Wood, D. W., *A Discource Concerning Certain Stochastic Optimization Algoritms and their Application to the Imaging of Cataclysmic Stars*, MS Thesis, Department of Mechanical and Aeronautical Engineering, University of Pretoria, Pretoria, Septamber 2004.

[4] Ibraev, S., *A New Parallel Method for Verified Global Optimization*, PHD Thesis, July 2001.

[5] Astolfi, A., *Optimization, An Introduction*, PHD thesis, Bergischen University, October 2005.

[6] Bilbro, G., L., *Fast Stochastic Global Optimization*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 24, No. 4, April 1994.

[7] Herrera, F., Lozano, M., Molina, D., *Continuous Scatter Search: An Analysis of the Integration of Some Combination Methods and Improvement Strategies*, European Journal of Operational Research, Article in Press, 2004.

[8] Ratz, D., Csendes, T., *On the Selection of Subdirections in Interval Branch-and-Bound Methods for Global Optimization*, Journal of Global Optimization, 7:183:207, 1995.

[9] Hedar, A. R., Fulushima, M., *Tabu Search Directed by Direct Search Methods for Nonlinear Global Optimization*, European Journal of Operational Research 170: 329–349, 2006.

[10] Huyer, W., Neumayer, A., *Snobfit – Stable Noisy Optimization by Branch and Fit*, ACM Transactions on Mathematical Software, __.

[11] Ali, M., M., Storey, C., Törn, A., *Application of Some Recent Stochastic Global Optimization Algorithms to Practical Problems*, Turku Centre for Computer Science TUCS Technical Report, No 47, October 1996.

[12] Ali, M., Törn, A., Viitanen, S., *A Direct Search Simulated Annealing Algorithm for Continuous Variables*, Turku Centre for Computer Science TUCS Technical Report, No 97, March 1996.

[13] Shashicala, H., Santechi N., K., Kirthi, S., S., *A New Approach to Global Optimization Using Ideas from Nonlinear Stability Theory*, Indian Institue of Sience, Bangalore, India.

[14] Elster, C., Neumaier, A., *A Grid Algorithm for Bound Constrained Optimization of Noisy Functions*, __.

[15] Hedar, A., Fukushima, M., *Evolution Strategies Learned with Automatic Termination Criteria*, Graduate School of Informatics, Kyoto University, Kyoto, Japan.

[16] Tsoulos, I. G., Lagaris, I. E., *MinFinder: Locating All the Local Minima of a Function*, Technical Report.

[17] Parsopoulos, K. E., Vrahatis, M. N., *Particle Swarm Optimizer in Noisy and Continuously Changing Environments*.

[18] Viitanen, S., Törn, A., *Topographical Global Optimization for Constrained Prolems*, Turku Centre for Computer Science TUCS Technical Report.

[19] Bezdek, J., X, *Some Notes on Alternationg Optimization*, Lecture Notes in Computer Science, Springer Verlag.

[20] Nelder J. A., Mead R., *Downhill Simplex Method*, Computer Journal, vol. 7, pp. 308-313, 1965.

زیرنویس‌ها

[1] Simulated Annealing
[2] Disjoint Set
[3] Alternating Optimization
[4] دسترسی به پیاده‌سازی انجام‌شده از طریق آدرس http://ce.aut.ac.ir/~bozorgzadeh/global-opt.zip میسر است.
[5] دسترسی به کلیه نتایج ثبت‌شده به صورت فایل Excel از طریق آدرس http://ce.aut.ac.ir/~bozorgzadeh/go-results.zip میسر است.