IDMC2008
11-12 NOV
Amir Kabir University

یک مدل ترکیبی برای حل مسایل بهینه سازی
(اتوماتای یادگیر سلولی + بهینه‌سازی حدی)

آیدین خاتم نژاد پاکزاد محمد رضا میبدی
آزمایشگاه محاسبات نرم
دانشکده مهندسی کامپیوتر و فناوری اطلاعات
دانشگاه صنعتی امیرکبیر
تهران ایران
khatamnejad@gmail.com, mmeybodi@aut.ac.ir

چکیده: الگوریتم بهینه‌سازی حدی تعمیم یافته (GEO) یک الگوریتم ابر اکتشافی است که برای حل مسائل بهینه‌سازی استفاده می‌شود. یکی از مهمترین ویژگی‌های این الگوریتم داشتن تنها یک پارامتر است که در نتیجه استفاده از آن را در حل مسایل ساده‌تر می‌کند. سرعت هم‌گرایی این الگوریتم بسیار بالا بوده و همچنین نحوه حرکت این الگوریتم به سمت جواب به گونه‌ای است که در صورت رسیدن به جواب، جواب را با دقت بسیار بالایی تولید و محاسبه می‌کند. با این حال، علی رقم تمهیداتی که برای جلوگیری از گرفتار شدن الگوریتم در بهینه‌های محلی در نظر گرفته شده است، این الگوریتم برای مسائلی که دارای نقاط بهینه محلی زیاد و پراکنده هستند، خوب عمل نکرده و جواب‌های مطلوبی ارائه نمی‌دهد. در این مقاله برای حل این مشکل پیشنهاد می‌شود که الگوریتم بهینه‌سازی حدی تعمیم یافته در کنار اتوماتای یادگیر سلولی (CLA) قرار گیرد تا در شرایطی که به تنهایی از کارایی بالایی برخوردار نیست بتواند با کمک گرفتن از اتوماتای یادگیر سلولی عملکرد خود را بهبود بخشد. برای این منظور در این مقاله یک مدل ترکیبی که از ترکیب بهینه‌سازی حدی تعمیم یافته و اتوماتای یادگیر سلولی حاصل شده برای حل مسایل بهینه سازی پیشنهاد می‌گردد. الگوریتم پیشنهادی یک الگوریتم ابر اکتشافی بوده که مشکل گرفتار شدن در نقاط بهینه محلی را نداشته و دامنه مساله را برای یافتن جواب به شکل بهتری بررسی می‌کند. نتایج آزمایش‌های انجام گرفته موید این مساله است.

کلمات کلیدی: اتوماتای یادگیر سلولی، بهینه‌سازی حدی، بهینه‌سازی حدی تعمیم یافته

A Hybrid Model for Solving Optimization Problems

(Cellular Learning Automata + Generalized Extremal Optimization)

A. Khatamnejad Pakzad M. R. Meybodi
Computer Engineering and Information Technology Department
Amirkabir University of Technology
Tehran Iran
(khatamnejad@gmail.com, mmeybodi@aut.ac.ir)

Abstract: Generalized Extremal Optimization (GEO) is a meta-heuristic algorithm for solving optimization problems. Some of the most notable features of this algorithm are its having only one parameter - which makes it very easy to use –, high converging speed and very accurate solutions whenever finds domain optimum. In spite of the corrections applied, still with great deal of probability, GEO algorithm converges to local optima rather than problem domain optimum; thus, this algorithm is not suitable for problems with many distributed local optima in their domain. In this paper, a new hybrid algorithm which is obtained by combining Generalized Extremal Optimization and Cellular Learning Automata (CLA) is proposed. This proposed method (CLA-GEO) examines the problem domain more efficiently and has the ability to escape from local minima. To show the efficiency of the proposed method it is tested on several function optimization problems. The results obtained for the proposed method are compared with the results obtained for GEO method. The results of comparison show the superiority of the proposed method.

Keywords: Cellular Learning Automata, Extremal Optimization, Generalized Extremal Optimization

1. مقدمه

الگوریتم‌های مبتنی بر بهینه‌سازی حدی[1] بر خلاف دیگر الگوریتم‌های بهینه‌سازی نظیر الگوریتم ژنتیک که همراه از مجموعه‌ای از جواب‌ها (جمعیت) تشکیل شده‌اند، در هر لحظه تنها یک جواب در اختیار داشته و در هر مرحله این جواب را بهبود می‌بخشند؛ بهبود جواب بوسیله انتخاب یکی از اجزای جواب و جایگزین کردن مقدار آن با مقداری جدید انجام می‌شود. برای انتخاب جزء تعویضی، الگوریتم از شایستگی محلی اجزا استفاده می‌کند؛ بنابراین این الگوریتم‌ها علاوه بر تابع شایستگی کلی جواب، به تابعی برای ارزیابی شایستگی اجزای جواب (شایستگی محلی) نیز احتیاج دارند. واضح است که این تابع وابسته به نوع مساله بوده و برای هر مساله باید به صورت جداگانه‌ای تعریف و پیاده‌سازی شود.

الگوریتم بهینه‌سازی حدی تعمیم یافته[2]، تعمیم یافته یکی از الگوریتم‌های مشتق شده از الگوریتم بهینه‌سازی حدی به نام الگوریتم بهینه‌سازی حدی با پارامتر τ است. این الگوریتم یک الگوریتم ابر اکتشافی[3] است که نیاز الگوریتم بهینه‌سازی حدی به تابع ارزیابی شایستگی محلی اجزا را از بین می‌برد. از ویژگی‌های جالب این الگوریتم می‌توان به سرعت بالای همگرایی ، دقت بالای جواب‌ها و تعداد پارامترهای کم (این الگوریتم تنها یک پارامتر به نام τ برای تنظیم دارد) اشاره کرد. هر چند این الگوریتم جواب‌های بسیار دقیقی برای مسائل تولید می‌کند، احتمال گرفتار شدن آن در نقاط بهینه محلی بالا است. در واقع این الگوریتم دامنه مساله را به شکل خوبی جستجو نمی‌کند و در نتیجه در اولین نقطه بهینه‌ای که پیدا می‌کند متوقف می‌شود. این مساله باعث شده است که استفاده از این الگوریتم تنها به مجموعه کوچکی از مسائل (مسائلی با تنها یک نقطه بهینه) محدود شود.

در این مقاله برای حل این مشکل پیشنهاد می‌شود که الگوریتم بهینه‌سازی حدی تعمیم یافته در کنار اتوماتای یادگیر سلولی[4] قرار گیرد تا در شرایطی که به تنهایی از کارایی بالایی برخوردار نیست بتواند با کمک گرفتن از آن عملکرد خود را بهبود بخشد. برای انتخاب اتوماتای یادگیر سلولی سه دلیل اصلی وجود دارد:

1. ساختار الگوریتم و ساختمان داده مورد استفاده الگوریتم بهینه‌سازی حدی تعمیم یافته، کاملا قابل انطباق با اتوماتای سلولی است؛ هر دوی این روش‌ها از آرایه‌ای از سلول‌ها استفاده می‌کنند.
2. نحوه انتخاب سلول تعویضی در هر مرحله از الگوریتم انعطاف‌پذیر نبوده و به هیچ نحوی با شرایط مساله تطبیق پیدا نمی‌کند. بنابراین استفاده از ساختاری انعطاف‌پذیر مانند اتوماتای یادگیر سلولی که بتواند با محیط (مساله) تعامل داشته و خود را با آن تطبیق دهد جالب به نظر می‌رسد.
3. به دلیل اینکه (1) الگوریتم بهینه‌سازی حدی تعمیم یافته، به نقطه شروع الگوریتم بسیار وابسته بوده و با نقاط شروع مختلف جواب‌های کاملا متفاوتی - گاه صحیح و گاه کاملا غلط- را تولید می‌کند و (2) سرعت همگرا شدن این الگوریتم زیاد بوده و معمولا در مراحل اولیه در نقطه‌ای (یکی از نقاط بهینه) متوقف می‌شود، تشخیص توقف و گرفتار شدن الگوریتم در نقاط بهینه و شروع مجدد آن از نقطه‌ای دیگر می‌تواند مفید باشد که این کار توسط اتوماتای یادگیر سلولی انجام می‌پذیرد.

نتایج آزمایش‌ها نشان می‌دهد که الگوریتم ترکیبی پیشنهادی که CLA-GEO نامیده می‌شود، در مقایسه با الگوریتم بهینه‌سازی حدی تعمیم یافته نتایج بهتری تولید کرده و با حفظ دقت بالای الگوریتم بهینه‌سازی حدی تعمیم یافته، از گرفتار شدن در نقاط بهینه محلی اجتناب می‌کند. ادامه این مقاله بدین صورت سازماندهی شده است که در بخش 2 بحران‌های خودسازمانده و بهینه‌سازی حدی و در بخش 3 اتوماتای

---

[1] Extremal Optimization
[2] Generalized Extremal Optimization
[3] Meta Heuristic
[4] Cellular Learning Automata

سلولی، اتوماتای یادگیر و اتوماتای یادگیر سلولی به اختصار شرح داده می‌شوند. در بخش 4 الگوریتم پیشنهادی به طور کامل معرفی می‌شود و دو بخش انتهایی به ارائه نتایج آزمایش‌ها و نتیجه‌گیری اختصاص دارد.

2. بحران‌های خود سازمانده[1] و بهینه‌سازی حدی تعمیم یافته

الگوریتم بهینه‌سازی حدی تعمیم یافته - همانطور که از نام آن مشخص است - تعمیم یافته الگوریتم دیگری به نام بهینه‌سازی حدی می‌باشد که از اعمال تغییراتی در مدل تکاملی بک- اسنپن به وجود آمده است. مدل بک-اسنپن نیز خود نمونه‌ای از یک سیستم با ویژگی بحران خود سازمانده است. در این بخش ابتدا بحران‌های خود سازمانده و مدل بک-اسنپن به صورت خلاصه معرفی می‌شوند. سپس بهینه‌سازی حدی و بهینه‌سازی حدی با پارامتر τ معرفی و نهایتا الگوریتم بهینه‌سازی حدی تعمیم یافته شرح داده می‌شود.

3. بحران خود سازمانده

بحران خود سازمانده [3] یکی از ویژگی‌های سیستم‌های طبیعی است که بسیاری از رفتارهای پیچیده این سیستم‌ها را توصیف می‌کند. این مفهوم اولین بار توسط بک[2] برای توصیف رفتار مدل توده شن معرفی شد. در حال حاضر از این مفهوم در شبیه‌سازی و پیش‌بینی زلزله و رانش‌زمین، شبیه‌سازی آتش‌سوزی در جنگل‌ها، بررسی انتشار بیماری‌ها، تغییرات آب و هوا و بررسی بازار خرید و فروش کالاها استفاده می‌شود.

سیستم‌هایی که دارای ویژگی بحران خود سازمانده هستند معمولا از یک حالت پایدار شروع به کار کرده و در طول زمان به یک نقطه بحرانی[3] نزدیک می‌شوند. در این نقطه، کوچکترین تغییری در وضعیت، باعث می‌شود که سیستم با یک رفتار سریع و وسیع که با نام بهمن[4] شناخته می‌شود، خود را به یک حالت پایدار برساند. این سیستم‌ها تا مرز آشفتگی رفته ولی توسط بهمن به حالت عادی بر می‌گردند. تعداد و وسعت بهمن‌ها از قانون توان[5] پیروی می‌کند:

$$y = ax^k \qquad (1)$$

در این رابطه x وسعت بهمن (با توجه به نوع سیستم می‌تواند واحدهای مختلفی داشته باشد)، a و k مقادیر ثابت که برای هر سیستم به صورت جداگانه تعریف می‌شود و y تعداد دفعات وقوع بهمن است. به عنوان مثال اگر زمین لرزه را به عنوان بهمن در نظر بگیریم، تعداد زمین لرزه‌ها با بزرگی‌های مختلف از این قانون پیروی می‌کند.

در حال حاضر روشی برای اثبات اینکه آیا سیستمی دارای ویژگی بحران خود سازمانده است یا خیر وجود ندارد و تشخیص وجود این ویژگی در یک سیستم به صورت تجربی انجام می‌شود. با این حال، تعدادی خصوصیت مشترک بین این نوع سیستم‌ها مشاهده شده است که عبارتند از: تاثیرگذاری اجزا سیستم به صورت محلی، قانون‌های ساده بین اجزا، انتقال تغییرات بوسیله قوانین بین اجزا، محرک خارجی با نرخ کم و پیروی حداقل یکی از اجزای سیستم از قانون توان. در صورتی که این ویژگی‌ها در سیستمی مشاهده شوند، می‌توان انتظار داشت که ویژگی بحران خود سازمانده نیز در این سیستم وجود دارد.

3-1. مدل بک- اسنپن[6]

مدل بک-اسنپن مدلی برای نمایش تکامل گونه‌های موجودات است که بر اساس مفهوم بحران خود سازمانده ایجاد شده است. این مدل از آرایه‌ای از سلول‌ها تشکیل شده است که در آن هر سلول نماینده یک گونه جانوری است؛ ابتدا و انتهای آرایه سلول‌ها در کنار هم قرار دارند به

---

[1] Self-Organized Criticality (SOC)

[2] Bak
[3] Critical Point
[4] Avalanche
[5] Power Law
[6] Bak-Sneppen Model

نحوی که آرایه صورت یک حلقه در می‌آید. در این مدل به هر گونه جانوری یک شایستگی اختصاص داده می‌شود که نشان دهنده شایستگی و مقدار تطابق آن با محیط است؛ این شایستگی به صورت عددی در $[0,1]$ در هر سلول نگهداری می‌شود.

روش کار الگوریتم مدل بک-اسنپن به این صورت است که در ابتدا همه سلول‌ها با عددی اتفاقی مقداردهی می‌شوند. سپس تا توقف الگوریتم در هر مرحله، سلولی که کمترین شایستگی را دارد (در صورتی که چندین سلول با کمترین شایستگی وجود داشتند یکی از آنها به صورت اتفاقی انتخاب می‌شود) به همراه سلول‌های چپ و راست آن مجددا به صورت اتفاقی مقداردهی می‌شوند. توقف الگوریتم می‌تواند به صورت دستی انجام شود و یا با تعریف آستانه‌ای به خود الگوریتم محول شود.

اگر در مرحله جاری، مجموعه‌ای از کوچکترین شایستگی در هر کدام از مراحل قبل را تشکیل دهیم، بزرگترین عضو این مجموعه شایستگی بحرانی نامیده می‌شود. واضح است که شایستگی بحرانی همواره مقدار ثابتی ندارد و در طول اجرا به صورت صعودی تغییر پیدا می‌کند. آزمایش‌هایی که بر روی این مدل انجام شده است نشان می‌دهد که شایستگی بحرانی دارای حدی برابر 0.667 است که با گذشت زمان به سمت آن میل می‌کند.

همانطور که اشاره شد مدل بک-اسنپن دارای ویژگی بحران خود سازمانده است، بنابراین باید دارای پدیده بهمن نیز باشد. بهمن در این مدل زمانی شروع می‌شود که مقدار اختصاص داده به یک سلول (سلول با شایستگی کم یا یکی از همسایه‌های آن) از شایستگی بحرانی آن مرحله کمتر شود. پایان بهمن زمانی است که با اختصاص مقادیر جدید به سلول‌ها، هیچ سلولی با شایستگی کمتر از شایستگی بحرانی مرحله وجود نداشته باشد. بزرگی بهمن نیز معادل تعداد سلول‌هایی است که در طول بهمن شایستگیی کمتر از شایستگی بحرانی پیدا می‌کنند.

مدل بک-اسنپن را می‌توان به اینصورت به دنیای واقعی ارتباط داد که تغییر خانه با کوچکترین شایستگی، معادل این واقعیت است که گونه‌ها با شایستگی‌های کمتر بیشتر در معرض انقراض قرار دارند و مقداردهی همسایه‌های آن نیز به این مطلب برمی‌گردد که در طبیعت تغییرات یک گونه در گونه‌های مجاور آن نیز تاثیرگذار است.

3-2. بهینه‌سازی حدی

بهینه‌سازی حدی [4] الگوریتمی اکتشافی است که از مدل بک-اسنپن الهام گرفته شده است. همانطور که از نام آن مشخص است این الگوریتم برای حل مسائل بهینه‌سازی استفاده می‌شود. در این الگوریتم سلول‌های مدل بک-اسنپن برابر اجزای جواب مساله قرار داده می‌شوند. جواب در هر مرحله (تکرار حلقه اصلی الگوریتم) با جایگزینی مقدار بدترین سلول (سلول با کمترین شایستگی) با مقداری دیگر بهبود داده می‌شود. در این الگوریتم بر خلاف مدل بک-اسنپن، با تغییر یک سلول (گونه در مدل بک-استپن) سلول‌های همسایه آن تغییر داده نمی‌شوند.

مشخص‌ترین تفاوت الگوریتم بهینه‌سازی حدی و دیگر الگوریتم‌های اکتشافی نظیر الگوریتم ژنتیک، نیاز الگوریتم به دانستن شایستگی سلول‌ها (شایستگی محلی) علاوه بر شایستگی کلی جواب است. در واقع این ویژگی الگوریتم بهینه‌سازی حدی، هسته مرکزی و عامل کار آن است. الگوریتم بهینه‌سازی حدی به صورت کلی شکل زیر را دارد:

1. جوابی به عنوان جواب اولیه تولید شده و به عنوان جواب جاری و همچنین به عنوان بهترین جواب در نظر گرفته می‌شود.
2. بدترین سلول جواب جاری انتخاب شده و مقدار آن با مقدار دیگری جایگزین می‌شود.
3. جواب ایجاد شده به عنوان جواب جاری در نظر گرفته می‌شود.
4. در صورتی که جواب جاری بهتر از بهترین جواب باشد به عنوان بهترین جواب نیز انتخاب می‌شود.
5. مرحله 2 تا رسیدن به جواب مورد نیاز تکرار می‌شود.
6. بهترین جواب برگردانده می‌شود.

الگوریتم بهینه‌سازی حدی برای حل مسائلی نظیر تقسیم گراف به دو زیر گراف، رنگ آمیزی گراف با سه رنگ و فروشنده دوره‌گرد مورد استفاده قرار گرفته و جواب‌های مطلوبی ارائه داده است.

3-3. بهینه‌سازی حدی با پارامتر τ

هرچند الگوریتم بهینه‌سازی حدی جواب‌های مطلوبی برای مسائل به کار گرفته شده ارائه می‌دهد، با این حال احتمال گرفتار شدن آن در نقاط بهینه محلی و متوقف شدن آن در مکانی به جز جواب مساله زیاد است. برای حل این مشکل پارامتر جدیدی به نام τ به الگوریتم بهینه‌سازی حدی اضافه شده و الگوریتم جدیدی به نام بهینه‌سازی حدی با پارامتر τ [4] معرفی شده است؛ تفاوت این الگوریتم جدید و الگوریتم

بهینه‌سازی حدی به مرحله انتخاب سلول تعویضی مربوط می‌شود. در این الگوریتم برای پیدا کردن سلول تعویضی، ابتدا سلول‌ها بر اساس شایستگی و به صورت صعودی مرتب می‌شوند. سپس به هر سلول مقداری برابر P_k که از رابطه 2 قابل محاسبه است اختصاص داده می‌شود؛ در این رابطه k برابر ترتیب سلول در فهرست مرتب شده است:

$$P_k \propto k^{-\tau} \qquad (2)$$

پس از نسبت دادن مقادیر P_k به سلول‌ها، سلول تعویضی به اینصورت انتخاب می‌شود که:

1. در حلقه‌ای به طور اتفاقی یکی از سلول‌ها انتخاب می‌شود.
2. عددی به صورت اتفاقی بین 0 و 1 انتخاب می‌شود.
3. در صورتی که عدد تولید شده کوچکتر از P_k سلول باشد سلول به عنوان سلول تعویضی انتخاب می‌شود و حلقه پایان می‌پذیرد، در غیر اینصورت حلقه تکرار می‌شود.

مراحل کلی الگوریتم بهینه‌سازی حدی با پارامتر τ به شرح زیر است:

1. جواب اولیه‌ای تولید شده به عنوان جواب جاری و همچنین به عنوان بهترین جواب در نظر گرفته می‌شود.
2. سلول‌ها با توجه به شایستگی آنها به صورت صعودی مرتب می‌شوند.
3. با توجه به رابطه 2 یک سلول به عنوان سلول تعویضی انتخاب و مقداردهی مجدد می‌شود.
4. جواب تولید شده به عنوان جواب جاری در نظر گرفته می‌شود.
5. در صورتی که جواب جاری بهتر از بهترین جواب باشد به عنوان بهترین جواب نیز انتخاب می‌شود.
6. مرحله 2 تا رسیدن به جواب مورد نیاز تکرار می‌شود.
7. بهترین جواب برگردانده می‌شود.

لازم به ذکر است که الگوریتم ارائه شده و نحوه انتخاب سلول تعویضی ساده‌ترین شکل پیاده‌سازی بهینه‌سازی حدی با پارامتر τ بوده و بسته به نوع مساله قابل تغییر خواهد بود.

بهینه‌سازی حدی با پارامتر τ مانند الگوریتم بهینه‌سازی حدی برای حل مسائلی نظیر تقسیم گراف به دو زیر گراف، رنگ آمیزی گراف با سه رنگ و فروشنده دوره‌گرد به کار گرفته شده است و جواب‌های مطلوب‌تری از الگوریتم مادر خود ارائه داده است. همچنین نتایج بدست آمده توسط این الگوریتم هم از نظر دقت و هم از نظر سرعت، کاملا قابل رقابت با الگوریتم‌هایی نظیر شبیه‌سازی تابکاری[1] و الگوریتم ژنتیک نشان داده است.

3-4. بهینه‌سازی حدی تعمیم یافته

همانطور که مطرح شد استفاده از الگوریتم بهینه‌سازی حدی نیازمند تعریف تابعی برای محاسبه شایستگی محلی سلول‌های جواب است و علاوه بر آن چگونگی تعویض سلول تعویضی به شکل مساله وابسته است. بنابراین نمی‌توان از آن به راحتی و بدون نیاز به پیاده‌سازی مجدد، برای حل دامنه وسیعی از مسائل استفاده کرد. به این منظور الگوریتم بهینه‌سازی حدی تعمیم یافته [5] بر اساس الگوریتم بهینه‌سازی حدی با پارامتر τ معرفی شده است؛ این الگوریتم یک الگوریتم ابر اکتشافی مانند الگوریتم ژنتیک و شبیه‌سازی تابکاری است که برای حل دسته وسیعی از مسائل استفاده می‌شود.

ساختار الگوریتم بهینه‌سازی حدی تعمیم یافته کاملا شبیه الگوریتم بهینه‌سازی حدی با پارامتر τ است. برای استفاده از الگوریتم بهینه‌سازی حدی تعمیم یافته، ابتدا متغیرهای مساله باید به یک رشته دودویی نگاشت شوند؛ به عنوان مثال اگر مساله پیدا کردن نقطه بهینه یک تابع دو بعدی باشد، ابتدا متغیرهای x و y باید به صورت دودویی درآمده و سپس در کنار هم قرار داده شوند تا جواب به شکل آرایه‌ای از بیت‌ها در آید. به این ترتیب هر سلول در الگوریتم بهینه‌سازی حدی تعمیم یافته معادل یک بیت از جواب خواهد بود؛ بنابراین تعداد سلول‌ها به دقت مورد نیاز و اندازه دامنه مساله بستگی دارد. واضح است که دامنه مقادیر مجاز برای سلول‌ها در الگوریتم اخیر، 0 یا 1 می‌باشد و از این نظر در مقایسه با الگوریتم بهینه‌سازی حدی بسیار ساده‌تر است.

---

[1] Simulated Annealing

مساله دیگری که در الگوریتم بهینه‌سازی حدی تعمیم یافته مطرح است، محاسبه شایستگی سلول‌ها است. همانطور که در بخش‌های اشاره شد، محاسبه شایستگی محلی مهمترین مرحله در الگوریتم بهینه‌سازی حدی است. در دو الگوریتم بهینه‌سازی حدی و بهینه‌سازی حدی با پارامتر τ، شایستگی محلی به صورت تابعی وابسته به مساله تعریف می‌شود؛ در الگوریتم بهینه‌سازی حدی تعمیم یافته روشی برای محاسبه شایستگی سلول‌ها ارائه شده که مستقل از نوع و شکل مساله است. طبق این روش در هر مرحله بر اساس جواب جاری و به تعداد سلول‌های آن، جواب‌هایی تولید می‌شود که به عنوان فرزندان جواب جاری شناخته می‌شوند. این جواب‌ها از هم متمایز بوده و هر کدام تنها در مقدار یک سلول با جواب جاری اختلاف دارند. به عنوان مثال اگر رشته دودویی جواب جاری برابر 01001 باشد، پنج فرزند با مقادیر 01000، 01010، 01101، 00001 و 11001 برای آن تولید می‌شود. بعد از تولید شدن جواب‌های جدید، شایستگی هر کدام از آنها بوسیله تابع شایستگی کلی محاسبه می‌شود؛ شایستگی هر سلول برابر خواهد بود با شایستگی فرزندی از جواب جاری که در مقدار آن سلول با جواب جاری اختلاف دارد منهای شایستگی جواب جاری. به این ترتیب شایستگی محلی اجزای جواب بدون نیاز به تابعی برای محاسبه شایستگی محلی و تنها بوسیله تابع شایستگی کلی مساله محاسبه می‌شوند. لازم به ذکر است که خروجی تابع شایستگی کلی برابر مقدار تابع در نقطه (جواب) مورد نظر است. همچنین باید توجه داشت که برای توابعی که مقدار بهینه یک نقطه ماکزیمم است شایستگی هر سلول باید منفی در نظر گرفته شود. مراحل الگوریتم بهینه‌سازی حدی تعمیم یافته به ترتیب در ادامه نشان داده شده‌اند:

1. جواب اولیه‌ای تولید شده و به عنوان جواب جاری و همچنین به عنوان بهترین جواب در نظر گرفته می‌شود.
2. فرزندان جواب جاری تولید شده و شایستگی آنها محاسبه می‌شود.
3. جواب‌های تولید شده (فرزندان جواب جاری) با توجه به شایستگی آنها به صورت صعودی مرتب می‌شوند.
4. با توجه به رابطه 2 فرزندی انتخاب می‌شود.
5. فرزند انتخاب شده به عنوان جواب جاری در نظر گرفته می‌شود.
6. در صورتی که جواب جاری بهتر از بهترین جواب باشد به عنوان بهترین جواب نیز انتخاب می‌شود.
7. مرحله 2 تا رسیدن به جواب مورد نیاز تکرار می‌شود.
8. بهترین جواب برگردانده می‌شود.

مهمترین مزیت الگوریتم بهینه‌سازی حدی تعمیم یافته در مقایسه با الگوریتم‌های اکتشافی دیگر، دارا بودن تنها یک پارامتر (τ) است و از این نظر استفاده از آن بسیار ساده می‌باشد. سرعت هم‌گرا شدن این الگوریتم بسیار بالا بوده و همچنین نحوه حرکت این الگوریتم به سمت جواب به گونه‌ای است که در صورت رسیدن به جواب، جواب را با دقت بسیار بالایی تولید و محاسبه می‌کند. با این حال، علی رقم تمهیداتی که برای جلوگیری از گرفتار شدن الگوریتم در نقاط بهینه محلی در نظر گرفته شده است، این الگوریتم برای مسائلی که دارای نقاط بهینه محلی زیاد و پراکنده هستند، خوب عمل نکرده و جواب‌های مطلوبی ارائه نمی‌دهد.

4. اتوماتای سلولی، اتوماتاهای یادگیر و اتوماتای یادگیر سلولی

در این بخش از مقاله اتوماتای سلولی، اتوماتاهای یادگیر و اتوماتای یادگیر سلولی به اختصار شرح داده می‌شود.

4-1. اتوماتای سلولی[1] (CA)

اتوماتای سلولی یک مدل ریاضی برای سیستم‌هایی است که از اجزا و مولفه‌های ساده تشکیل شده، پویا بوده، با مرور زمان تغییر کرده و بر اساس ارتباط محلی بین اجزای خود فعالیت می‌کنند. اجزای سیستم در مدل اتوماتای سلولی بوسیله مجموعه‌ای منظم از سلول‌ها نمایش داده می‌شوند که در آن هر سلول معادل یکی از اجزای سیستم است. در هر قدم زمانی به هر سلول مقداری از مجموعه‌ای نامتناهی نسبت داده می‌شود که نشان دهنده وضعیت سلول است. این مقداردهی بر اساس مقدار جاری سلول، سلول‌های همسایه آن و قانون تعریف شده برای اتوماتای سلولی

[1] Cellular Automata

انجام می‌شود؛ از آنجایی که قانون در نظر گرفته شده تنها مقدار سلول و همسایه‌های آن را مورد استفاده قرار می‌دهد، قانون محلی نیز نامیده می‌شود.

دسته‌بندی‌های مختلفی را می‌توان برای اتوماتای سلولی در نظر گرفت که از جمله آنها می‌توان به موارد زیر اشاره کرد:

- اگر مجموعه مقادیر سلول‌ها از دو عنصر تشکیل شده باشد، اتوماتا از نوع دودویی و در صورتی که تعداد عناصر بیشتر از دو عدد باشد اتوماتا از نوع چند مقداره خواهد بود.
- در صورتی که قوانین شکلی قطعی داشته باشند اتوماتا از نوع قطعی و در صورتی که احتمالی باشند اتوماتا از نوع احتمالی می‌باشد.
- هرگاه تمامی سلول‌های اتوماتا همزمان مقداردهی شوند اتوماتا از نوع همگام و در مقابل اگر سلول‌ها با ترتیبی خاص و غیر همزمان مقداردهی شوند اتوماتا از نوع ناهمگام خواهد بود.
- اگر سلول‌های اتوماتا به شکلی منظم در کنار یکدیگر قرار گرفته باشند و تشکیل آرایه‌ای یک بعدی، دو بعدی و یا چند بعدی را بدهند، اتوماتا از نوع منظم و در صورتی که سلول‌ها به شکلی نامنظم (مثلا به عنوان گره‌های یک گراف) قرار گرفته باشند اتوماتا از نوع نامنظم خواهد بود.

هر چند اتوماتای سلولی امکانات زیادی را برای شبیه‌سازی و تعریف سیستم‌های گوناگون در اختیار قرار می‌دهد با این حال استفاده از آن برای شبیه‌سازی سیستم‌های طبیعی، بزرگ و ناشناخته با یک اشکال بزرگ همراه است و آن تعیین فرم قطعی قوانین است؛ زیرا در اغلب سیستم‌ها وجود نویز و عدم قطعیت مانع از تعیین قوانینی قطعی برای سیستم می‌شود. البته راهکار احتمالی کردن قوانین برای حل این مشکل پیشنهاد شده است. با این حال، تعیین مقادیر احتمالات قوانین برای سیستم‌های ناشناخته خود مشکلی بزرگ است که مانع از استفاده مطلوب از این مدل می‌شود[9][10].

2-4. اتوماتای یادگیر[1]

اتوماتای یادگیر [1] ماشینی است که می‌تواند تعداد محدودی عمل را انجام دهد؛ هرگاه این ماشین عملی را انتخاب می‌کند، عمل انتخاب شده توسط محیط ارزیابی شده و نتیجه آن به صورت یک سیگنال بازخوردی مثبت (در صورت مناسب بودن عمل) یا منفی (در صورت نامناسب بودن عمل) به اتوماتا بازگردانده می‌شود. مقدار این سیگنال در انتخاب اعمال بعدی تاثیر می‌گذارد. هدف این فرایند این است که اتوماتا بعد از گذشت مدتی به سمت مناسب‌ترین عمل خود در محیط میل کرده و یا به عبارت دیگر یاد می‌گیرد که کدام عمل بهترین عمل است. نحوه تعامل اتوماتای یادگیر و محیط در شکل 1 نشان داده شده است. همانطور که در شکل مشاهده می‌شود، محیط یک اتوماتای یادگیر همانند یک تابع رفتار می‌کند؛ ورودی این تابع عمل انتخاب شده توسط اتوماتا و خروجی آن سیگنال بازخوردی است.

شکل (1) : تعامل اتوماتای یادگیر و محیط

دو نوع پیاده‌سازی برای اتوماتای یادگیر وجود دارد که عبارتند از اتوماتای یادگیر با ساختار ثابت[2] و اتوماتای یادگیر با ساختار متغیر[3]. اتوماتای یادگیر با ساختار متغیر که در این مقاله استفاده شده است از برداری به نام بردار احتمال انتخاب اعمال استفاده می‌کند. جمع احتمالات این بردار برابر 1 و تعداد اعضای آن برابر تعداد اعمال اتوماتای یادگیر می‌باشد. اتوماتای یادگیر برطبق این بردار عمل بعدی خود را انتخاب می‌کند و بعد از

[1] Learning Automata
[2] Fixed Structure Learning Automata
[3] Variable Structure Learning Automata

دریافت پاسخ محیط آن را به روز می‌کند. الگوریتمی که برای به روز رسانی بردار احتمال بر اساس ورودی (پاسخ محیط) مورد استفاده قرار می‌گیرد الگوریتم یادگیری نام دارد. یکی از متداول‌ترین این الگوریتم‌ها، الگوریتم یادگیری خطی است. در این الگوریتم هرگاه در مرحله n ام عمل i انتخاب شده باشد و این عمل از طرف محیط پاداش گرفته باشد بردار احتمال با توجه به رابطه 3 به روز می‌شود:

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)]$$
$$p_j(n+1) = (1-a)\,p_j(n) \qquad \forall j \quad j \neq i \tag{3}$$

و در صورتی که عمل انتخاب شده از طرف محیط جریمه شود به روز رسانی طبق رابطه 4 انجام می‌شود:

$$p_i(n+1) = (1-b)\,p_i(n)$$
$$p_j(n+1) = (b/r-1) + (1-b)\,p_j(n) \quad \forall j \quad j \neq i \tag{4}$$

در این روابط a پارامتر پاداش و b پارامتر جریمه می‌باشند.

3-4. اتوماتای یادگیر سلولی (CLA)

بسیاری از مسائل را نمی‌توان با یک اتوماتای یادگیر حل کرد. در این موارد مجموعه‌ای از اتوماتاهای یادگیر می‌توانند به حل مساله کمک کنند. در [14] مدلی متشکل از دو مفهوم اتوماتای سلولی و اتوماتای یادگیر ارائه شده است که با نام اتوماتای یادگیر سلولی شناخته می‌شود. در این مدل هر سلول در اتوماتای سلولی با یک اتوماتای یادگیر جایگزین شده است. به این ترتیب این مدل نه تنها امکان استفاده از چندین اتوماتای یادگیر را به صورت توام فراهم می‌کند بلکه مشکل تعیین فرم قطعی قوانین در اتوماتای سلولی را نیز مرتفع می‌کند.

نحوه عملکرد و به روز رسانی سلول‌های اتوماتای یادگیر سلولی شباهت بسیار زیادی به اتوماتای سلولی دارد با این تفاوت که قبل از انجام اعمال به روز رسانی، ابتدا هر کدام از اتوماتاهای یادگیر عملی را انتخاب می‌کنند، سپس عمل انتخاب شده توسط هر سلول در کنار اعمال انتخاب شده توسط اتوماتاهای همسایه آن به وسیله قانون محلی اتوماتای سلولی بررسی و نتیجه آن به عنوان پاسخ محیط و سیگنال بازخوردی به سلول برگردانده می‌شود. در نهایت، اتوماتای یادگیر هر سلول با توجه به سیگنال بازخوردیی که از محیط دریافت کرده است، یا به وضعیت جدیدی می‌رود (در صورتی که ساختار ثابتی داشته باشد) و یا توسط الگوریتم یادگیری بردار احتمال خود را به روز می‌کند (در صورتی که ساختار متغیری داشته باشد).

انواع مختلفی از همسایگی‌ها برای اتوماتای سلولی و اتوماتای یادگیر سلولی وجود دارد که با توجه به کاربرد و شرایط مساله انتخاب و یا تعریف می‌شوند. پرکاربردترین همسایگی‌هایی که در این اتوماتاها استفاده می‌شوند ون نیومن، مور، اسمیت و کول می‌باشند.

قانون‌های محلی برای اتوماتای یادگیر سلولی بر اساس چگونگی استفاده از وضعیت و اعمال سلول‌ها به سه دسته قانون عمومی[1]، قانون کلی[2] و قانون خارجی کلی[3] تقسیم می‌شوند. برای اطلاعات بیشتر در باره اتوماتای سلولی یادگیر می‌توان به مراجع [2][15][17][18] مراجعه نمود.

5. الگوریتم ترکیبی پیشنهادی (CLA-GEO)

هدف اصلی ترکیب اتوماتای یادگیر سلولی و بهینه‌سازی حدی تعمیم یافته ایجاد یک الگوریتم ابر اکتشافی جدید برای حل دسته بزرگتری از مسائل با حفظ دقت الگوریتم بهینه‌سازی حدی تعمیم یافته بوده است. لازم به ذکر است که روشی که برای ترکیب این دو مفهوم انتخاب شده است تنها یکی از انواع روش‌هایی است که می‌توان برای ترکیب این دو مفهوم استفاده کرد.

الگوریتم CLA-GEO از نظر کلیات شباهت بسیار زیادی به الگوریتم بهینه‌سازی حدی تعمیم یافته دارد و برای برقراری ارتباط با اتوماتای سلولی، قسمت‌هایی از آن تغییر داده شده و یا قسمت‌هایی به آن اضافه شده است. همچنین در الگوریتم CLA-GEO به جای پارامتر τ دو پارامتر بهره[1] و آستانه[2] معرفی شده‌اند.

---

[1] General
[2] Totalistic
[3] Outer totalistic

الگوریتم CLA-GEO در دو قسمت از اتوماتای یادگیر سلولی استفاده می‌کند: (1) انتخاب سلول تعویضی و (2) تشخیص متوقف شدن الگوریتم. در ادامه هر کدام از این دو وظیفه به همراه ساختار الگوریتم، چگونگی به روز رسانی اتوماتا و چگونگی شروع مجدد الگوریتم توضیح داده می‌شود.

ساختار CLA-GEO: اضافه کردن اتوماتای یادگیر سلولی به الگوریتم بهینه‌سازی حدی تعمیم یافته به اینصورت انجام شده است که آرایه سلول‌ها در الگوریتم بهینه‌سازی حدی به یک اتوماتای یادگیر سلولی تبدیل می‌شود؛ هر سلول در این اتوماتا دارای دو مقدار است: (1) یک مقدار دودویی که معادل مقدار سلول در الگوریتم بهینه‌سازی حدی تعمیم یافته است و (2) یک مقدار عددی در [0,1] که معادل احتمال تغییر سلول است. اتوماتاهای یادگیر -که در این ساختار معادل سلول‌ها هستند- دارای دو عمل (1) تغییر مقدار سلول و (2) بدون تغییر نگاه داشتن مقدار سلول می‌باشند. هرگاه از اتوماتای یادگیر سلول درخواست می‌شود تا عملی را انتخاب کند، اتوماتا با احتمالی برابر با مقدار احتمال تغییر سلول یکی از این دو عمل را انتخاب می‌کند.

هرگاه سلولی پاداش دریافت می‌کند احتمال تغییر مقدار آن سلول به اندازه پارامتر بهره کم می‌شود و در صورتی که سلول جریمه شود احتمال تغییر آن به اندازه پارامتر بهره زیاد می‌شود. به این ترتیب سلول‌هایی که جریمه می‌شوند با احتمال بیشتری در معرض تعویض قرار می‌گیرند، حال آنکه سلول‌هایی که پاداش می‌گیرند با احتمال بیشتری بدون تغییر باقی می‌مانند. این روش پاداش دادن و جریمه کردن موجب می‌شود که سلول‌های که تغییر مقدار آنها باعث بهتر شدن جواب مساله شده است به راحتی تعویض نشده و حتی الامکان بدون تغییر باقی بمانند.

انتخاب سلول تعویضی: برای محاسبه شایستگی محلی سلول‌ها (اجزای جواب)، این الگوریتم نیز مانند الگوریتم بهینه‌سازی حدی تعمیم یافته، فرزندان جواب جاری را تولید کرده و سپس با محاسبه شایستگی کلی هر کدام و با توجه به شایستگی جواب جاری به صورتی که توضیح داده شد شایستگی سلول‌ها را محاسبه می‌کند.

انتخاب سلول در این الگوریتم با الگوریتم بهینه‌سازی حدی تعمیم یافته تفاوت دارد؛ انتخاب سلول تعویضی به اینصورت انجام می‌شود که سلول‌ها به صورت گردشی از بدترین سلول (سلول با کمترین شایستگی) تا بهترین سلول عملی را انتخاب می‌کنند تا نهایتا یکی از آنها عمل تغییر مقدار را انتخاب کنند. اولین سلولی که عمل تغییر مقدار را انتخاب کند عملیات انتخاب سلول تعویضی به پایان می‌رسد.

به روز رسانی اتوماتای یادگیر سلولی: بعد از انتخاب سلول تعویضی، تمامی سلول‌هایی که عمل بدون تغییر ماندن را انتخاب کرده و یا انتخاب عملی از آنها درخواست نشده است به جز سلول‌های همسایه سلول انتخاب شده جریمه می‌شوند. علت این کار بالا بردن احتمال انتخاب سلول‌های بدون تغییر در مراحل بعدی است؛ به این ترتیب احتمال تعویض سلولی که در تعداد مراحل زیادی انتخاب نشده است بسیار بالا رفته و سلول در معرض تعویض قرار می‌گیرد. این نوع جریمه کردن باعث می‌شود که نقاط بیشتری از دامنه مورد بررسی قرار بگیرد.

در صورتی که جواب حاصل شده از تغییر مقدار سلول انتخاب شده بهتر از جواب جاری باشد سلول به همراه همسایه‌های چپ و راست آن پاداش می‌گیرند. در غیر اینصورت پاداش یا جریمه‌ای به آنها اختصاص داده نمی‌شود. علت پاداش دادن به همسایه‌های سلول انتخاب شده غیر خطی‌تر کردن رفتار الگوریتم می‌باشد. همچنین جریمه نکردن سلولی که تغییر آن جواب را بدتر کرده است به الگوریتم اجازه بررسی جواب‌هایی با مقدار جدید سلول را می‌دهد و به این ترتیب نقاط بیشتری از دامنه مساله بررسی می‌شوند. البته از آنجایی که به این سلول پاداشی داده نمی‌شود احتمال تغییر آنها از سلول‌هایی که تغییر آنها موجب بهبود جواب شده است بیشتر خواهد شد.

تشخیص توقف الگوریتم: در الگوریتم CLA-GEO برای تشخیص گرفتار شدن الگوریتم نیز از مقادیر احتمال تغییر سلول‌ها استفاده می‌شود؛ هرگاه میانگین احتمال تغییر سلول‌ها از آستانه‌ای عبور کرد و جواب جاری بهترین جواب اجرا نبود، فرض می‌شود که الگوریتم در نقطه‌ای متوقف شده است؛ در چنین شرایطی الگوریتم CLA-GEO به وضعیت شروع مجدد می‌رود.

از آنجایی که در هر تکرار حلقه اکثر سلول‌ها جریمه می‌شوند (همه سلول‌ها به غیر از سلول تعویضی و همسایه‌های آن)، میانگین تغییر احتمال همواره شکل صعودی دارد، به این ترتیب الگوریتم نمی‌تواند به صورت نامحدود در نقطه‌ای متوقف شود و دیر یا زود به وضعیت شروع مجدد می‌رود. آستانه میانگین در ابتدای هر شروع مجدد برابر پارامتر آستانه قرار می‌گیرد؛ برای آنکه به الگوریتم اجازه داده شود تا در مناطقی که

---

[1] Gain
[2] Threshold

احتمال وجود جواب بهینه می‌رود بررسی بیشتری انجام دهد، هرگاه جواب جاری بهترین جواب اجرا بود مقدار آستانه میانگین به اندازه پارامتر بهره زیاد می‌شود. این کار باعث می‌شود که میانگین احتمال تغییر سلول‌ها دیرتر آستانه میانگین را پشت سر گذاشته و در نتیجه دقت جواب بالاتر رود.

شروع مجدد: در صورتی که الگوریتم در نقطه‌ای متوقف شد - به این معنی که میانگین احتمال تغییر سلول بیشتر از آستانه میانگین باشد و جواب جاری بهترین جواب نباشد- سلول‌ها و متغیر آستانه میانگین مجددا مقداردهی شده و اجرا از نقطه‌ای جدید ادامه پیدا می‌کند. لازم به ذکر است که شروع مجدد الگوریتم به معنی پایان اجرای الگوریتم و شروع مجدد اجرای آن نیست. شروع مجدد الگوریتم در اینجا به این معنی است که پس از تشخیص گرفتار شدن الگوریتم، جواب جاری به جای تغییرات کوچک (تغییر یک جزء آن) جهش بزرگی پیدا کرده و به نقطه دورتری منتقل شود. بنابراین در هر اجرا ممکن است چندین شروع مجدد به وقوع به پیوندد. در مرحله شروع مجدد الگوریتم، به ترتیب عملیات زیر انجام می‌شود:

1. تولید جواب جدید از روی جواب جاری و مقادیر احتمال تغییر سلول‌ها
2. مقداردهی احتمال تغییر تمامی سلول‌ها با مقدار اولیه 0.5
3. مقداردهی آستانه میانگین با مقدار پارامتر آستانه

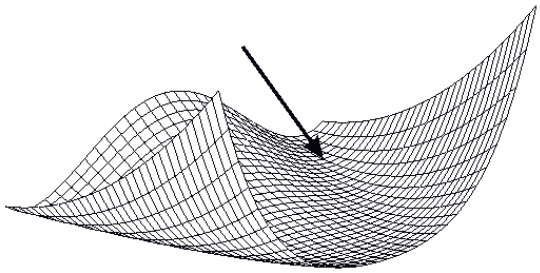
تولید جواب جدید به اینصورت انجام می‌شود که مقدار سلول‌ها با احتمالی برابر احتمال تغییر سلول بدون تغییر باقی‌مانده، در غیر اینصورت با یک مقدار بولی اتفاقی جایگزین می‌شود. به این ترتیب سلول‌هایی که دارای احتمال تغییر کمتری هستند در مرحله شروع مجدد با احتمال بیشتری تغییر پیدا می‌کنند. این نوع تغییر مقادیر سلول‌ها از این مطلب نتیجه‌گیری شده است که در صورت توقف الگوریتم در یک نقطه، سلول‌هایی که قبل از تشخیص توقف تغییری نداشته‌اند علت اصلی توقف و بهبود نیافتن جواب هستند.

شکل کلی الگوریتم: الگوریتم CLA-GEO از نظر کلیات شباهت بسیار زیادی به الگوریتم بهینه‌سازی حدی تعمیم یافته دارد. مراحل این الگوریتم به شرح زیر است:

1. جواب اولیه‌ای تولید شده و به عنوان جواب جاری و همچنین به عنوان بهترین جواب در نظر گرفته می‌شود.
2. فرزندان جواب جاری تولید شده و شایستگی آنها محاسبه می‌شود.
3. جواب‌های تولید شده (فرزندان جواب جاری) با توجه به شایستگی آنها به صورت صعودی مرتب می‌شوند.
4. سلول‌های متناظر با جواب‌های تولید شده به صورت گردشی عملی را انتخاب می‌کنند تا نهایتا یکی از آنها عمل تغییر مقدار را انتخاب کند.
5. تمامی سلول‌ها به غیر از سلول انتخاب شده و دو همسایه آن جریمه می‌شوند.
6. در صورتی که جواب انتخاب شده بهتر از جواب جاری باشد سلول انتخاب شده و دو همسایگی آن پاداش می‌گیرند.
7. جواب متناظر با سلول انتخاب شده به عنوان جواب جاری در نظر گرفته می‌شود.
8. در صورتی که جواب جاری بهتر از بهترین جواب بود، به عنوان بهترین جواب انتخاب و آستانه میانگین به اندازه پارامتر بهره زیاد می‌شود.
9. در صورتی که میانگین احتمال تغییر سلول‌ها بیشتر از آستانه میانگین بوده و جواب جاری بهترین جواب اجرا نباشد، الگوریتم شروع مجدد انجام می‌دهد و جواب تولید شده را به عنوان جواب جاری در نظر می‌گیرد.
10. مرحله 2 تا رسیدن به جواب مورد نیاز تکرار می‌شود.
11. بهترین جواب برگردانده می‌شود.

6. نتایج آزمایش‌ها

برای نشان دادن رفتار دو الگوریتم بهینه‌سازی حدی تعمیم یافته و CLA-GEO و مقایسه آنها، نحوه عملکرد آنها برای پیدا کردن نقطه بهینه تابع Rosenbrock's Valley مورد بررسی قرار می‌گیرد. شکل 2 تابع Rosenbrock's Valley و محل نقطه بهینه آن را نشان می‌دهد.



شکل (2): تابع Rosenbrock's Valley

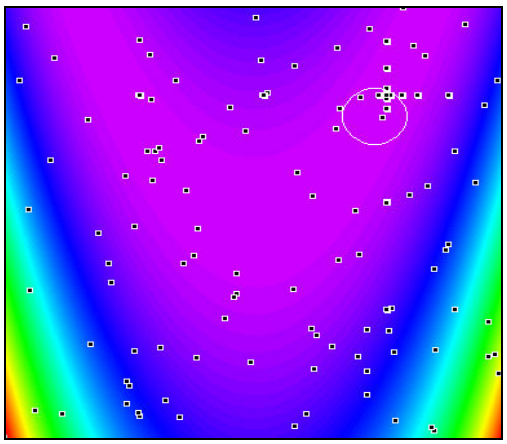
شکل‌های 3 و 4 نحوه جستجو و جواب‌های تولید شده توسط الگوریتم‌های بهینه‌سازی حدی تعمیم یافته و CLA-GEO برای پیدا کردن نقطه بهینه تابع Rosenbrock's Valley را در راستای محور z (عمود به دامنه) نشان می‌دهند. نقطه بهینه تابع توسط دایره‌ای در این اشکال نمایش داده شده است. همانطور که در شکل 3 مشاهده می‌شود الگوریتم بهینه‌سازی حدی تعمیم یافته با انتخاب بهترین جواب از بین جمعیت اولیه، بر روی آن تمرکز کرده و جواب‌های بعدی را در اطراف آن به منظور پیدا کردن بهترین جواب تولید می‌کند. بنابراین پیدا کردن جواب صحیح کاملا به جمعیت اولیه و پراکندگی آن وابسته است؛ در صورتی که پراکندگی این جمعیت مناسب نبوده و یا تغییرات تابع و نقاط بهینه محلی آن زیاد باشد، تمرکز الگوریتم می‌تواند بر نقطه‌ای نادرست و بسیار دورتر از محل اصلی جواب قرار بگیرد و در انتها جواب مناسب تولید نشود.

شکل 4 یک نمونه از اجرای الگوریتم CLA-GEO را با تعداد تکرارهایی برابر با اجرای الگوریتم بهینه‌سازی حدی تعمیم یافته (شکل 3) نشان می‌دهد. همانطور که مشاهده می‌شود، الگوریتم CLA-GEO شباهت بسیار زیادی به الگوریتم بهینه‌سازی حدی تعمیم یافته دارد. نحوه عملکرد این الگوریتم مانند مجموعه‌ای از چندین اجرای الگوریتم بهینه‌سازی حدی تعمیم یافته است. در واقع همانطور که در بخش‌های قبل اشاره شد، الگوریتم CLA-GEO با کم کردن تعداد جواب‌های متمرکز در یک نقطه و گسترش آن به نقاط دیگر، دامنه مساله را به شکل مناسب‌تری جستجو می‌کند. همانطور که شکل 4 نشان می‌دهد، این الگوریتم بر روی هر محلی که امکان حضور جواب وجود داشته باشد تمرکز کرده و بهترین جواب را در آن محل بدست می‌آورد.

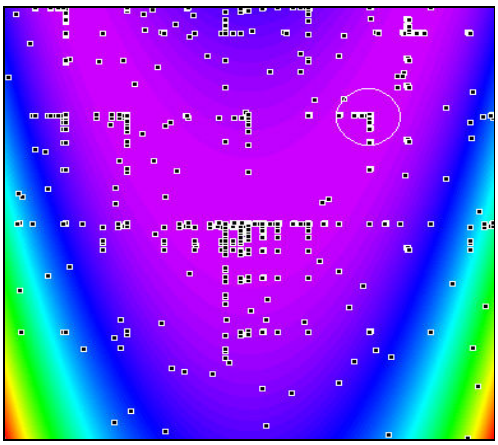
دو الگوریتم CAL-GEO و الگوریتم بهینه‌سازی حدی تعمیم یافته بر روی 5 تابع که فهرست آنها در جدول 1 آمده مورد آزمایش قرار گرفته‌اند که نتایج حاصل از آنها در ادامه ارائه می‌گردد. این توابع از نظر تعداد نقاط بهینه محلی و چگونگی توزیع آنها متفاوت هستند و برای بررسی الگوریتم‌های بهینه‌سازی به طور گسترده‌ای استفاده می‌شوند[6][7][8][9].

آزمایشهای برای تعداد تکرارهای1000، 10000 و 30000 که به ترتیب با نامهای تعداد تکرار کم، متوسط و زیاد به آنها رجوع میشود انجام شده است. هر کدام از الگوریتم‌ها 300 بار به ازای هر کدام از تعداد تکرارها بر روی هر یک از توابع آزمایشی اجرا شده و جواب‌های تولید شده به عنوان داده آماری برای مقایسه استفاده شده‌اند. برای این آزمایشها مقدار پارامتر τ الگوریتم بهینه‌سازی حدی تعمیم یافته برابر 1.25 قرار داده شده است. همچنین برای الگوریتم CLA-GEO پارامتر بهره برابر 0.03 و پارامتر آستانه برابر 0.2 در نظر گرفته شده‌اند. جمعیت اولیه هر دو الگوریتم برابر 100 انتخاب شده است.

نمودارهای توزیع جواب‌ها در نزدیکی جواب بهینه به ازای تعداد تکرار مشخص و هر کدام از توابع آزمایشی در شکل‌های 5 الی 9 آمده است. در این نمودارها محور افقی نشان دهنده مقدار جواب و محور عمودی نشان دهنده تعداد جواب‌ها است. جواب بهینه در این نمودارها در منتهی الیه سمت چپ نمودار قرار گرفته است. به عنوان مثال، شکل 6-الف نشان می‌دهد که الگوریتم بهینه‌سازی حدی تعمیم یافته برای تابع Rosenbrock's Vally به ازای تعداد تکرار کم، در 13 اجرا از 300 اجرا به جواب بهینه (مقدار صفر) رسیده است در حالی که الگوریتم CLA-GEO تنها 11 بار جواب صحیح را تولید کرده است. برای بالا بردن خوانایی نمودارها تنها 10 جواب نزدیک به نقطه بهینه نمایش داده شده و در هر نمودار دقت محور عمودی با توجه به بیشترین تعداد جواب‌ها تنظیم شده است. علاوه بر نمودارهای توزیع جواب، مقدار میانگین آماری و کمترین مقدار جواب‌ها به ازای تعداد تکرارها و توابع آزمایشی نیزدر جداول 2، 3 و 4 آمده است؛ نتایج آزمایشها نشان دهنده برتری الگوریتم CLA-GEO بر الگوریتم بهینه‌سازی حدی تعمیم یافته است.



شکل (3): نحوه حرکت الگوریتم بهینه‌سازی حدی تعمیم یافته برای تابع Rosenbrock's Valley با 10000 تکرار



شکل (4): نحوه حرکت الگوریتم CLA-GEO برای تابع Rosenbrock's Valley با 10000 تکرار

با بررسی نمودارها و جداول مشاهده می‌شود که الگوریتم CLA-GEO در توابعی که دارای چندین نقطه بهینه هستند بسیار بهتر از الگوریتم بهینه‌سازی حدی تعمیم یافته عمل می‌کند. این نتایج همچنین نشان می‌دهند که برای تابع Rosenbrock's Vally که تنها یک نقطه بهینه دارد ولی شیب تابع به شکل واضحی به سمت آن نیست نیز الگوریتم CLA-GEO جواب‌های بهتری در مقایسه با الگوریتم بهینه‌سازی حدی تعمیم یافته تولید می‌کند. تنها تابعی که الگوریتم بهینه‌سازی حدی تعمیم یافته نتایج مطلوبی را برای آن تولید می‌کند، تابع Sphere است که تنها یک نقطه بهینه داشته و شیب تابع کاملا به سمت این نقطه است؛ البته در این تابع نیز مشاهده می‌کنیم که برای تکرارهای کم این الگوریتم بدتر از الگوریتم CLA-GEO عمل می‌کند.

برای مشاهده نتایج الگوریتم پیشنهادی بر روی توابع آزمایشی دیگر می‌توان به [16] مراجعه نمود.

7. نتیجه گیری

در این مقاله یک مدل ترکیبی که از ترکیب بهینه‌سازی حدی تعمیم یافته و اتوماتای یادگیر سلولی حاصل شده برای حل مسایل بهینه سازی پیشنهاد گردید.. این الگوریتم ابر اکتشافی، مشکل بزرگ الگوریتم بهینه‌سازی حدی تعمیم یافته که گرفتار شدن در نقاط بهینه محلی است را نداشته، دامنه مساله را به شکل بهتری بررسی و جواب‌هایی با دقت بالا تولید می‌کند. نتایج آزمایش‌ها دلیلی بر این مدعا می‌باشد.

مراجع

[1] محمد شیبانی، "اتوماتای یادگیر سلولی، انواع و کاربردهای آن"، سمینار کارشناسی ارشد، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، 1384

[2] محمدرضا میبدی، حمید بیگی و مسعود طاهرخانی، "اتوماتای یادگیر سلولی"، در مجموعه مقالات ششمین کنفرانس انجمن کامپیوتر ایران، ص 163-153، 1379.

[3] Donald L Tucotte, "Self-Organized Criticality", Rep. Prog. Phys., Vol. 62, 1999, pp. 1377-1429

[4] Stefan Boettcher, Allon G. Percus, "Extremal Optimization: An Evolutionary Local-Search Algorithm", http://arxiv.org/abs/cs.NE/0209030

[5] Fabiano Luis de Sousa, Valeri Vlassov, Fernando Manuel Ramos, "Generalized Extremal Optimization for Solving Complex Optimal Design Problems", Lecture Notes in Computer Science, Vol. 2723, 2003, pp.375-276

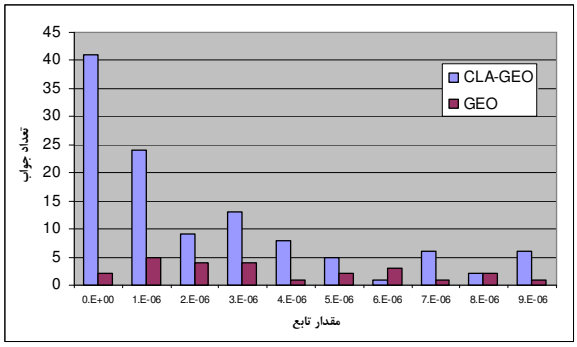
[6] Nasimul Noman and Hitoshi Iba, "Enhancing Differential Evolution Performance with Local Search for High Dimensional Function Optimization", GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation, 2005, pp. 967-974

[7] Jason G. Digalakis, Konstantinos G. Margaritis, "An Experimental Study of Benchmarking Functions for Genetic Algorithm", Intern. J. Computer Math, Vol. 79(4), 2002, pp. 403-416

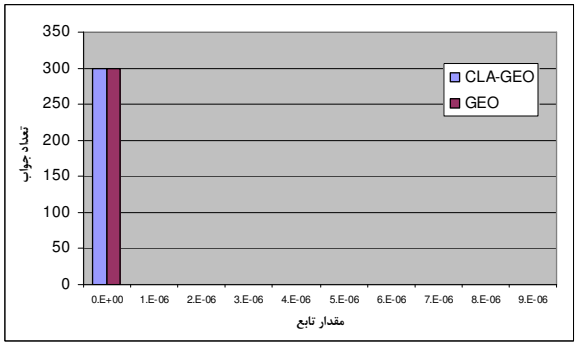
[8]http://www.geatbx.com/docu/fcnindex-01.html

[9]http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume24/ortizboyer05a-html/node6.html

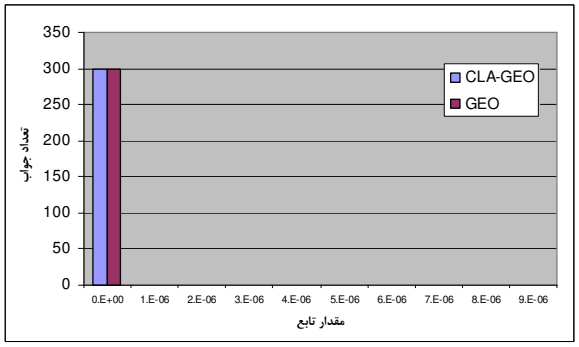
[9] S. Wolfram, "Cellular Automata", Los Alamos Science, vol. 9, pp. 2-21, Fall 1983.
[10] S. Wolfram, "Universality and Complexity in Cellular Automata", Physica D, no. 10, pp. 1-35, January 1984.
[11] Narendra K. S. and Thathachar M.A.L., "Learning Automata: An Introduction", Prentice Hall, 1989.
[12] Beigy, H. and Meybodi, M. R., "A Mathematical Framework for Cellular Learning Automata", Advances on Complex Systems, Vol. 7, Nos. 3-4, pp. 295-320, September/December 2004
[13] M. A. L. Thathachar and P. S. Sastry, "Varieties of Learning Automata: An Overview", IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics, Vol. 32, No. 6, PP. 711-722, 2002.
[14] Beigy, H. and Meybodi, M. R., "A Mathematical Framework for Cellular Learning Automata", Proceedings of 9th Annual CSI Computer Conference, Computer Engineering Department, Sharif University, Tehran, Iran, pp.151-159 , Feb. 2004
[15] Masoodifar, B., Meybodi, M. R. and Rastegar, R., "Asynchronous CLA-EC", Proceedings of 11th Annual CSI Computer Conference of Iran, Fundamental Science Research Center (IPM), Computer Science Research Lab., Tehran, Iran, pp. 447-458, Jan. 24-26, 2006
[16] Khatamnejad, A. and Meybodi, M. R., "More Experiments for CLA-GEO Algorithms, Technical Report, Computer Engineering Department, Amirkabir University of Technology, 2007.
[17] Beigy, H. and Meybodi, M. R., "Open Synchronous Cellular Learning Automata" Advances in Complex Systems, 2007, to appear .
[18] Beigy, H. and Meybodi, M. R., "Asynchronous Cellular Learning Automata" Automatica, Journal of International Federation of Automatic Control, 2007, to appear.



(الف)

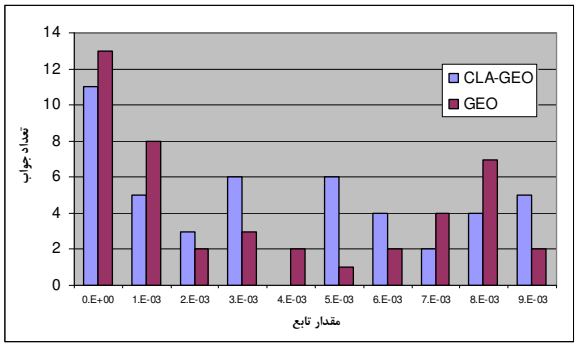


(ب)

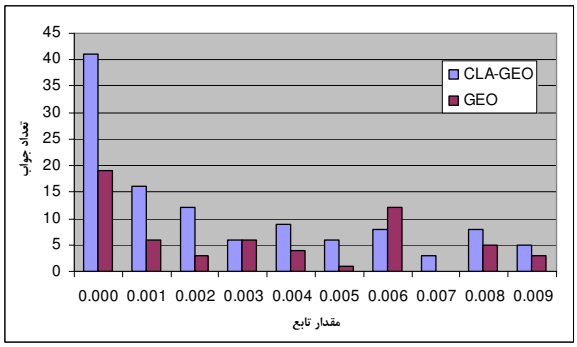


(ج)

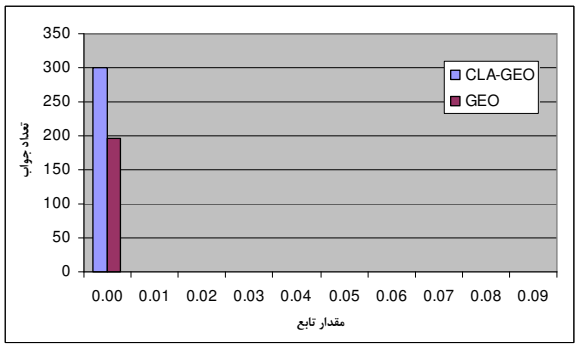
شکل (5): مقایسه الگوریتم‌ها برای تابع Sphere (De Jong's 1 برای 300 اجرا؛ شکل (الف) 1000 تکرار، (ب) 10000 تکرار و (ج) 30000 تکرار



(الف)

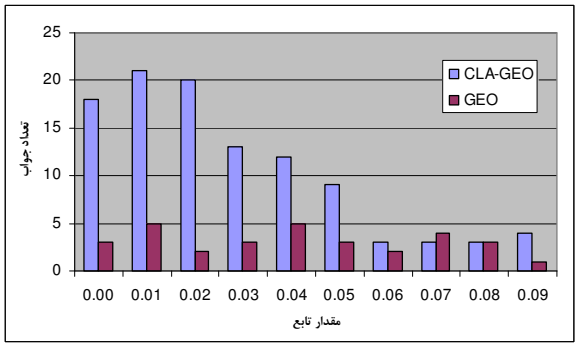


(ب)

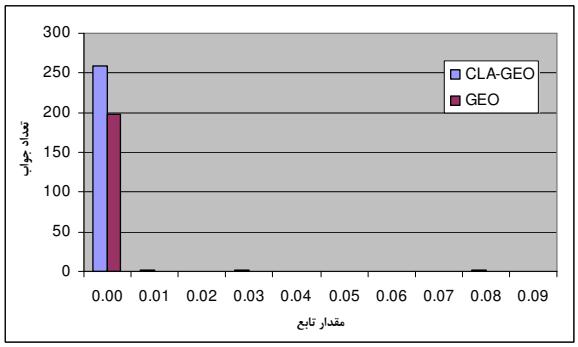


(ج)

شکل (6): مقایسه الگوریتم‌ها برای تابع Rosenbrock's Valley برای 300 اجرا؛ شکل (الف) 1000 تکرار، (ب) 10000 تکرار و (ج) 30000 تکرار



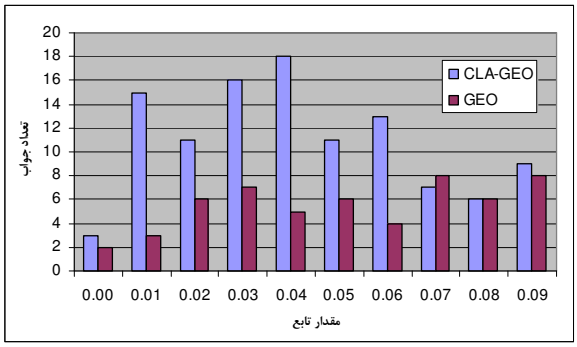
(الف)



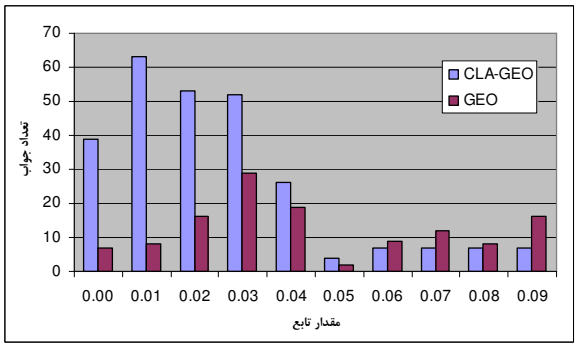
(ب)

(ج)

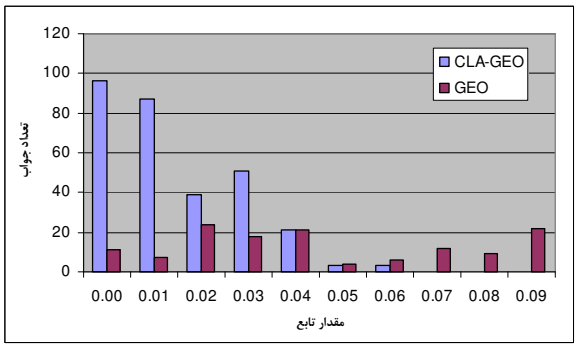
شکل (7): مقایسه الگوریتم‌ها برای تابع Ackley's Path برای 300 اجرا؛ شکل (الف) 1000 تکرار، (ب) 10000 تکرار و (ج) 30000 تکرار



(الف)

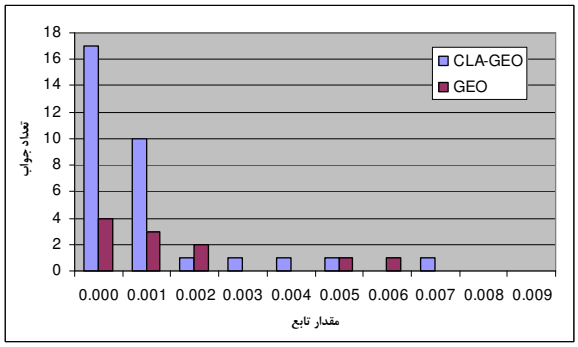


(ب)

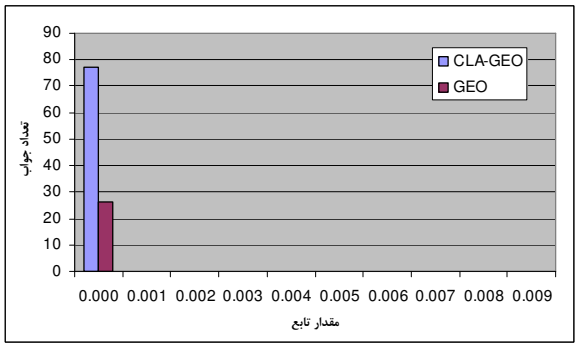


(ج)

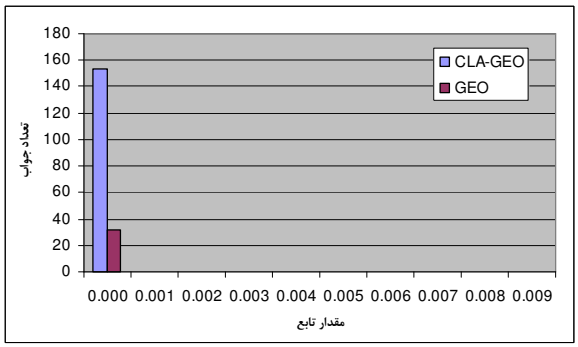
شکل (8): مقایسه الگوریتم‌ها برای تابع Griewank برای 300 اجرا؛ شکل (الف) 1000 تکرار، (ب) 10000 تکرار و (ج) 30000 تکرار



(الف)



(ب)



(ج)

شکل (9): مقایسه الگوریتم‌ها برای تابع Rastrigin برای 300 اجرا؛ شکل (الف) 1000 تکرار، (ب) 10000 تکرار و (ج) 30000 تکرار

جدول (1) : توابع تست استفاده شده

تابع	رابطه تابع	بازه مقایر ورودی	مقدار بهینه	چندین بهینه
Sphere	$f(x)=\sum_{i=1}^{n} x_i^2$	$-5.12 \le x_i \le 5.12$	0	خیر
Rosenbrock's valley	$f(x)=\sum_{i=1}^{n-1} 100\left(x_{i+1}-x_i^2\right)^2+\left(1-x_i\right)^2$	$-2.048 \le x_i \le 2.048$	0	خیر
Rastrigin's Function	$f(x)=10n+\sum_{i=1}^{n}\left(x_i^2-10\cos\left(2\pi x_i\right)\right)$	$-5.12 \le x_i \le 5.12$	0	بله
Griewangk's Function	$f(x)=\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\left(\cos\left(\frac{x_i}{\sqrt{i}}\right)+1\right)$	$-600 \le x_i \le 600$	0	بله
Ackley's Path Function	$f(x)=-ae^{-b\sqrt{\sum_{i=1}^{n} x_i^2}}-e^{\frac{\sum_{i=1}^{n}\cos(cx_i)}{n}}+a+e^1$ $a=20, b=0.2, c=2.\pi$	$-32.768 \le x_i \le 32.768$	0	بله

جدول (2) : مقایسه میانگین و مقدار کمینه الگوریتم‌ها برای 1000 تکرار و 300 اجرا

تابع	GEO		CLA-GEO	
	Minimum	Average	Minimum	Average
Ackley's Path Function	0.001154875	1.925694786	0.000366326	1.21319103
De Jong F1 (Sphere)	5.6721E-08	0.005300753	2.79701E-11	0.000136543
De Jong F2 (Rosenbrock's valley)	6.93181E-06	0.293826812	1.81338E-05	0.273187986
Griewangk's Function	5.22586E-06	0.413742526	0.00042237768	0.2859457811
Rastrigin's Function	0.000116434	2.960419066	6.28324E-08	2.738726014

جدول (3) : مقایسه میانگین و مقدار کمینه الگوریتم‌ها برای 10000 تکرار و 300 اجرا

تابع	GEO		CLA-GEO	
	Minimum	Average	Minimum	Average
Ackley's Path Function	4.4E-16	0.914787754	4.4E-16	0.322155958
De Jong F1 (Sphere)	0	0	0	4.099E-14
De Jong F2 (Rosenbrock's valley)	5.17704E-06	0.247004753	0	0.107824478
Griewangk's Function	0	0.265740761	0	0.0506267880
Rastrigin's Function	0	2.740808937	0	1.335447435

جدول (4) : مقایسه میانگین و مقدار کمینه الگوریتم‌ها برای 30000 تکرار و 300 اجرا

تابع	GEO		CLA-GEO	
	Minimum	Average	Minimum	Average
Ackley's Path Function	4.4E-16	0.977413772	4.4E-16	3.77136E-08
De Jong F1 (Sphere)	0	0	0	0
De Jong F2 (Rosenbrock's valley)	1.60253E-05	0.256432419	0	0.016283972
Griewangk's Function	0	0.2676790143	0	0.0140889841
Rastrigin's Function	0	2.515645606	0	0.549908912