Journal of Iranian Association of Electrical and Electronics Engineers
Vol. 22 No. 2, 2025
DOI: 10.61186/jiaeee.22.2.111

An intelligent Gradient Algorithm Using Particle Swarm Optimization

Zahra Ghavami[1], Mohammad Mollaie Emamzadeh[2] , Maliheh Maghfoori Farsangi[3]

[1] MSc, Department of Electrical Engineering, Shahid Bahonar University of Kerman, Kerman, Iran
zahra.ghavami@eng.uk.ac.ir
[2] Assistant Professor, Department of Electrical Engineering, Shahid Bahonar University of Kerman, Kerman, Iran
molaie@uk.ac.ir
[3] Professor, Department of Electrical Engineering, Shahid Bahonar University of Kerman, Kerman, Iran
mmaghfoori@uk.ac.ir

Abstract:
The Gradient algorithm is the simplest and most widely used method in optimization problems and machine learning. The convergence rate of this method strongly depends on choosing the suitable value for the step size. Choosing a very small value can cause a low convergence rate, on the other hand, choosing a very large value may also cause divergence and oscillation around the optimal point. Usually, the step size is chosen larger in the initial steps of optimization and as the execution steps progress and the optimal solution is approached, its value decreases. The optimal setting of this hyper-parameter is experimentally and by trial and error and has to be done separately for every problem, so it takes a lot of time. On the other hand, in swarm intelligence optimization methods, including the particle swarm optimization (PSO) algorithm, the step size (the length of movement) is automatically adjusted during the execution of the optimization method. Also, in these methods, few parameters need to be tuned and a predetermined range is available for this purpose. In this paper, by combining the PSO and Gradient method, an intelligent optimization method based on the gradient is presented, which does not need to adjust the step size. The performance of the proposed gradient algorithm is evaluated on ten benchmark functions and it is observed that the proposed algorithm has a better convergence rate than the Classic Gradient algorithm in reaching the optimal solution.

Keywords: Gradient algorithm, Particle swarm optimization (PSO), Step size, Optimum directional search, Intelligentization

Article Type: Research

Received: 16. 07. 2023

Revised: 24. 06. 2024

Accepted: 11. 08. 2024

Corresponding author: M. Mollaie Emamzadeh

Corresponding author's address: Imam Khomeini Highway, Pazhoohesh Square, Electrical Engineering Department, Shahid Bahonar University of Kerman, Kerman, Iran

1. Motivation of the work

In the Gradient Descent method, determining the step-size is a very important issue that must be established separately for each optimization problem through trial and error, which is time-consuming and problem-dependent because the step-size can vary by hundreds or even thousands of times from one problem to another. Therefore, its experimental selection is difficult and time-consuming, and the efficiency and performance of the Gradient algorithm highly depend on the appropriate selection of the step-size. If the step-size is too large, the Gradient method may oscillate and lead to divergence from the optimal solution. Conversely, if the step-size is too small, the optimization process may converge very slowly. On the other hand, in swarm intelligence methods such as PSO, this process of parameter tuning does not exist. Drawing inspiration from PSO method, this paper presents An intelligent Gradient method that addresses the step-size adjustment issue.

2. Contributions

In the proposed method, the aim is to present an intelligent Gradient algorithm by utilizing the normalized gradient and drawing inspiration from the dynamics of the PSO algorithm. This approach not only determines the movement size intelligently at each step but also enhances the convergence rate in reaching the optimal solution compared to the Gradient method. As a result, the proposed method achieves a better solution in fewer steps than the Gradient method. Additionally, in the proposed method, all parameters have predefined ranges, making their selection and adjustment easier for different optimization problems.

3. Procedures

In this paper, the goal is to present an intelligent Gradient algorithm inspired by the PSO algorithm, such that the challenge of selecting the step-size is solved, and the size of movement changes automatically similar to the PSO algorithm. The proposed gradient algorithm is applied to ten benchmark functions, and the results are compared with those from the classic Gradient algorithm. It is worth mentioning that all simulations in this paper were done in the MATLAB Software on a computer with 6 GB of RAM and a Core(TM) i5-4210 processor, running Windows 10 (64-bit). Additionally, all results are reported after thirty independent runs and then the average, the worst and the best results are reported. The simulation results indicated that the proposed algorithm outperformed in all benchmark functions.

4. Findings

In this paper, by examining the PSO method, an intelligent dynamic applicable to one particle is presented. By combining this dynamic with the Gradient method, an intelligent gradient-based approach is proposed that addresses the main issue of the gradient method in selecting an appropriate step-size and enhances the convergence rate of the method. This innovation has improved the efficiency of the proposed method, making its use straightforward. Additionally, the issue of determining the step-size in the Gradient method has been solved in the proposed approach. The proposed algorithm has demonstrated better performance across all benchmark functions compared to the classic Gradient algorithm and has increased the convergence rate relative to the classical gradient algorithm.

5. Conclusion

In this paper, inspired by the advantages of the Particle Swarm Optimization (PSO) algorithm, such as a limited number of hyper-parameters and the existence of a standard range for selecting these hyper-parameters, as well as considering the advantage of low computational time in the Gradient algorithm, a smart Gradient algorithm is presented. In this algorithm, the step size and direction of movement are automatically adjusted, eliminating the need for selecting the step size hyper-parameter. To evaluate the efficiency and performance of the proposed method, ten benchmark functions were optimized, and the results obtained from the proposed method were compared with those from the classic Gradient algorithm. The results of the simulations showed that for all benchmark functions, the proposed algorithm performed better in terms of both convergence rate and achieving optimal solutions.

نشریه مهندسی برق و الکترونیک ایران

نشریه مهندسی برق و الکترونیک ایران، سال بیست و دوم، شماره دوم، تابستان ۱۴۰۴، صفحات ۱۱۱-۱۲۶

DOI: 10.61186/jiaeee.22.2.111

هوشمندسازی الگوریتم گرادیان با الهام از روش بهینه‌سازی ازدحام ذرات

زهرا قوامی[۱]، محمد ملایی امامزاده[۲]، ملیحه مغفوری فرسنگی[۳]

۱- کارشناسی ارشد - مهندسی برق - دانشکده فنی و مهندسی- دانشگاه شهید باهنر کرمان- کرمان- ایران

zahra.ghavami@eng.uk.ac.ir

۲- استادیار - مهندسی برق - دانشکده فنی و مهندسی- دانشگاه شهید باهنر کرمان- کرمان- ایران

molaie@uk.ac.ir

۳- استاد - مهندسی برق - دانشکده فنی و مهندسی- دانشگاه شهید باهنر کرمان- کرمان- ایران

mmaghfoori@uk.ac.ir

چکیده: الگوریتم گرادیان ساده‌ترین و پرکاربردترین روش در بهینه‌سازی و یادگیری ماشین می‌باشد. سرعت همگرایی این روش به‌شدت به انتخاب مقدار مناسب برای طول‌گام بستگی دارد. انتخاب طول‌گام بسیارکوچک می‌تواند باعث سرعت همگرایی کند شود. ازطرفی انتخاب طول‌گام بسیاربزرگ نیز ممکن است باعث واگرایی و نوسان حول نقطه بهینه گردد. معمولاً طول‌گام را در مراحل اولیه بهینه‌سازی بزرگ‌تر انتخاب کرده و با پیش‌رفتن گام‌های اجرا و نزدیکی به جواب بهینه، مقدار آن کاهش می‌یابد که تنظیم بهینه مقدار این پارامتر به‌صورت تجربی و با سعی‌وخطا برای هر مسئله‌ای باید انجام شود و زمان زیادی را می‌طلبد. ازطرفی در روش‌های بهینه‌سازی مبتنی‌بر هوش جمعی، از جمله در الگوریتم بهینه‌سازی ازدحام ذرات (PSO) ، طول‌گامِ حرکت به‌صورت خودکار و درحین اجرای روش تنظیم می‌شود. همچنین در این روش‌ها، پارامترهای اندکی نیاز به تنظیم دارند و محدوده ازپیش-تعیین شده‌ای برای این منظور موجود است. در این مقاله با ترکیب دو روش PSO و گرادیان، یک روش بهینه‌سازی هوشمند مبتنی بر گرادیان ارائه شده است که نیازی به تنظیم طول‌گام ندارد. عملکرد الگوریتم گرادیان پیشنهادی برروی ده تابع محک مورد بررسی قرارگرفته و مشاهده می‌شود که الگوریتم پیشنهادی سرعت همگرایی بهتری نسبت به الگوریتم گرادیان کلاسیک در رسیدن به جواب بهینه دارد.

کلمات کلیدی: الگوریتم گرادیان نزولی-الگوریتم بهینه‌سازی ازدحام ذرات (PSO)- طول‌گام- جهت حرکت بهینه-هوشمندسازی

نوع مقاله: پژوهشی

دریافت: ۱۴۰۲/۰۴/۲۵

بازنگری: ۱۴۰۳/۰۴/۰۴

پذیرش: ۱۴۰۳/۰۵/۲۱

نام نویسنده‌ی مسئول: آقای دکتر محمد ملایی امامزاده

نشانی نویسنده‌ی مسئول: ایران – کرمان – بزرگراه امام خمینی – میدان پژوهش – دانشگاه شهید باهنر کرمان – بخش مهندسی برق

۱- مقدمه

در دهه‌های اخیر مصرف انرژی به سرعت افزایش یافته به‌نحوی‌که با افزایش مصرف جهانی، ذخایر سوخت فسیلی رو به کاهش است و مهم‌تر از آن، این امر منجر به افزایش دمای کره زمین و مشکلات اقلیمی شده است. این مسائل، چالشی جدید ایجاد کرده که سبب شده تا توجه جهانیان در مرحله اول به سمت استفاده بهینه از این منابع فسیلی و در مرحله دوم به سمت پیدا کردن جایگزینی برای آن‌ها از جمله منابع انرژی تجدیدپذیر معطوف شود. این توجه و تمایل به استفاده بهینه از منابع انرژی (هم فسیلی و هم تجدیدپذیر) و همچنین تلاش برای حل چالش‌های جدید در این زمینه‌ها، سبب مطرح شدن مسائل بهینه‌سازی زیادی گردیده است. بنابراین محققان در سراسر جهان به جستجو، بررسی و همچنین ارائه روش‌های بهینه‌سازی مختلف برای حل این مسائل پرداخته‌اند [۱].

در یک مسئله بهینه‌سازی، هدف اصلی کمینه کردن موارد نامطلوب (تابع هزینه[۱]) یا بیشینه کردن موارد مطلوب (تابع سود[۲]) تحت برخی محدودیت‌ها می‌باشد. در این مقاله هدف از بهینه‌سازی، کمینه کردن توابع هزینه درنظر گرفته شده است. روش‌های بهینه‌سازی به دو دسته اصلی تقسیم می‌شوند: ۱) بهینه‌سازی مبتنی بر مشتق ۲) بهینه‌سازی بی‌نیاز از مشتق. در روش‌های دسته اول، جهت جستجو با توجه به اطلاعات به‌دست آمده از گرادیان تابع هزینه مشخص می‌شود. درحالی‌که در روش‌های دسته دوم، برای جستجوی جواب بهینه در یک مسئله، نیازی به مشتق تابع هزینه وجود ندارد. در این روش‌ها به ارزیابی تابع هزینه به ازای نقاطی از دامنه جستجوی فضای مسئله پرداخته می‌شود و سپس با استفاده از نتایج به‌دست آمده و با درنظرگرفتن قواعدی ابتکاری و شهودی، جهت حرکت به سمت جواب بهینه در تکرارهای متوالی تعیین می‌شود [۳,۲].

جهان برای مدت طولانی شاهد استفاده گسترده از سیستم‌های هوش مصنوعی بوده است که این امر مستلزم استفاده و به‌کار گرفتن روش‌های بهینه‌سازی مناسب در یادگیری ماشین می‌باشد و به نوبه خود نیازمند محاسبات بالا است. الگوریتم گرادیان ساده‌ترین و متداول‌ترین روش از بین روش‌های بهینه‌سازی دسته اول می‌باشد. از مهم‌ترین مزایای این الگوریتم می‌توان به زمان محاسباتی پایین و کارایی بالا در مسائل با ابعاد بالا و حجم داده زیاد اشاره کرد. زمانی که حجم داده در مسائل بهینه‌سازی زیاد باشد، استفاده از الگوریتم گرادیان سبب سادگی پیاده‌سازی، عدم نیاز به حافظه خیلی بالا، تضمین رسیدن به جواب بهینه و جلوگیری از گیر افتادن در نقاط زینی[۳] شده که باعث جلوگیری از افزایش هزینه و پیچیدگی محاسباتی می‌شود. در نتیجه در مسائل با ابعاد بالا و حجم داده زیاد، روش گرادیان یکی از کارآمدترین گزینه‌ها به شمار می‌رود. الگوریتم گرادیان با استفاده از یک رویکرد تکراری روشی را ارائه می‌دهد که به تدریج مقادیر پارامترها را به منظور بهینه کردن تابع هزینه تصحیح می‌کند. این الگوریتم مقرون به صرفه بوده و در آموزش مدل‌های پیچیده و غیرمحدب شبکه‌های عصبی مصنوعی (با ارائه راه‌حل‌های مناسب برای کمینه کردن خطاهای آموزشی) موفقیت زیادی از خود نشان داده است و یکی از محبوب‌ترین الگوریتم‌های بهینه‌سازی مورد استفاده در یادگیری ماشین نیز به‌شمار می‌رود [۴-۷].

یکی از کلیدی‌ترین فراپارامترهایی[۴] که عملکرد الگوریتم گرادیان را تحت تاثیر قرار می‌دهد، طول‌گام[۵] حرکت است. منظور از فراپارامتر، پارامترهایی از مسئله می‌باشد که مقدار آن‌ها در ابتدا و قبل از اجرا باید مشخص شود. طول‌گام در واقع اندازه بروزرسانی موقعیت (متغیرهای مسئله) را در هر گام مشخص می‌کند. انتخاب طول‌گام، از الزامات اجرا و پیاده‌سازی الگوریتم گرادیان می‌باشد. در واقع می‌توان گفت که کارایی و عملکرد الگوریتم گرادیان به‌شدت به انتخاب مقدار مناسب برای طول‌گام بستگی دارد. لذا انتخاب مقدار مناسب برای طول‌گام یک مسئله مهم و حساس است. درصورتی‌که طول‌گام بسیاربزرگ باشد، ممکن است فرایند بهینه‌سازی نوسانی شده و باعث دور شدن از جواب بهینه و واگرایی گردد. از طرف دیگر اگر طول‌گام بسیارکوچک باشد، ممکن است فرایند بهینه‌سازی با سرعت همگرایی بسیارپایینی انجام شود. بنابراین انتخاب طول‌گام مناسب برای موفقیت الگوریتم گرادیان، یک مسئله ضروری است [۹,۸].

در طبیعت، موجودات زنده از طریق فرآیند تکامل، رفتارها و شیوه‌های زندگی منحصربه‌فردی ایجاد کرده‌اند. محققان از این پدیده‌ها الهام گرفته و ایده‌های جدید متعددی را جهت حل و بررسی مسائل بهینه‌سازی عملی ارائه داده‌اند. در سال‌های اخیر الگوریتم‌های بهینه‌سازی هوشمند از جمله الگوریتم‌های هوش جمعی به روشی مهم در حل مسائل بهینه‌سازی تبدیل شده‌اند. الگوریتم‌های هوش جمعی جزو روش‌های بهینه‌سازی دسته دوم هستند که براساس حافظه جمعی و جستجوی تصادفیِ مبتنی‌بر احتمال می‌باشند و از رفتارهای گروهی و دسته‌جمعی موجودات الهام گرفته شده‌اند [۱۰]. به عنوان مثال، الگوریتم بهینه‌سازی کلونی مورچگان[۶] با الهام از رفتار جستجوی غذا در مورچه‌ها در [۱۱]، الگوریتم کلونی زنبورهای عسل مصنوعی[۷] با الهام از رفتار جستجوی منابع غذایی در زنبورهای عسل در [۱۲]، الگوریتم بهینه‌سازی ازدحام ذرات[۸] با الهام از رفتار دسته جمعی پرندگان در طبیعت در [۱۳]، الگوریتم جستجوی کلاغ[۹] با الهام از رفتار هوشمندانه کلاغ‌ها در پنهان کردن غذای اضافی در مخفی‌گاه‌ها در [۱۴]، الگوریتم بهینه‌سازی گرگ‌های خاکستری[۱۰] با الهام از نحوه شکار دسته جمعی در گرگ‌های خاکستری در [۱۵]، الگوریتم جستجوی گرانشی[۱۱] با الهام از قانون گرانش نیوتن و قانون حرکت در [۱۶]، نمونه‌هایی از الگوریتم‌های هوش جمعی هستند که از رفتارهای گروهی و پدیده‌های طبیعی الهام گرفته شده‌اند. علاوه بر این، الگوریتم ژنتیک[۱۲]، الگوریتم ذوب شبیه‌سازی شده[۱۳]، الگوریتم کسینوس سینوسی[۱۴]، الگوریتم ایمنی[۱۵]، الگوریتم جستجوی ممنوعه[۱۶] و الگوریتم جستجوی همسایگی متغیر[۱۷] نیز انواع دیگری از الگوریتم‌های بهینه‌سازی هوشمند می‌باشند [۱۷].

PSO یکی از پرکابردترین و متداول‌ترین روش‌های هوش جمعی است که توسط کندی و ابرهارت در سال ۱۹۹۵ ارائه شد و به رفتار جستجوی غذا در پرندگان اشاره دارد. در این الگوریتم موقعیت هر ذره در فضای جستجو نشان دهندهِ یک جواب برای مسئله بهینه‌سازی است. هر ذره موقعیت خود را بر اساس تجربه خود و کل جمعیت بروزرسانی می‌کند. یکی از مهم‌ترین مزیت‌های این الگوریتم این است که تعداد پارامترهای انتخابی محدودی در مدل خود دارد و بازه استانداردی جهت تنظیم هریک از این پارامترها موجود است. از سایر مزایای الگوریتم PSO می‌توان به سادگی، پیاده‌سازی آسان و کارایی آن برای طیف گسترده‌ای از مسائل بهینه‌سازی اشاره کرد که باعث محبوبیت این الگوریتم در جامعه علمی شده و توجه بسیاری از محققان در حوزه‌های مختلف را به خود جلب کرده است. اما مواردی از جمله همگرایی زودرس و دقت نامناسب به عنوان ضعف‌های این الگوریتم به‌شمار می‌روند [۱۸-۲۱].

در این مقاله با ترکیب دو روش PSO و گرادیان، یک روش بهینه‌سازی هوشمند مبتنی بر گرادیان ارائه شده است که از یک طرف مزایای روش PSO در مسئله انتخاب فراپارامترها (این مسئله از معایب روش گرادیان است) و از طرف دیگر برتری روش گرادیان که همواره در جهت بهینه و کاهش تابع هزینه حرکت می‌کند، را در برداشته باشد.

روش گرادیان هوشمند ارائه شده در این مقاله، مبتنی بر گرادیان بوده و حتماً به مشتق تابع هزینه نیاز دارد. بنابراین فقط در مسائلی که مشتق تابع هزینه در دسترس است، مانند یادگیری شبکه‌های عصبی، می‌تواند مورد استفاده قرار گیرد. با توجه به اینکه الگوریتم گرادیان هوشمند از روش PSO الهام گرفته شده، بنابراین مسئله انتخاب مقدار بهینه برای طول‌گام حل شده است و به جای انتخاب این فراپارامترِ وابسته به مسئله که برای هر مسئله باید جداگانه انتخاب شود، چند پارامتر با مقادیر نسبتاً استاندارد مشابه روش PSO که از قبل بازه مشخصی جهت تنظیم آن‌ها برای همه مسائل وجود دارد، در نظر گرفته شده است.

این مقاله به صورت زیر سازماندهی شده است : در بخش دوم به بیان روش بهینه‌سازی گرادیان و در بخش سوم به بررسی الگوریتم PSO از دسته روش‌های بهینه‌سازی هوش جمعی پرداخته خواهد شد. در بخش چهارم با الهام گرفتن از الگوریتم PSO، روش پیشنهادی جهت هوشمندسازی الگوریتم گرادیان ارائه می‌شود. در بخش پنجم، نتایج شبیه‌سازی روش گرادیان هوشمند پیشنهادی به منظور بهینه‌سازی ده تابع محک آورده شده و نتایج حاصله بررسی شده است. به منظور ارزیابی بیشتر عملکرد روش گرادیان پیشنهادی، نتایج حاصله از آن با نتایج به‌دست آمده از روش گرادیان کلاسیک مورد بررسی و مقایسه قرار گرفته است. در بخش ششم نیز به جمع‌بندی مقاله و نتیجه‌گیری پرداخته می‌شود.

۲- روش بهینه‌سازی گرادیان

حل مسائل بهینه‌سازی در مقیاس بزرگ معمولاً به الگوریتم بهینه‌سازی مناسب برای پردازش مقادیر زیادی داده نیاز دارد. درغیر این‌صورت ممکن است حل مسئله با چالش‌هایی همچون زمان محاسباتی طولانی و نیاز به حافظه زیاد مواجه شده و منجر به هزینه محاسباتی زیادی شود. به همین دلیل باید از الگوریتم‌های مناسب، ساده و مقرون به‌صرفه استفاده کرد که از برجسته‌ترین نمونه این الگوریتم‌ها، الگوریتم گرادیان است [۲۲].

در مسائل بهینه‌سازی الگوریتم گرادیان در دو حالت افزایشی و کاهشی مورد استفاده قرار می‌گیرد که در حالت افزایشی، هدف بیشینه کردن تابع سود (با حرکت در جهت گرادیان تابع سود) و در حالت کاهشی، هدف کمینه کردن تابع هزینه (با حرکت درخلاف جهت گرادیان تابع هزینه) می‌باشد. الگوریتم گرادیان توسط کوشی در سال ۱۸۴۷ معرفی شد و یکی از اصولی‌ترین روش‌ها برای کمینه کردن توابع هزینه چندمتغیره و مشتق‌پذیر به‌شمار می‌رود [۲۳]. متداول‌ترین روش برای بهینه کردن تابع هزینه $J(x)$ با فرض مشتق‌پذیر بودن، روش گرادیان است که در بسیاری از مسائل مورد استفاده قرار می‌گیرد. الگوریتم‌های یادگیری ماشین نیز از الگوریتم گرادیان برای یافتن راه‌حل مناسب مسائل بهنیه‌سازی پیچیده و پرهزینه استفاده می کنند [۲۴]. در این مقاله نیز الگوریتم گرادیان کاهشی مورد بررسی قرار خواهد گرفت.

الگوریتم گرادیان، روشی مبتنی بر تکرار است که برای کمینه کردن تابع هزینه $J(x)$، پارامترهای $x \in \mathbb{R}^d$ را در خلاف جهت گرادیان تابع هزینه بروزرسانی می‌کند. موقعیت در گام k، مطابق با رابطه (۱) بروزرسانی می‌شود.

$$\begin{cases} X(k+1) = X(k) - \eta g_i \\ g_i = \nabla_x J(x) \end{cases} \quad (۱)$$

که در رابطه فوق، k شماره گام حرکت فعلی، $k+1$ شماره گام حرکت بعدی، X موقعیت، $J(X)$ تابع هزینه، η طول‌گام و $\nabla_X J(X)$ گرادیان تابع هزینه نسبت به موقعیت می‌باشند.

در روش گرادیان انتظار می‌رود که حرکت در جهت منفی گرادیان تابع هزینه در هر گام، باعث نزدیک‌تر شدن به نقطه کمینه اصلی شود اما برای مسائل واقعی، سطوح خطا معمولاً پیچیده بوده و ممکن است کمینه‌های محلی متعددی وجود داشته باشد [۲۵]. در الگوریتم گرادیان کاهشی، جهت گرادیان تقریباً عمود بر جهت نقطه کمینه[۱۸] است و در گام‌های بعد حول نقطه کمینه نوسان کرده و سرعت همگرایی کند خواهد بود. الگوریتم گرادیان با فراپارامتر تکانه[۱۹] می‌تواند سرعت همگرایی را در گام‌های اولیه به طور چشمگیری افزایش دهد. اما در گام‌های بعدی بهینه‌سازی ممکن است باعث نوسان حول جواب بهینه و حتی دور شدن از آن شود [۲۶].

طول‌گام که در رابطه (۱) با η نمایش داده شده، یک فراپارامتر است و اندازه حرکت و جابجایی در هر گام را به منظور رسیدن به نقطه کمینه تعیین می‌کند. از الزامات اجرا و پیاده‌سازی الگوریتم گرادیان، انتخاب

طول‌گام می‌باشد. در واقع می‌توان گفت که کارایی و عملکرد الگوریتم گرادیان به‌شدت به انتخاب مقدار مناسب برای طول‌گام بستگی دارد. انتخاب مقدار مناسب برای طول‌گام یک مسئله مهم و حساس می‌باشد زیرا طول‌گام بسیارکوچک می‌تواند منجر به سرعت همگرایی پایین شود، درحالی‌که طول‌گام بسیاربزرگ ممکن است باعث ایجاد نوسان حول نقطه بهینه و مانع همگرایی گردد. معمولاً انتخاب طول‌گام مناسب به‌صورت تجربی و با سعی و خطا انجام می‌شود که نیازمند درک شهودی از مسئله می‌باشد [۲۷].

تلاش‌های زیادی جهت تعیین و تخمین طول‌گام مناسب در هر گام الگوریتم گرادیان صورت گرفته است. عملکرد این روش‌ها بدین‌صورت است که در زمان مناسب و دور از نقاط کمینه، طول‌گامِ حرکت (سرعت یادگیری) را افزایش داده و در نزدیکی یک کمینه محلی، این طول‌گامِ حرکت را کاهش می‌دهند [۲۸]. بنابراین اندازه دقیق و خودکار تخمین زده شده طول‌گام مستقل از تنظیمات مسئله، در روش‌های بهینه‌سازی مبتنی بر گرادیان ضروری است. روش‌هایی که هدف‌شان حل مسئله تخمین طول‌گام است را می‌توان در سه گروه روش‌های دستی، نیمه‌خودکار و خودکار دسته بندی کرد.

روش رابینز-مونرو[۲۰]، قانون کِستن[۲۱] و قانون گاورونسکی[۲۲] از انتخاب دستی طول‌گام بهره می‌برند. استفاده از روش‌های انتخاب دستی در عمل دشوار است، زیرا مسئله‌های کاربردی مختلف به تنظیمات متفاوتی نیاز دارند و همچنین طول‌گام بین توابع هزینه با مرتبه‌های مختلف، می‌تواند متفاوت باشد. علاوه بر این، انتخاب دستی زمان‌بر است. یک نقطه ضعف استفاده از طول‌گام ثابت این است که معمولاً برای تضمین عملکرد الگوریتم، به اطلاعات اضافی مانند آگاهی از ثابت لیپ‌شیتس[۲۳] گرادیان، نیاز دارد. جستجوی خطی نیز به‌عنوان یک روش جهت پیاده‌سازی طول‌گام‌های متغیر در الگوریتم‌های بهینه‌سازی گرادیان شناخته می‌شود. در این روش، طول‌گام در هر تکرار با توجه به شرایط از پیش تعیین شده، به گونه‌ای انتخاب می‌شود که تابع هزینه را در جهت جستجو کمینه کند. این روش منجر به همگرایی سریع‌تر و جواب مناسب‌تری نسبت به طول‌گام ثابت می‌شود اما روشی طولانی بوده و به دلیل انجام یک جستجو در هر گام که مستلزم هزینه محاسباتی نمایی می‌باشد، روشی پرهزینه به شمار می‌رود و برای مجموعه داده‌های بسیاربزرگ به طور عملی غیرقابل استفاده است. روش نیمه‌خودکار نیز به دلیل نیاز به داده آموزشی و تعمیم نتایج آموزشی به موارد جدید، پیچیده و وقت‌گیر است [۳۰,۲۹].

در برخی روش‌ها از تنظیم طول‌گام به مقدار اولیه و کاهش تدریجی آن به‌صورت خودکار در حین آموزش استفاده می‌شود که روش تطبیقی تعیین طول‌گام نام دارد. این روش‌ها در بعضی از مراحل به تنظیم دستی برخی پارامترها نیاز دارند. علاوه بر آن، تعیین طول‌گام به‌صورت تطبیقی در مراحل اولیه سرعت بیشتری داشته اما توانایی تعمیم یادگیری ضعیفی دارد [۳۱-۳۳].

۳- الگوریتم بهینه‌سازی ازدحام ذرات

الگوریتم‌های بهینه‌سازی هوش جمعی به عنوان روش‌های بهینه‌سازی مدرن در سال‌های اخیر توجه گسترده‌ای را به خود جلب کرده‌اند و به نتایج قابل‌توجهی در زمینه‌های کنترل فرآیند، بهینه‌سازی، مهندسی، یادگیری ماشین و زمینه‌های دیگر دست یافته‌اند. بیشتر رفتار حیوانات در طبیعت، رفتار دسته جمعی است زیرا آن‌ها به مرور زمان یاد گرفته‌اند که زندگی در یک گروه شانس بقا را افزایش می‌دهد. الگوریتم‌های هوش‌جمعی از رفتار گروه‌هایی از حیوانات یا دسته‌ای از حشرات در هنگام تعامل بین خود و محیط‌شان الهام گرفته‌اند و رفتارهای جمعی حشرات، گله‌ها، پرندگان و ماهی‌ها را شبیه‌سازی می‌کنند که باعث ایجاد یک هوش جمعی می‌شود. رفتار اصلی این گروه‌ها جستجوی غذا است، بدین‌صورت که به‌طور مشارکتی به جستجوی غذا پرداخته و به طور مداوم اطلاعات غذایی را مبادله می‌کنند تا غذا را سریع‌تر پیدا کنند. این موضوع به الگوریتم‌های هوش جمعی اجازه می‌دهد تا با استفاده از رفتار دسته‌ای از عوامل جستجوگر که برای رسیدن به یک هدف با یکدیگر تعامل دارند، بتوانند مسائل بهینه‌سازی پیچیده را حل کنند [۳۵,۳۴].

بهینه‌سازی ازدحام ذرات (PSO) یکی از ابتدایی‌ترین و پرکاربردترین روش‌های مبتنی بر هوش جمعی است و با توجه به مزایای زیادی که دارد، در حل مسائل بهینه‌سازی بسیار زیاد مورد استفاده قرار می‌گیرد. PSO یک روند محاسباتی تکاملی مبتنی بر شبیه‌سازی مدل‌های اجتماعی ساده شده، مانند دسته پرندگان، پرورش ماهی، و نظریه ازدحام است [۳۶]. الگوریتم PSO از تحقیقات بر روی رفتار جستجوی پرندگان نشات گرفته است که با ترکیبی از ماهیت تصادفی و رفتار جمعی، توسط کندی و ابرهارت در سال ۱۹۹۵ معرفی شد. ویژگی جذاب PSO به توانایی ذرات در یادگیری از تجربه دیگران (رفتار اجتماعی) و همچنین از تجربه فردی آنها (رفتار شناختی) نسبت داده می‌شود [۳۷,۲۰].

سادگی ساختار، راحتی در پیاده‌سازی، کاربرد و کارایی بالا و همچنین تعداد فراپارامترهای محدود از مزایای این الگوریتم به شمار می‌روند که سبب شده‌اند تا این الگوریتم توانایی خوبی را در حل مسائل بهینه‌سازی از خود نشان دهد و توجه گسترده محققان را برای بررسی و بهبود عملکرد الگوریتم به خود جلب کند. به خصوص در مسائل پیچیده مهندسی که راه‌حل‌های تقریبی را می‌توان از طریق محاسبات بهینه‌سازی تکراری یافت، الگوریتم PSO به طور گسترده استفاده می‌شود. اما سرعت همگرایی پایینی در مسائل پیچیده و دارای ابعاد بالا دارد [۳۹,۳۸].

الگوریتم PSO بر روی تکرار نسل‌ها کار می‌کند و شباهت‌های زیادی با روش‌های محاسباتی تکاملی دارد. در تکرار اول، مسئله با جمعیتی از ذرات تشکیل می‌شود که هر ذره خود بیانگر یک جواب برای مسئله می‌باشد و همه پارامترهای بهینه‌سازی را در برمی‌گیرد. در تکرارهای بعدی، هر ذره با توجه به کارکرد خود و سایر ذرات در تکرارهای قبلی،

موقعیت خود را اصلاح می‌کند. به‌منظور قانون‌مند کردن رفتار این ذرات، برای هر ذره مشخصه‌هایی شامل: مکان فعلی، سرعت فعلی و بهترین مکان شخصی تعریف می‌شود که بهترین مکان شخصی به مکانی در تکرارهای قبلی گفته می‌شود که ذره مورد نظر در آن بهترین عملکرد را داشته است. همچنین مناسب‌ترین گزینه در بین بهترین مکان‌های شخصی همه ذرات، به عنوان بهترین مکان جمعی درنظر گرفته می‌شود [۴۰-۴۲].

با توجه به موارد گفته شده برای این الگوریتم، سرعت ذره i-ام در تکرار k-ام به صورت زیر در رابطه (۲) بروزرسانی می‌شود.

$$V_i(k+1)=\omega V_i(k)+c_1r_1\left(P_{best,i}-X_i(k)\right)+c_2r_2\left(G_{best}-X_i(k)\right) \quad (۲)$$

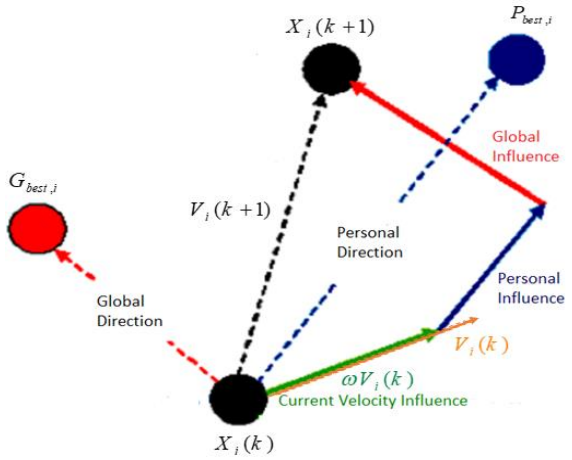
که در آن، k شماره تکرار فعلی، $k+1$ شماره تکرار بعدی، V_i سرعت ذره i-ام، X_i موقعیت ذره i-ام، $P_{best,i}$ بهترین موقعیت شخصی ذره i-ام تاکنون، G_{best} بهترین موقعیت جمعی تا لحظه k، c_1 و c_2 ضرایب شتاب حقیقی، r_1 و r_2 اعداد تصادفی توزیع شده به‌صورت یکنواخت و ω وزن اینرسی ذره می‌باشد.

در رابطه (۲)، ضرایب ثابت c_1 و c_2 میزان تاثیر بهترین موقعیت فردی و جمعی بر سرعت هر ذره را کنترل می‌کنند. همچنین اعداد r_1 و r_2 تصادفی بوده و در بازه $[0\ ,\ 1]$ می‌باشند و برای حفظ سطح تنوع در جمعیت استفاده می‌شوند. در این روش هوشمند، پارامترهای تنظیم (فراپارامترها) محدود به دو پارامتر c_1 و c_2 می‌شود که البته برای تنظیم این دو فراپارامتر (برخلاف انتخاب طول‌گام در روش گرادیان) بازه استانداردی وجود دارد و در نتیجه انتخاب این دو پارامتر، مسئله مشکل‌سازی نیست.

موقعیت ذره i-ام در گام k، مطابق با معادله (۳) بروزرسانی می‌شود.

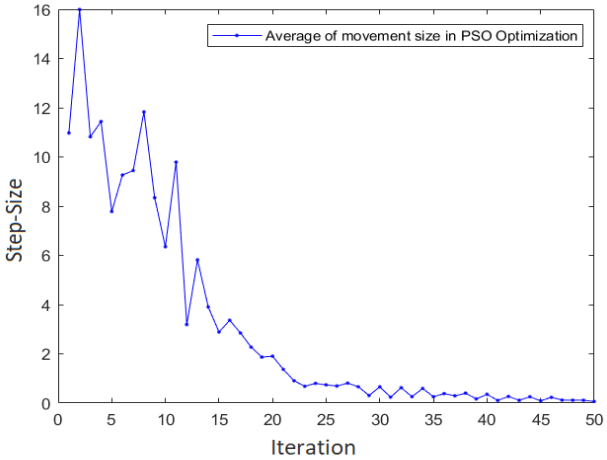
$$X_i(k+1)=X_i(k)+V_i(k+1) \quad (۳)$$

شکل (۱) وضعیت بروزرسانی موقعیت یک ذره را در الگوریتم PSO نمایش می‌دهد. در این شکل نحوه بروزرسانی سرعت ذره i-ام نشان داده شده و سپس از بردار سرعت جهت بروزرسانی موقعیت ذره i-ام از جمعیت استفاده شده است [۱۹].



شکل (۱): طرح بروزرسانی موقعیت ذرات در الگوریتم PSO [۱۹]

در الگوریتم PSO اگر اندازه میانگین بردار سرعت (که همان میانگین جابجایی ذرات در هر تکرار است) رسم گردد، مشاهده می‌شود که در گام‌های ابتدایی و با فاصله دورتر از نقطه بهینه، سرعت حرکت به سمت جواب بهینه زیاد می‌باشد. با نزدیک شدن به نقطه بهینه، به صورت خودکار سرعتِ حرکت رفته رفته کاهش می‌یابد و با رسیدن به نقطه بهینه، سرعتِ حرکت صفر می‌شود. این طول‌گام متغیر از ساختار روش PSO به دست می‌آید و نیازی به تنظیم آن نیست و این مسئله از مزایای روش PSO در مقایسه با روش گرادیان است. این مطلب در شکل (۲) نشان داده شده است. محور عمودی در شکل (۲)، بیانگر طول‌گام است که اندازه جابجایی را نشان می‌دهد و بنابراین همواره مقداری مثبت است.



شکل (۲): میانگین اندازه جابجایی ذرات در الگوریتم PSO

۴- الگوریتم گرادیان پیشنهادی

در این مقاله، هدف آن است که با الهام گرفتن از الگوریتم PSO، یک الگوریتم گرادیان اصلاح شده‌ای ارائه شود به‌نحوی‌که مسئله انتخاب طول‌گام حرکت در آن وجود نداشته باشد و طول‌گام مشابه روش PSO به‌صورت خودکار تغییر کند. در الگوریتم گرادیان فقط یک موقعیت (ذره) وجود دارد، در حالی‌که در الگوریتم‌های تکاملی و هوش جمعی و مخصوصاً PSO، چندین موقعیت (ذره) در هر گام وجود دارد. بنابراین برای اینکه بتوان این دو روش را با هم ادغام کرد، باید صورت مسئله هر دو را به هم نزدیک کرد. لذا، ابتدا باید معادله حرکت در روش PSO (که مربوط به چندین ذره می‌باشد) را به حالتی که فقط یک عضو وجود دارد، تغییر داد. در این صورت، معادله حرکت الگوریتم PSO به ازای یک ذره مطابق با رابطه (۴) می‌باشد.

$$V(k+1)=\omega V(k)+c_1r_1\left(P_{best}-X\right)+c_2r_2\left(G_{best}-X\right) \quad (۴)$$

حال با توجه به وجود فقط یک ذره، می‌توان نتیجه گرفت که P_{best} (بهترین موقعیت فردی) و G_{best} (بهترین موقعیت جمعی) با یکدیگر برابر خواهند شد، در نتیجه معادله (۴) به رابطه (۵) تبدیل خواهد شد.

$$\begin{cases} c_0 := c_1 r_1 + c_2 r_2 \\ P_{best} = G_{best} \end{cases} \Rightarrow \quad V(k+1) = \omega V(k) + c_0 \left(P_{best} - X \right) \tag{۵}$$

۴-۱- ارائه روش پیشنهادی با ترکیب دو روش گرادیان و PSO

در ادامه برای ورود الگوریتم گرادیان به مسئله فوق، فرض می‌کنیم که بهترین موقعیت P_{best}، همان جواب بهینه مسئله می‌باشد. در این صورت جهت $P_{best} - X$ به‌کار برده شده در رابطه (۵) را می‌توان معادل جهت رسیدن به جواب بهینه در نظر گرفت. در این مقاله برای ترکیب دو روش گرادیان و PSO، به‌جای جهت رسیدن به جواب بهینه، از جهت گرادیان استفاده شده است. در این‌صورت رابطه (۶) به دست می‌آید.

$$\begin{cases} \overline{V}(k+1) = \omega \overline{V}[k] - c_0 g_k \\ g_k = \nabla_X J(X[k]) \end{cases} \tag{۶}$$

در روش PSO اندازه حرکت $P_{best} - X$ با واقعیت (فاصله موقعیت فعلی نسبت به بهترین جواب) هم‌خوانی دارد ولی در روش گرادیان، اندازه g_k هیچ تناسبی با فاصله موقعیت فعلی نسبت به جواب بهینه ندارد. لذا در روش پیشنهادی، به‌جای گرادیان، از گرادیان نرمالیزه شده (نسبت به اندازه حرکت $\overline{V}$) به‌صورت رابطه (۷) استفاده می‌شود.

$$\begin{cases} \overline{V}(k+1) = \omega \overline{V}[k] - c_0 \overline{g}_k \\ \overline{g}_k := g_k \dfrac{\left\| \overline{V} \right\|}{\left\| g_k \right\|} \end{cases} \tag{۷}$$

همچنین برای حل مشکل در زمانی که $g_k \to 0$، رابطه فوق به صورت زیر در رابطه (۸) اصلاح می‌شود

$$\overline{g}_k := g_k \frac{\left\| \overline{V} \right\| + \beta}{\left\| g_k \right\| + \beta} = g_k \frac{\sqrt{\overline{V}^T \overline{V}} + \beta}{\sqrt{g_k^T g_k} + \beta} \tag{۸}$$

در این‌صورت جهت حرکت در روش پیشنهادی مطابق با رابطه (۹) بروزرسانی می‌شود.

$$\overline{V}[k+1] = \omega \overline{V}[k] - c_0 g_k \frac{\sqrt{\overline{V}^T \overline{V}} + \beta}{\sqrt{g_k^T g_k} + \beta} \tag{۹}$$

در نهایت بروزرسانی موقعیت در الگوریتم گرادیان پیشنهادی نیز به صورت رابطه (۱۰) بیان می‌شود.

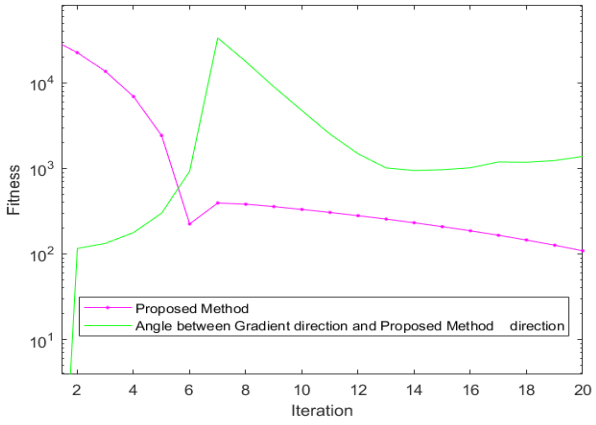
$$X[k+1] = X[k] + \overline{V}[k+1] \tag{۱۰}$$

که $\overline{V}[k+1]$ همان جهت حرکت در روش پیشنهادی (ترکیب دو روش گرادیان و روش PSO) می‌باشد. متغیرهای روابط (۹) و (۱۰) در جدول (۱) ذکر شده‌اند.

جدول (۱): متغیرهای الگوریتم گرادیان پیشنهادی

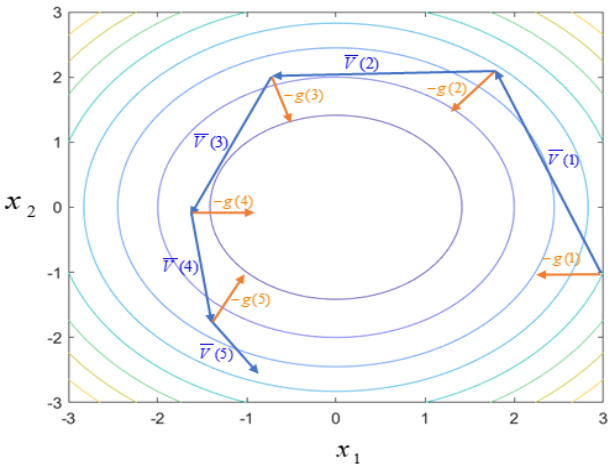
متغیر	توضیحات
$\overline{V}[k+1]$	جهت حرکت پیشنهادی در گام $k+1$
$\overline{V}[k]$	جهت حرکت پیشنهادی در گام k
$X[k+1]$	موقعیت در گام $k+1$
$X[k]$	موقعیت در گام k
ω	ضریب وزن اینرسی
c_0	ضرایب شتاب حقیقی
g_k	گرادیان تابع هزینه نسبت به موقعیت
β	مقدار ثابت جهت جلوگیری از صفر شدن مخرج

در اجرای روش پیشنهادی گاهی اوقات مشاهده می‌شود که تابع هزینه کاهشی نبوده و جهت حرکت در روش پیشنهادی خیلی از جهت گرادیان فاصله گرفته و سبب جابجایی موقعیت در جهت اشتباه می‌شود. مشکل فوق در شکل (۳) و در گام دوم بهینه‌سازی مشاهده می‌شود که تابع هزینه بجای کاهش، افزایش یافته است.



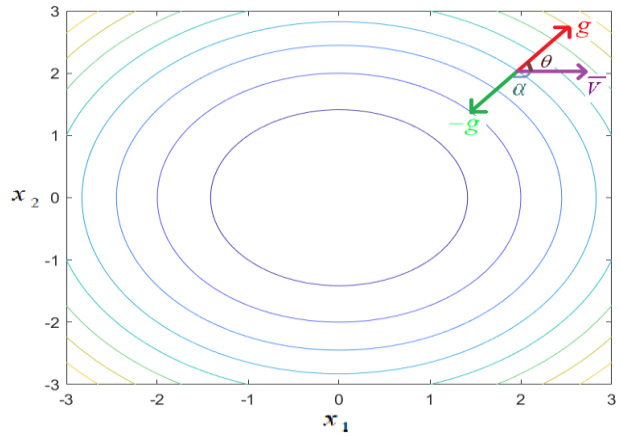
شکل (۳): مشکل عدم هم‌خوانی جهت حرکت پیشنهادی و جهت گرادیان

با بررسی بیشتر، مشاهده شد که این اتفاق زمانی رخ می‌دهد که جهت حرکت پیشنهادی با جهت گرادیان هم‌خوانی نداشته و زاویه بین آن دو بیشتر از ۹۰ درجه شده است، یا زمانی که جهت حرکت پیشنهادی تا لحظه قبل درست بوده اما طول‌گام به‌قدری بزرگ است که الگوریتم پیشنهادی برای گام بعدی، از نقطه بهینه رد شده است. بنابراین در گام بعدی جهت گرادیان جدید برعکس جهت آن در گام‌های قبلی می‌شود. اما با توجه به دینامیک تعریف شده در روابط (۶) تا (۹)، بردار $\overline{V}$ دارای دینامیکی شبیه انتگرال‌گیر بوده و کند می‌باشد و لذا $\overline{V}$ سریع نمی‌تواند این تغییر جهت بردار گرادیان را تعقیب کند و در نتیجه مشابه شکل (۴)، بردار $\overline{V}$ در گام پنجم از جهت مشخص شده توسط گرادیان فاصله می‌گیرد.



شکل (۴): مشکل عدم هم‌خوانی جهت حرکت مبتنی بر حافظه و جهت گرادیان در یک مسئله دو بعدی

در شکل‌های (۴) تا (۶) کانتورهای یک تابع هزینه درجه دو و دو بعدی رسم گردیده‌اند. بردار g نشان دهنده گرادیان تابع هزینه و بردار $\overline{V}$ نشان‌دهنده جهت حرکت پیشنهادی اشتباه می‌باشد. همانطور که مشاهده می‌گردد در شکل (۵) ، بردار $\overline{V}$ نسبت به منفی جهت گرادیان دارای زاویه بیشتر از ۹۰ درجه بوده و در نتیجه سبب افزایش تابع هزینه می‌شود.



شکل (۵): انحراف بیش از حد جهت حرکت نسبت به جهت گرادیان لحظه‌ای در یک مسئله دو بعدی

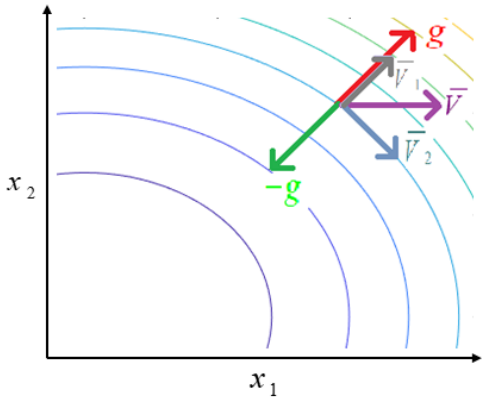
۴-۲- اصلاح الگوریتم پیشنهادی با روش تصویر بردار

برای تضمین همیشگی جهت حرکت کاهشی، باید زاویه بین جهت حرکت پیشنهادی و جهت روش گرادیان همواره بین ۰ تا ۹۰ درجه باشد. بدین منظور در گام‌هایی که زاویه بین جهت حرکت پیشنهادی و جهت روش گرادیان از ۹۰ درجه بیشتر شود، باید با استفاده از روشی مناسب، این زاویه را به ۹۰ درجه کاهش داد که در این مقاله، از روش تصویر بردار برای این منظور استفاده شده است [۴۳].

زاویه بین جهت حرکت روش پیشنهادی و جهت گرادیان (α برحسب درجه) از ضرب داخلی آن‌ها و به‌صورت رابطه (۱۱) قابل محاسبه است.

$$\alpha = \frac{180}{\pi}\left(\cos^{-1}\left(\frac{\overline{V}\, g_k^T}{\|\overline{V}\|\|g_k\|}\right)\right) \qquad (۱۱)$$

با استفاده از تصویر بردار، بردار $\overline{V}$ را می‌توان نسبت به بردار g، به دو مولفه مماسی و عمودی تجزیه کرد. با توجه به شکل (۶) ، بردار $\overline{V}_1$ (تصویر $\overline{V}$ روی g) همان مولفه مماسی بوده و بردار $\overline{V}_2$ نیز مولفه عمودی می‌باشد. هر زمان که زاویه α بیشتر از ۹۰ درجه شد، مولفه $\overline{V}_1$ در خلاف جهت جواب بهینه بوده و سبب افزایش تابع هزینه می‌شود. لذا در این حالت از این مولفه صرف‌نظر کرده و بجای بردار $\overline{V}$، فقط از مولفه عمودی یعنی از بردار $\overline{V}_2$ استفاده می‌شود.



شکل (۶): تصویر بردار حرکت پیشنهادی بر روی بردار گرادیان

هدف آن است که مولفه عمودی یعنی $\overline{V}_2$ به عنوان جهت حرکت پیشنهادی اصلاح شده (هر زمانی‌که زوایه α بیشتر از ۹۰ درجه شود) در نظر گرفته شود و این جهت حرکت پیشنهادی اصلاح شده $\overline{\overline{V}}$ نامیده می‌شود و مطابق با رابطه (۱۲) قابل محاسبه است.

$$\overline{\overline{V}} = \overline{V}_1 + \overline{V}_2 \qquad (۱۲)$$

مولفه مماسی جهت حرکت پیشنهادی یعنی $\overline{V}_1$ از رابطه (۱۳) محاسبه می‌شود.

$$\overline{V}_1 = \|\overline{V}\| \cos\alpha \frac{g}{\|g\|} \qquad (۱۳)$$

که در آن $\cos\alpha$ را می‌توان به‌صورت رابطه (۱۴) به‌دست آورد.

$$\cos\alpha = \frac{\overline{V}.g}{\|\overline{V}\|\|g\|} \qquad (۱۴)$$

در این‌صورت با ترکیب دو رابطه (۱۳) و (۱۴)، می‌توان مولفه مماسی را مطابق روابط (۱۵) و (۱۶) به‌دست آورد:

$$\overline{V}_1 = \left\|\overline{V}\right\| \frac{\overline{V}.g}{\left\|\overline{V}\right\|\|g\|}\frac{g}{\|g\|} = \frac{\overline{V}.g}{\|g\|^2}\, g \qquad (۱۵)$$

$$\overline{V}_1 = \frac{\overline{V}^T g}{g^T g}\, g \qquad (۱۶)$$

حال با استفاده از روابط (۱۲) و (۱۶)، می‌توان مولفه عمودی را به صورت رابطه (۱۷) محاسبه کرد:

$$\overline{V}_2 = \overline{V} - \overline{V}_1 = \overline{V} - \frac{\overline{V}^T g}{g^T g}\, g \qquad (۱۷)$$

همان‌طور که در شکل (۶) مشاهده می‌شود، در مواردی که زوایه α از ۹۰ درجه بیشتر می‌شود، بردار $\overline{V}$ (که در راستای مناسب و در جهت کاهش تابع هزینه قرار ندارد) تبدیل به دو بردار $\overline{V}_1$ و $\overline{V}_2$ شده که مولفه $\overline{V}_1$ در جهت عکس بهینه‌سازی بوده و مولفه $\overline{V}_2$ خنثی می‌باشد. در این حالت مولفه $\overline{V}_1$ را حذف کرده و فقط مولفه $\overline{V}_2$ را نگه داشته و لذا به جای $\overline{V}$ از $\overline{V}_2$ استفاده می‌شود.

برای پیاده‌سازی روش پیشنهادی در هر گام، ابتدا زاویه جهت حرکت پیشنهادی با جهت گرادیان محاسبه شده و در صورت بزرگ‌تر بودن این زاویه از ۹۰ درجه، جهت حرکت پیشنهادی به‌صورت ذکر شده (روش تصویر بردار) اصلاح می‌شود به نحوی‌که زاویه آن با جهت گرادیان همیشه بین ۰ تا ۹۰ درجه محدود بماند. جهت حرکت اصلاح شده، $\overline{\overline{V}}$ نامیده می‌شود. با اعمال این اصلاح، هر زمانی‌که زوایه α بخواهد بیشتر از ۹۰ درجه شود، علاوه بر محدود کردن زاویه به ۹۰ درجه، اندازه طول‌گام نیز نسبت به حالت قبل کوچک‌تر خواهد شد (زیرا یک مولفه از $\overline{V}$ یعنی $\overline{V}_1$ حذف شده و در نتیجه اندازه حرکت جدید یعنی $\overline{\overline{V}}$ از اندازه حرکت قبلی یعنی $\overline{V}$ کوچک‌تر می‌شود).

حال موقعیت جدید X در روش پیشنهادی اصلاح شده به صورت رابطه (۱۸) بروزرسانی می‌شود:

$$\begin{cases} \overline{\overline{V}} = \begin{cases} \overline{V} & ; \quad if \quad \alpha \le 90 \\ \overline{V}_2 & ; \quad if \quad \alpha > 90 \end{cases} \\ X[k+1] = X[k] + \overline{\overline{V}}[k+1] \end{cases} \qquad (۱۸)$$

با توجه به اصلاحات انجام شده روی روش پیشنهادی انتظار می‌رود که دیگر مشکل جهت حرکت اشتباه و ایجاد بالازدگی در برخی گام‌ها رفع شده باشد. البته هنوز ممکن است مواردی وجود داشته باشد که با وجود محدود بودن زاویه جهت حرکت نسبت به جهت گرادیان، باز هم تابع هزینه کاهشی نباشد که در این موارد، علت افزایش تابع هزینه جهت حرکت اشتباه نیست بلکه مشکل در مقدار بسیار بزرگ طول‌گام می‌باشد که سبب شده است تا جواب مسئله از جواب بهینه دورتر شود.

۵- ارزیابی و شبیه‌سازی

در ادامه الگوریتم گرادیان پیشنهادی را بر روی ده تابع محک[۲۴] اعمال کرده و نتیجه با الگوریتم گرادیان کلاسیک مقایسه خواهد شد. لازم به ذکر است تمام شبیه‌سازی‌های این مقاله در محیط برنامه‌نویسی متلب[۲۵] و توسط کامپیوتر با ۶ گیگ رم و پردازنده Core(TM) i5-4210 در محیط ویندوز ۱۰ و ۶۴ بیت انجام شده‌اند. همچنین تمامی نتایج پس از سی مرتبه اجرا و سپس میانگین‌گیری گزارش شده‌اند. نام، فرمول، ابعاد، بازه تغییرات ورودی توابع محک و مراجع آن‌ها در جدول (۲) و سپس مقادیر پارامترهای تنظیم شده در جدول (۳) ذکر شده‌اند.

جدول (۲): توابع محک

بازه تغییرات ورودی	ابعاد	فرمول	نام	مرجع
[-۵,۵]	۲	$f_1(x) = 4 - 4x_1^3 + 4x_1 + x_2^2$	Trecannie	[۴۴]
[-۵۰۰,۵۰۰]	۲	$f_2(x) = (2x_1^3 x_2 - x_2^3)^2 + (6x_1 - x_2^2 + x_2)^2$	Price	[۴۴]
[-۵,۵]	۲	$f_3(x) = (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	Six Hump Camel	[۱۱]
[-۱۰,۱۰]	۴	$f_4(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1(x_2 - 1)^2 + (x_4 - 1)^2 + 19.8(x_2 - 1)^2 + (x_4 - 1)^2$	Colville	[۴۵]
[-۵,۱۰]	۴	$f_5(x) = \sum_{k=1}^{D=4} x_k^2 + \sum_{k=1}^{D=4} (0.5kx_k)^2 + \sum_{k=1}^{D=4} (0.5kx_k)^4$	Zakharov	[۴۵]
[-۱۰۰,۱۰۰]	۱۰	$f_6(x) = \sum_{k=1}^{D=10} x_k^2$	Sphere	[۴۵]
[-۵,۵]	۱۰	$f_7(x) = \frac{1}{2}\sum_{k=1}^{D=10} (x_k^4 - 16x_k^2 + 5x_k)$	Styblinski-tang	[۱۱]

[۴۶]	Rastrigin	$f_8(x)=\sum_{k=1}^{D=30}(x_k^2-10\cos(2\pi x_k)+10)$	۳۰	[-۵/۱۲,۵/۱۲]
[۴۴]	Schwefel 2.20	$f_9(x)=\sum_{k=1}^{D=30}\lvert x_k\rvert$	۳۰	[-۱۰۰,۱۰۰]
[۴۴]	Schwefel 2.23	$f_{10}(x)=\sum_{k=1}^{D=30}x_k^{10}$	۳۰	[-۱۰,۱۰]

جدول (۳): مقدار پارامترهای مورد استفاده در روش‌های پیاده‌سازی شده برای توابع محک

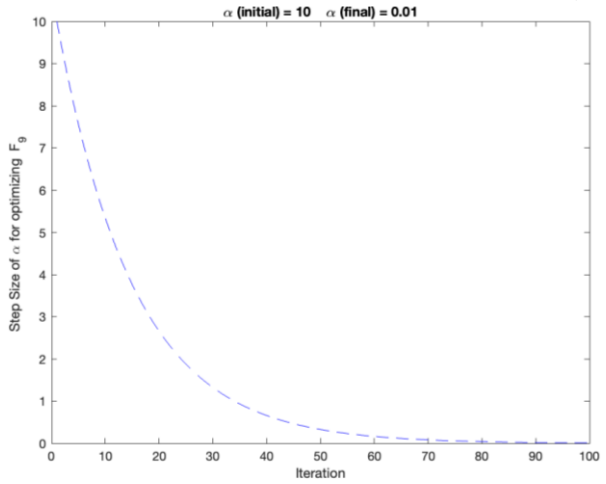
متغیر	مقدار اولیه پارامترها	مقدار نهایی پارامترها
η_1	۰/۰۱	۱e-۰۵
η_2	۱e-۱۷	۱e-۱۹
η_3	۰/۰۰۱	۰/۰۰۱
η_4	۲e-۰۵	۱e-۰۵
η_5	۱e-۰۴	۱e-۰۵
η_6	۰/۱۰	۰/۰۱
η_7	۰/۰۱	۰/۰۰۱
η_8	۰/۰۰۱	۰/۰۰۰۲
η_9	۱۰	۰/۰۱
η_{10}	۱e-۰۹	۱e-۰۹
ω	۰/۵۰	۰/۵۰
c_0	۰/۵۱	۰/۵۱
β	۰/۰۱	۰/۰۱

در این شبیه‌سازی‌ها، در گام اول، بردار حرکت اولیه به‌صورت زیر تعیین می‌شود:

$$\begin{cases}\overline{V}(1)=-\mu g_1\\ \mu=10\eta_i\end{cases} \qquad (۱۹)$$

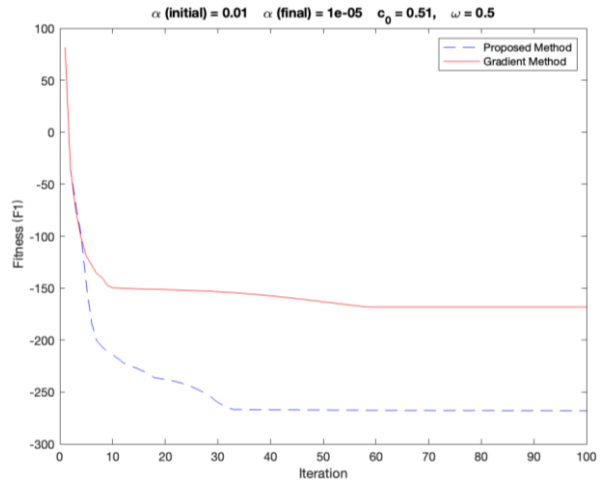
که در آن η_i یک تخمین اولیه برای طول گام می‌باشد. در روش پیشنهادی، گاهی اوقات جهت حرکت به جای کاهشی، تبدیل به افزایشی شده که باعث می‌شود الگوریتم در جهت اشتباه حرکت کرده و از روش گرادیان فاصله بگیرد که در شبیه‌سازی‌ها برای رفع این مشکل، از روش تصویر بردار استفاده شده است.

همان‌طور که پیش‌تر بیان شد، عملکرد الگوریتم گرادیان به انتخاب مقدار مناسب برای طول‌گام آن بستگی دارد که انتخاب این مقدار نیز به صورت تجربی انجام می‌شود. در این مقاله جهت انتخاب بهترین طول‌گام برای هر کدام از توابع محک، دو مقدار اولیه و نهایی به صورت تجربی برای طول‌گام در نظر گرفته شده است و مناسب‌ترین مقدار طول‌گام به صورت متغیر در این بازه برای هر تابع محک انتخاب شده و برای پارامترهای مربوط به روش پیشنهادی نیز مقادیر ثابتی در نظر گرفته شده است. مقدار ابتدایی و انتهایی طول‌گام برای هر یک از توابع محک، در جدول (۳) و شکل‌ها نیز درج شده است. در شکل (۷)، نمونه‌ای از تغییرات طول‌گام برای تابع محک نهم در طول ۱۰۰ گام رسم شده است.



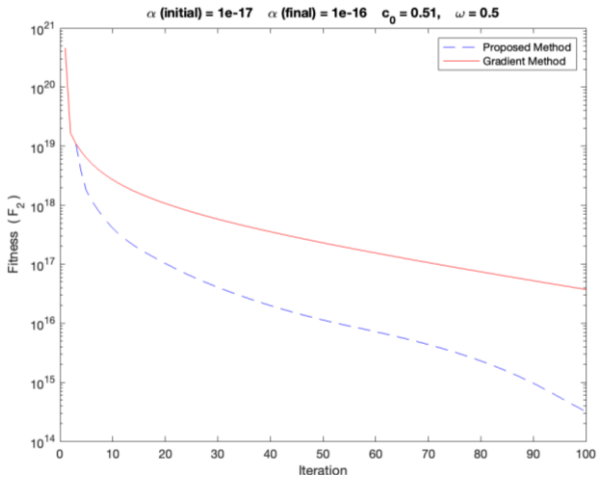
شکل (۷): نمونه‌ای از تغییرات طول‌گام و انتخاب طول‌گام بهینه روش گرادیان

در شکل های (۸) تا (۱۷) منحنی الگوریتم گرادیان کلاسیک و الگوریتم پیشنهادی پس از سی مرتبه اجرا و میانگین‌گیری، رسم شده‌اند. لازم به ذکر است که محور افقی تعداد گام‌های اجرا و محور عمودی مقدار تابع محک به ازای موقعیت را نشان می‌دهد که شایستگی[۲۶] تعریف شده است.



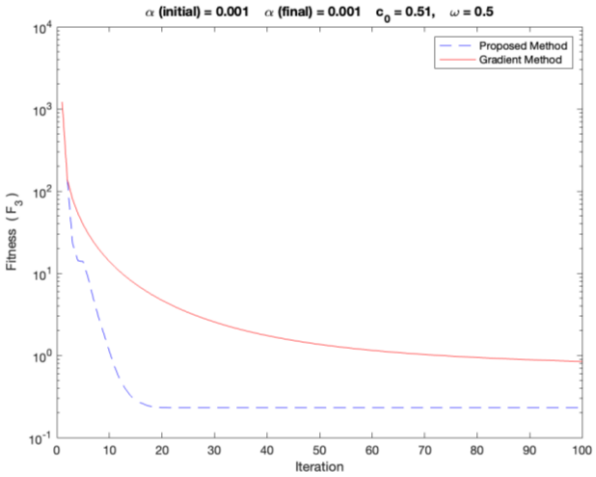
شکل (۸): ارزیابی عملکرد الگوریتم گرادیان کلاسیک و الگوریتم گرادیان پیشنهادی توسط تابع محک F1

در شکل (۸) برای بهینه‌سازی تابع محک اول، روش گرادیان عملکرد نسبتاً مناسبی از خود نشان نداده است. با توجه به جدول (۲) بازه تغییرات متغیرها در این تابع محک، [۵,۵-] می‌باشد درحالی‌که جواب بهینه متغیر x_1 در این بازه قرار ندارد. بنابراین هرکدام از روش ها باید بتوانند مقدار x_1 را روی مرز یعنی $x_1 = ۵$ و مقدار x_2 را روی جواب بهینه $x_2 = ۰$ تنظیم کند که ضعف روش گرادیان، در تطبیق تنظیم این دو مقدار می‌باشد اما روش پیشنهادی، به خوبی توانسته این تنظیم را انجام دهد. همچنین مقادیر اولیه و نهایی متفاوت برای طول-گام روش گرادیان درنظر گرفته شد اما منجر به نتایج بهتر نگردید و بهترین مقدار اولیه برای طول‌گام برابر ۰/۰۱ و بهترین مقدار نهایی طول‌گام برابر با ۱e-۰۵ انتخاب شد.



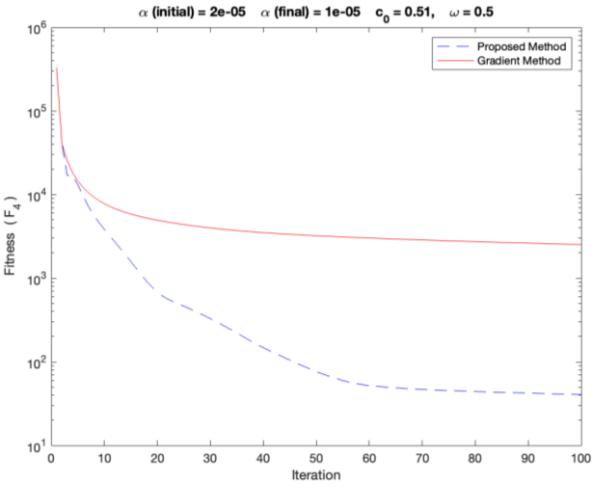
شکل (۹): ارزیابی عملکرد الگوریتم گرادیان کلاسیک و الگوریتم گرادیان پیشنهادی توسط تابع محک F2

در شکل (۹) برای بهینه‌سازی تابع محک دوم، مقادیر اولیه و نهایی طول‌گام چندین مرتبه تغییر داده شد، اما نتایج مناسب‌تری برای روش گرادیان حاصل نشد. بنابراین مقدار اولیه طول‌گام برابر با ۱e-۱۷ و مقدار نهایی آن نیز ۱e-۱۶ در نظر گرفته شد. روش پیشنهادی بدون هیچ تنظیم اختصاصی، نتایج بسیار خوبی از خود نشان داده است. البته اگر برای تابع محک دوم کاری خلاف عرف انجام شده و مقدار نهایی طول گام، بیشتر از مقدار شروع آن لحاظ شود، نتایج روش گرادیان کمی بهتر خواهد شد اما بازهم به نتایج مناسبی که روش پیشنهادی برای این تابع محک (که هیچ تنظیم اختصاصی برای آن انجام نشد) نمی‌رسد.



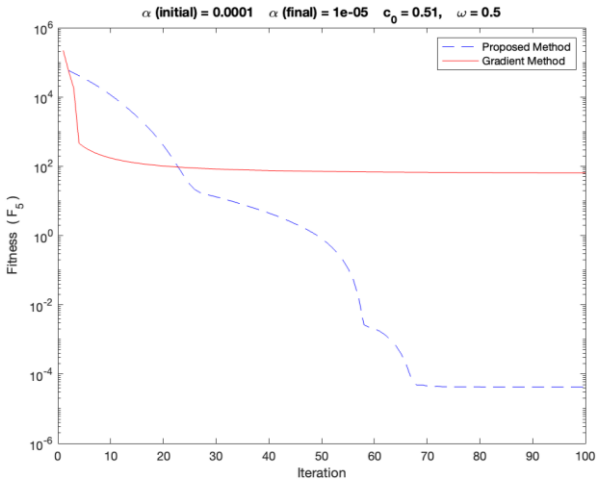
شکل (۱۰): ارزیابی عملکرد الگوریتم گرادیان کلاسیک و الگوریتم گرادیان پیشنهادی توسط تابع محک F3

در شکل (۱۰) برای بهینه‌سازی تابع محک سوم، بهترین نتایج روش گرادیان به ازای در نظر گرفتن مقدار یکسان اولیه و نهایی طول‌گام و برابر با ۰/۰۰۱ به دست آمده که البته باز هم نتایج روش پیشنهادی نسبت به روش گرادیان بهتر بوده است.



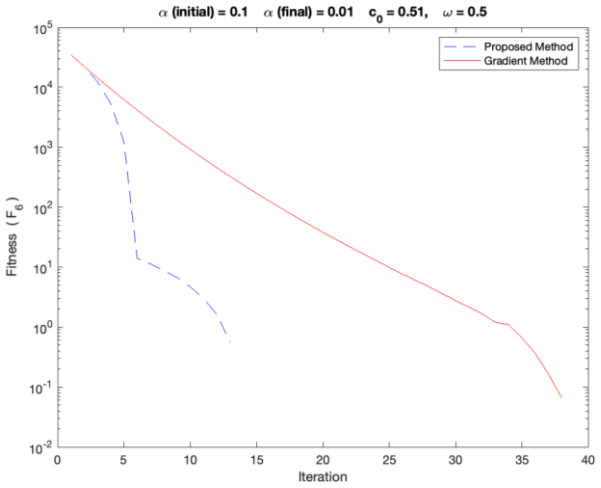
شکل (۱۱): ارزیابی عملکرد الگوریتم گرادیان کلاسیک و الگوریتم گرادیان پیشنهادی توسط تابع محک F4

در شکل (۱۱) برای بهینه‌سازی تابع محک چهارم، بهترین نتایج روش گرادیان به ازای مقدار اولیه طول‌گام برابر ۲e-۰۵ و مقدار نهایی e-۰۵ ۱ به دست آمد. که نتایج روش پیشنهادی نسبت به روش گرادیان بسیار مناسب‌تر است. همچنین روش گرادیان سریع و در گام‌های ابتدایی به یک جواب بهینه محلی همگرا شده درحالی‌که روش پیشنهادی در گام‌های انتهایی توانسته به جواب بهینه بهتری برسد.



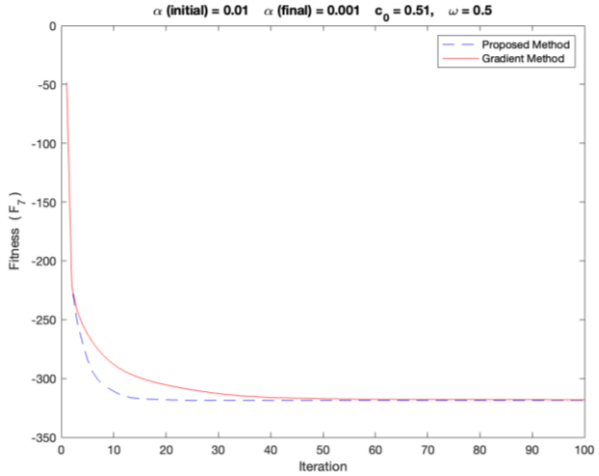
شکل (۱۲): ارزیابی عملکرد الگوریتم گرادیان کلاسیک و الگوریتم گرادیان پیشنهادی توسط تابع محک F5

در شکل (۱۲) برای بهینه‌سازی تابع محک پنجم، مقدار اولیه طول‌گام برابر با ۱e-۰۴ و مقدار نهایی طول‌گام برابر با ۱e-۰۵ در نظر گرفته شد که به صورت تجربی بهترین مقادیر انتخابی بوده‌اند و انتخاب مقادیر اولیه و نهایی متفاوت نیز تغییر مثبتی در نتایج روش گرادیان ایجاد نکرده است. در بهینه‌سازی تابع محک پنجم نیز ابتدا سرعت همگرایی روش گرادیان بیشتر از روش پیشنهادی بوده اما سریع به سمت یک جواب بهینه محلی همگرا شده است درحالی که روش پیشنهادی بدون هیچ تنظیم اختصاصی، نتایج بسیار عالی دارد.



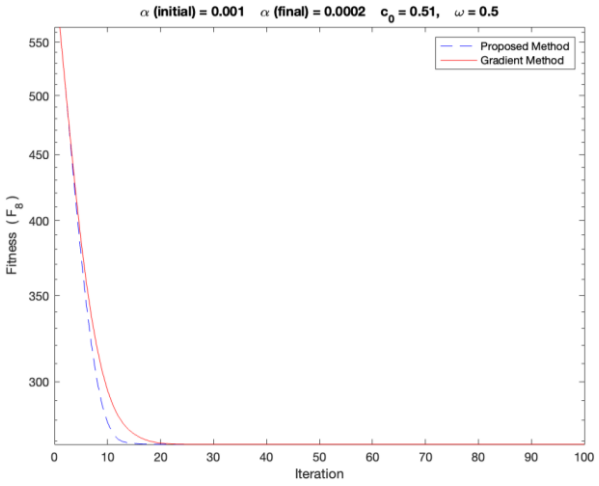
شکل (۱۳): ارزیابی عملکرد الگوریتم گرادیان کلاسیک و الگوریتم گرادیان پیشنهادی توسط تابع محک F6

در شکل (۱۳) برای بهینه‌سازی تابع محک ششم، هر دو روش نتایج خوبی دارند ولی بازهم نتایج روش پیشنهادی بهتر است. برای روش گرادیان مقدار اولیه طول‌گام برابر با ۰/۰۱ و مقدار نهایی برابر ۰/۰۰۱ لحاظ شده است. شکل (۱۳) به علت این‌که هر دو روش خیلی زود به جواب بهینه دقیقا صفر رسیده‌اند، در نمایش لگاریتمی محور عمودی فقط تا گام قبل از رسیدن به صفر نمایش داده شده است زیرا جایی که جواب به مقدار بهینه صفر می‌رسد (صفر در حالت لگاریتمی منفی بینهایت می‌شود) را نمی‌توان رسم کرد.



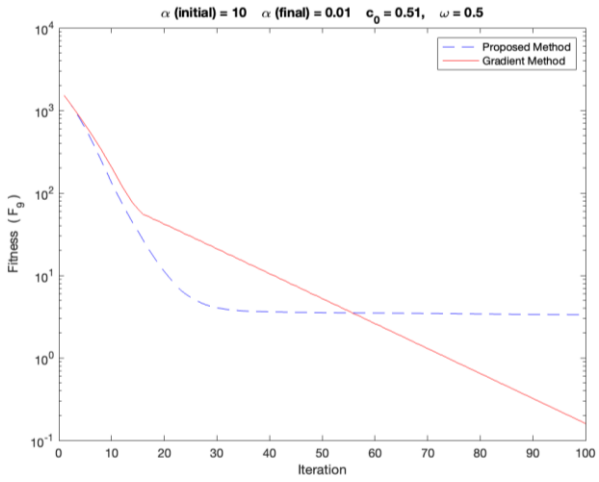
شکل (۱۴): ارزیابی عملکرد الگوریتم گرادیان کلاسیک و الگوریتم گرادیان پیشنهادی توسط تابع محک F7

در شکل (۱۴) برای بهینه‌سازی تابع محک هفتم، بهترین نتایج روش گرادیان به ازای مقدار اولیه طول‌گام برابر با ۰/۰۱ و مقدار نهایی طول-گام برابر با ۰/۰۰۱ به دست آمده است. در این تابع محک نیز نتایج روش پیشنهادی (بدون تنظیم اختصاصی) بهتر بوده است. البته اگر در روش پیشنهادی ضرایب c_0 و ω با توجه به این مسئله (بهینه‌سازی تابع محک هفتم) تنظیم شود، نتایج بهتری به دست خواهد آمد البته بدون این تنظیمات هم نتایج روش پیشنهادی قابل قبول است.



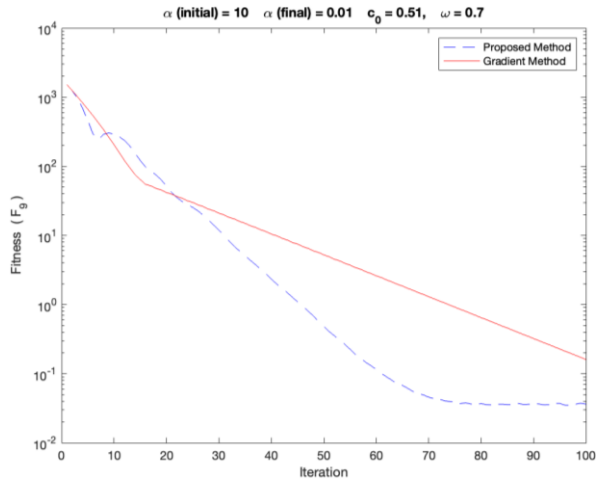
شکل (۱۵): ارزیابی عملکرد الگوریتم گرادیان کلاسیک و الگوریتم گرادیان پیشنهادی توسط تابع محک F8

در شکل (۱۵) برای بهینه‌سازی تابع محک هشتم، مقدار اولیه و نهایی طول‌گام روش گرادیان به ترتیب برابر با ۰/۰۰۱ و ۲e-۰۴ به دست آمده است. با وجود اینکه هر دو روش نتایج خوبی دارند ولی نتایج روش پیشنهادی بهتر است.



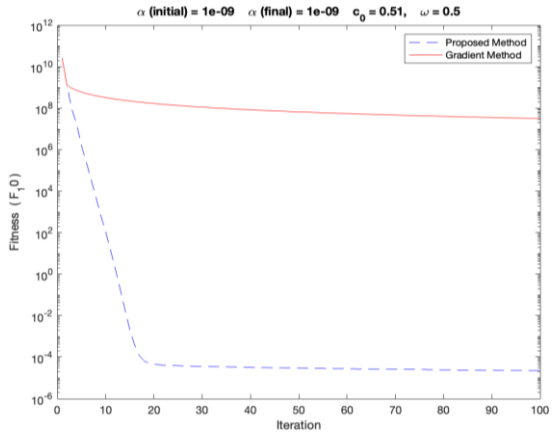
شکل (۱۶): ارزیابی عملکرد الگوریتم گرادیان کلاسیک و الگوریتم گرادیان پیشنهادی توسط تابع محک F9

در شکل (۱۶) برای بهینه‌سازی تابع محک نهم، بهترین نتایج روش گرادیان به ازای مقدار اولیه طول‌گام برابر با ۱۰ و مقدار نهایی برابر با ۰/۰۱ حاصل شده است که در بهینه‌سازی این تابع محک، نتایج روش گرادیان (با تنظیم تجربی طول‌گام) مقداری بهتر از نتایج روش پیشنهادی (بدون تنظیم اختصاصی) است. البته اگر در روش پیشنهادی ضرایب c_0 و ω با توجه به این مسئله (بهینه‌سازی تابع محک نهم) تنظیم شود، نتایج روش پیشنهادی بهتر خواهد شد که در شکل (۱۷)، نتایج روش پیشنهادی به ازای c_0=۰/ ۵۱ و ω=۰/۷ آمده است.



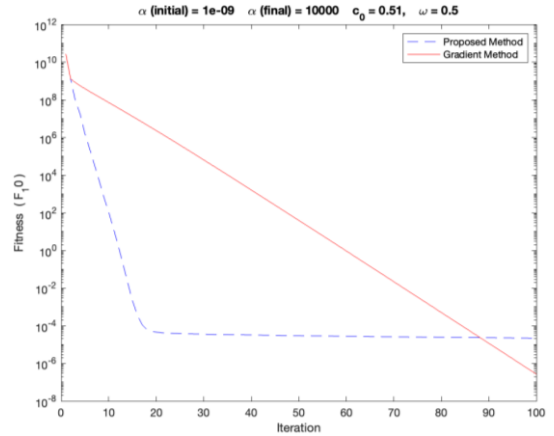
شکل (۱۷): ارزیابی عملکرد الگوریتم گرادیان کلاسیک و الگوریتم گرادیان پیشنهادی با تنظیم اختصاصی توسط تابع محک F9

در شکل (۱۷)، مقدار ω به میزان ٪۴۰ تغییر کرده و نتایج روش پیشنهادی برای بهینه‌سازی تابع محک نهم به مراتب بهتر از نتایج گرادیان شده است. بنابراین مشاهده شد که در ٪۹۰ موارد روش پیشنهادی بدون تنظیم پارامترها نتایج خوبی دارد و فقط در ٪۱۰ موارد پارامترهای (ضرایب) روش پیشنهادی حدود ٪۴۰ تغییر، نیاز به تنظیم اختصاصی مسئله دارند. درحالی‌که در روش گرادیان میزان تغییرات طول‌گام از یک مسئله تا مسئله دیگر صدها یا هزاران برابر ممکن است متغیر باشد و این مزیت اصلی روش پیشنهادی است که با کمترین تنظیم و تغییر در پارامترها، بهترین نتایج را در مسئله‌های متفاوت را به دست می‌آورد.



شکل (۱۸): ارزیابی عملکرد الگوریتم گرادیان کلاسیک و الگوریتم گرادیان پیشنهادی توسط تابع محک F10

در شکل (۱۸) و بهینه‌سازی تابع محک دهم، اگر مقدار طول‌گام به صورت متعارف تنظیم شود (مقدار طول‌گام روش گرادیان در شروع اجرا بیشترین مقدار را داشته باشد و با نزدیک شدن به جواب بهینه، کاهش یابد) بهترین نتایج روش گرادیان برای مقدار یکسان اولیه و نهایی طول‌گام برابر با ۱e-۰۹ حاصل می شود که چنانچه در شکل (۱۸) مشاهده می‌شود، به خوبی نتایج روش پیشنهادی و قابل مقایسه با آن نیست. البته اگر برای بهبود عملکرد روش گرادیان انتخاب نامتعارف مقدار طول‌گام انجام شود (ضریب طول‌گام روش گرادیان در شروع مقدار کمی داشته باشد و با نزدیک شدن به جواب بهینه افزایش یابد) بهترین نتایج روش گرادیان برای مقدار اولیه طول‌گام برابر با ۱e-۰۹و مقدار نهایی برابر با ۱e+۰۴ مطابق با شکل (۱۹) حاصل می-شود که قابل مقایسه با نتایج روش پیشنهادی است اما چنین انتخابی برای ضریب طول‌گام روش گرادیان کاملا نامتعارف است و ممکن نیست به ذهن طراح مسئله برسد.



شکل (۱۹): ارزیابی عملکرد الگوریتم گرادیان کلاسیک با انتخاب نامتعارف طول‌گام و الگوریتم گرادیان پیشنهادی توسط تابع محک F10

با توجه به شکل‌های (۸) تا (۱۹)، در روش گرادیان برای هر کدام از توابع محک، چندین بار روش اجرا می‌شود تا بتوان به صورت تجربی مقادیر مناسب طول‌گام روش گرادیان را تعیین کرد درحالی‌که در روش پیشنهادی بدون نیاز به تنظیم هیچ پارامتری، نتایج قابل قبولی برای همه توابع محک به دست آمده است.

در روش پیشنهادی علاوه بر اینکه سرعت همگرایی بیشتر و عملکرد مناسب‌تری نسبت به الگوریتم گرادیان کلاسیک داشته، بلکه مشکل جهت حرکت اشتباه روش پیشنهادی در برخی گام‌ها نیز برطرف شده است. در روش پیشنهادی طول‌گام حرکت به صورت خودکار تنظیم شده و نیازی به تنظیم دستی ندارد. این هوشمندسازی طول‌گام سبب شده تا سرعت بیشتری به الگوریتم گرادیان داده شود و همچنین باعث شده در تعداد گام‌های یکسان، روش پیشنهادی به جواب مناسب‌تری نسبت به روش گرادیان دست یابد. لازم به ذکر است که روش گرادیان می‌تواند به جواب بهینه اصلی (در صورت گیر نیفتادن در جواب بهینه محلی) برسد اما تعداد گام‌های زیادتری را نیاز دارد. همچنین علت بالاتر بودن سرعت همگرایی روش پیشنهادی نسبت به روش گرادیان کلاسیک، اصلاح جهت حرکت و تنظیم طول‌گام بهینه است که سبب شده روش پیشنهادی در تعداد گام‌های کمتر، به جواب بهتری دست یابد. در جدول (۴)، نتایج مربوط به بهترین جواب، بدترین جواب، میانگین جواب‌هایی که هر یک از روش‌های گرادیان و روش پیشنهادی در ۱۰۰ گام اجرا به آن همگرا شده‌اند و متوسط زمان اجرای هر گام هر یک از دو روش برحسب ثانیه نمایش داده شده است. به طور واضح‌تر می‌توان گفت که در ستون "بهترین جواب"، کمترین مقدار جوابی که هر روش در طول سی مرتبه اجرای مستقل به آن رسیده، در ستون "بدترین جواب"، بیشترین مقدار جوابی که هر روش در طول سی مرتبه اجرای مستقل به آن دست یافته، در ستون "میانگین جواب‌ها"، میانگین جوابی که هر روش در طول سی مرتبه اجرای مستقل به آن همگرا شده‌اند و در ستون "متوسط زمان اجرای هر گام (ثانیه)" نیز میانگین زمان اجرای هر گام روش‌ها در طول سی مرتبه اجرای مستقل هر روش بر حسب ثانیه ذکر شده است. لازم به ذکر است جهت به دست آوردن مقادیر فوق، هر دو روش در ۵۰۰ گام اجرا شده‌اند تا جواب بهینه به دست آید و نتایج موجود در جدول (۴)، لزوماً در ۱۰۰ گام حاصل نشده است. با توجه به نتایج ذکر شده در جدول (۴)، روش پیشنهادی در بهینه‌سازی همه توابع محک از لحاظ بهترین جواب، بدترین جواب و میانگین جواب‌هایی که در طول سی مرتبه اجرای مستقل به آن همگرا شده است، عملکرد بهتری نسبت به روش گرادیان کلاسیک از خود نشان داده است.

در برخی توابع محک، بهترین جوابی که روش گرادیان کلاسیک و روش پیشنهادی در طول سی مرتبه تکرار اجرا، به آن همگرا شده‌اند، نزدیک به یکدیگر می‌باشد درحالیکه میانگین اجرای روش پیشنهادی نسبت به روش گرادیان، به طرز قابل توجهی، مناسب‌تر بوده است که این موضوع نشان‌دهنده قابلیت اطمینان بالای روش پیشنهادی می‌باشد.

جدول (۴): ارزیابی عملکرد روش‌های پیاده‌سازی شده با بهینه‌سازی توابع محک

تابع	روش	بهترین جواب	بدترین جواب	میانگین جواب‌ها	متوسط زمان اجرای هر گام (ثانیه)
F_1	GD	-۴۷۵,۹۹۹۹	۱۴,۷۴۶۳	-۱۶۸,۱۹۵۳	۱/۵۲۶۶e-۰۵
	روش پیشنهادی	-۴۷۶	۲,۴۶۰۴	-۲۶۸,۶۶۷۲	۲/۲۳۶e-۰۵
F_2	GD	۳,۲۷۴۰e+۰۶	۱,۱۸۶۰e+۱۷	۳,۶۱۴۱e+۱۶	۱/۸۹۴۷e-۰۵
	روش پیشنهادی	۲,۰۵۹۸e-۰۴	۲,۷۵۰۶e+۱۵	۲,۱۹۵۵e+۱۰	۲/۲۶۵۷e-۰۵
F_3	GD	-۰,۹۵۰۴	۲,۲۶۳۴	۰,۸۴۱۰	۱/۸۱۵۶e-۰۵
	روش پیشنهادی	-۱,۰۳۱۶	۲,۱۰۴۳	۰,۲۳۱۳	۲/۲۷۳۱e-۰۵
F_4	GD	۲۰۴,۴۵۸۸	۸,۲۵۵۲e+۰۳	۲,۵۱۴۸e+۰۳	۱/۶۹۶۱e-۰۵
	روش پیشنهادی	۲۳,۹۹۹۴	۲۷,۳۶۱۱	۲۴,۷۹۴۰	۲/۰۲۱۶e-۰۵
F_5	GD	۱۱,۵۷۰۹	۱۵۴,۷۷۹۳	۶۴,۱۳۰۷	۱/۶۷۰۸e-۰۵
	روش پیشنهادی	۲,۱۱۰۴e-۰۵	۳,۰۰۶۸e-۰۴	۴,۲۳۵۶e-۰۵	۲/۳۳۳۹e-۰۵
F_6	GD	۰	۰	۰	۱/۲۲۹۹e-۰۵
	روش پیشنهادی	۰	۰	۰	۲/۱۸۹۴e-۰۵
F_7	GD	-۳۶۳,۳۸۸۲	-۲۷۸,۵۶۷۹	-۳۱۸,۱۹۸۲	۱/۷۷۰۰e-۰۵
	روش پیشنهادی	-۳۶۳,۳۸۸۲	-۲۷۸,۵۶۷۸	-۳۱۸,۶۲۱۸	۲/۴۴۰۶e-۰۵
F_8	GD	۲۰۴,۹۶۰۵	۳۵۹,۱۷۷۵	۲۶۸,۲۷۲۴	۱/۶۳۶۰e-۰۵
	روش پیشنهادی	۲۰۴,۹۶۱۸	۳۵۹,۱۸۹۲	۲۶۸,۲۷۷۸	۲/۲۳۰۸e-۰۵
F_9	GD	۰,۰۸۷۵	۰,۱۹۶۶	۰,۱۵۰۱	۱/۱۰۵۱e-۰۵
	روش پیشنهادی	۰,۰۲۲۲	۰,۰۵۰۸	۰,۰۳۷۲	۱/۹۴۵۷e-۰۵
F_{10}	GD	۱,۷۰۸۰e-۰۷	۱,۸۸۱۷e-۰۷	۱,۸۴۲۱e-۰۷	۲/۴۱۰۱e-۰۵
	روش پیشنهادی	۲,۴۷۵۸e-۰۶	۱,۹۳۹۴e-۰۵	۸,۲۵۴۴e-۰۶	۲/۹۸۶۵e-۰۵

۶- نتیجه‌گیری

در این مقاله با الهام از مزایای الگوریتم بهینه‌سازی ازدحام ذرات مانند تعداد محدود پارامترهای انتخابی و وجود بازه استاندارد جهت انتخاب این پارامترها و همچنین با توجه به مزیت پایین بودن زمان محاسباتی در الگوریتم گرادیان، یک الگوریتم گرادیان هوشمند ارائه شد که در آن، طول‌گام و جهت حرکت به صورت خودکار تنظیم می‌شود و نیازی به انتخاب فراپارامتر طول‌گام وجود ندارد. برای بررسی کارایی و ارزیابی عملکرد روش پیشنهادی، ده تابع محک مورد بهینه‌سازی قرار گرفته و نتایج به دست آمده از روش پیشنهادی با نتایج حاصله از الگوریتم گرادیان کلاسیک مورد بررسی و مقایسه قرار گرفته است. با بررسی نتایج شبیه‌سازی‌ها، مشاهده شد که در همه توابع محک، الگوریتم پیشنهادی عملکرد بهتری هم از لحاظ سرعت و هم از لحاظ رسیدن به جواب بهتر داشته است. یک چالش در عملکرد روش پیشنهادی وجود داشت به طوری که در بعضی از گام‌ها، جهت حرکت پیشنهادی با جهت گرادیان هم‌خوانی لازم را نداشت و زاویه بین آن دو از ۹۰ درجه بیشتر می‌شد. این مسئله سبب حرکت اشتباه و بالازدگی در تعدادی از

گام‌ها می‌شد که با استفاده از روش تصویر بردار، این مشکل برطرف شد. با شبیه‌سازی روش پیشنهادی برای بهینه کردن ده تابع محک، مشاهده شد که الگوریتم پیشنهادی در همه توابع محک عملکرد بهتری نسبت به الگوریتم گرادیان کلاسیک داشته است و علاوه بر اینکه سرعت همگرایی بیشتری نسبت به الگوریتم گرادیان کلاسیک داشته است، در همه موارد توانسته است به جواب مناسب‌تری برسد و تابع محک را بیشتر کاهش دهد.

ضمایم

بررسی همگرایی روش گرادیان و روش پیشنهادی:

در روش گرادیان، حرکت در خلاف جهت گرادیان تابع هزینه انجام می‌شود.

$$\begin{cases} X_{k+1} = X_k + \Delta x_k \\ \Delta x_k = \eta \nabla_x J(x_k) \end{cases} \quad (۲۰)$$

این حرکت در خلاف جهت گرادیان سبب تغییر تابع هزینه در هر گام می‌شود که تغییرات تابع هزینه در صورت کوچک بودن Δx مطابق زیر قابل بیان است:

$$\dot{J} = \frac{\partial J(x)}{\partial x}.\dot{x} \quad \Rightarrow \Delta J = \frac{\partial J(x)}{\partial x}.\Delta x \quad (۲۱)$$

در روش گرادیان کاهشی:

حرکت در خلاف جهت گرادیان تابع هزینه تضمین می‌کند که اگر طول‌گام به درستی انتخاب شود، الگوریتم گرادیان همواره در جهت کاهش تابع هزینه حرکت می‌کند. بنابراین می‌توان نوشت:

$$\begin{cases} \Delta x = -\eta \nabla J \\ \nabla J = \dfrac{\partial J(x)}{\partial x} \end{cases} \Rightarrow \Delta J = \nabla J.(-\eta \nabla J) = -\eta \nabla J^T \nabla J = -\eta \left\| \nabla J^2 \right\| < 0 \quad (۲۲)$$

در نتیجه $\Delta J < 0$ بوده یعنی حرکت با توجه به رابطه (۲۰) سبب می‌شود که تابع هزینه کاهش یابد. البته شرط کوچک بودن اندازه Δx در رابطه (۲۱) لازم است و این مسئله که $\Delta x = -\eta \nabla J$، اهمیت انتخاب طول‌گام η در همگرایی روش گرادیان را نشان می‌دهد. بنابراین روش گرادیان در صورت انتخاب مناسب طول‌گام η، همواره در جهت کاهش تابع هزینه حرکت می‌کند.

در روش پیشنهادی:

در روش پیشنهادی، $\Delta x_k = \overline{V}(\nabla J_k, \nabla J_{k-1}, ...)$ که لزوماً هم‌جهت $\nabla J(x_k)$ نیست. در این حالت تغییرات تابع هزینه به صورت زیر است:

$$\Delta J = \nabla J.\Delta x = \left|\nabla J\right| \left|\Delta x\right| \cos\theta \quad (۲۳)$$

با توجه به شکل (۵)، θ زاویه بین Δx و ∇J و α زاویه بین Δx و $-\nabla J$ است که $\alpha + \theta = 180^\circ$ می‌شود. حال اگر $|\theta| > 90^\circ$ باشد، بنابراین $\cos\theta < 0$ می‌شود و در نتیجه:

$$\left|\nabla J\right| \left|\Delta x\right| \cos\theta < 0 \Rightarrow \Delta J < 0 \quad (۲۴)$$

بنابراین صورتی $|\theta| > 90^\circ$ یا $|\alpha| < 90^\circ$ باشد، روش پیشنهادی سبب کاهش تابع هزینه می‌شود.

اما در روش پیشنهادی زاویه بین Δx و ∇J ثابت نیست و نمی‌توان شرط $|\theta| > 90^\circ$ را تضمین کرد. به همین دلیل، در روش پیشنهادی زاویه α در هر گام بررسی می‌شود و اگر $|\alpha| < 90^\circ$ یعنی $-90 < \alpha < 90$ بود، روش پیشنهادی اجرا شده و سبب کاهش تابع هزینه J می‌شود، در غیر این‌صورت از مسئله تصویر بردار مماسی جهت حرکت پیشنهادی به صورت رابطه (۱۱)-(۱۸) استفاده می‌شود که تضمین می‌کند تا زاویه α حتما کمتر یا مساوی ۹۰ درجه باشد.

مراجع

[1] O. A. Al-Shahri et al., "Solar photovoltaic energy optimization methods, challenges and issues: A comprehensive review", Journal of Cleaner Production, vol. 284, p. 125465, 2021.

[2] Nezamabadi-pour, Hossein. *Genetic Algorithm: Basic and Advanced Concepts*. Kerman: Shahid Bahonar University of Kerman, 2010.

[3] R. Bansal, "Optimization methods for electric power systems: An overview", International Journal of Emerging Electric Power Systems, vol. 2, no. 1, 2005.

[4] A. Mustapha, L. Mohamed, and K. Ali, "An overview of gradient descent algorithm optimization in machine learning: Application in the ophthalmology field", in Smart Applications and Data Analysis: Third International Conference, Morocco, 2020, pp. 349-359.

[5] T.T., Trung, H. T., Nguyen. "Backtracking gradient descent method and some applications in large scale optimisation. Part 2: Algorithms and experiments", Applied Mathematics & Optimization, Vol. 84, no. 3, pp. 2557-2586, 2021.

[6] Y., Lei, K., Tang. "Learning rates for stochastic gradient descent with nonconvex objectives", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 43, no. 12, pp. 4505-4511, 2021.

[7] Ghiassirad H, Aliyari Shoorehdeli M, Farivar F. "To Analysis and Compare 21 Weight Constraints in Stochastic Gradient Descent Algorithm Using Kernel Method", Journal of Iranian Association of Electrical and Electronics Engineers 2022; 19 (3) :145-152. URL: http://jiaeee.com/article-1-1276-fa.html

[8] S. Ruder, "An overview of gradient descent optimization algorithms", arXiv preprint arXiv:1609.04747, 2016.

[9] M. D. Zeiler, "Adadelta: an adaptive learning rate method", arXiv preprint arXiv:1212.5701, 2012.

[10] Z. Fu, S.-C. Chu, J. Watada, C.-C. Hu, and J.-S. Pan, "Software and hardware co-design and implementation of intelligent optimization algorithms", Applied Soft Computing, vol. 129, p. 109639, 2022.

[11] C. Blum, "Ant colony optimization: Introduction and recent trends", Physics of Life reviews, vol. 2, no. 4, pp. 353-373, 2005.

[12] E. Kaya, B. Gorkemli, B. Akay, and D. Karaboga, "A review on the studies employing artificial bee colony algorithm to solve combinatorial optimization problems", Engineering Applications of Artificial Intelligence, vol. 115, p. 105311, 2022.

[13] M. G. Sahab, V. V. Toropov, and A. H. Gandomi, "A review on traditional and modern structural optimization: problems and techniques", Metaheuristic applications in structures and infrastructures, pp. 25-47, 2013.
[14] A. G. Hussien et al., "Crow search algorithm: theory, recent advances, and applications", IEEE Access, vol. 8, pp. 173548-173565, 2020.
[15] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer", Advances in engineering software, vol. 69, pp. 46-61, 2014.
[16] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: a gravitational search algorithm", Information sciences, vol. 179, no. 13, pp. 2232-2248, 2009.
[17] H. Zhang, J. Li, M. Hong, and Y. Man, "Artificial intelligence algorithm-based multi-objective optimization model of flexible flow shop smart scheduling", Applications of Artificial Intelligence in Process Systems Engineering, pp. 447-472, 2021.
[18] B. A. S. Emambocus, M. B. Jasser, and A. Amphawan, "A survey on the optimization of artificial neural networks using swarm intelligence algorithms", IEEE Access, vol. 11, pp. 1280-1294, 2023.
[19] M. Jain, V. Saihjpal, N. Singh, and S. B. Singh, "An Overview of Variants and Advancements of PSO Algorithm", Applied Sciences, vol. 12, no. 17, p. 8392, 2022.
[20] J. Kennedy and R. Eberhart, "Particle swarm optimization", in Proceedings of ICNN'95-international conference on neural networks, 1995, vol. 4: IEEE, pp. 1942-1948.
[21] L. Zhen, Y. Liu, W. Dongsheng, and Z. Wei, "Parameter estimation of software reliability model and prediction based on hybrid wolf pack algorithm and particle swarm optimization", IEEE Access, vol. 8, pp. 29354-29369, 2020.
[22] R. Padmanabhan and P. Seiler, "Analysis of Gradient Descent with Varying Step Sizes using Integral Quadratic Constraints", arXiv preprint arXiv:2210.00644, 2022.
[23] M. Bazaraa, H. Sherali, and C. Shetty, Nonlinear theory and algorithm, 3rd ed. USA: Wiley-Interscience, 2006.
[24] S. Biswas, S. Nath, S. Dey, and U. Majumdar, "Tangent-cut optimizer on gradient descent: an approach towards Hybrid Heuristics", Artificial Intelligence Review, pp. 1-27, 2022.
[25] E. Taş and M. MEMMEDLİ, "Near optimal step size and momentum in gradient descent for quadratic functions", Turkish Journal of Mathematics, vol. 41, no. 1, pp. 110-121, 2017.
[26] S. Guan and B. Biswal, "Spline adaptive filtering algorithm based on different iterative gradients: Performance analysis and comparison", Journal of Automation and Intelligence, vol. 2, no. 1, pp. 1-13, 2023.
[27] Z. Gu, K. Jin, Y. Meng, L. Xue, and L.-H. Zhang, "On Kahan's automatic step-size control and an anadromic gradient descent iteration", Technical report, 2022, URL https://www. researchgate. net/publication/362126629_ On_Kahan's_automatic_ step-size_control_and_an_ anadromic_gradient_descent_iteration, 2022.
[28] A. Soodabeh and V. Manfred, "A learning rate method for full-batch gradient descent", Műszaki Tudományos Közlemények, vol. 13, no. 1, pp. 174-177, 2020.
[29] Y. Qiao, B. van Lew, B. P. Lelieveldt, and M. Staring, "Fast automatic step size estimation for gradient descent optimization of image registration", IEEE transactions on medical imaging, vol. 35, no. 2, pp. 391-403, 2015.
[30] Y. Xue, Y. Wang, and J. Liang, "A self-adaptive gradient descent search algorithm for fully-connected neural networks", Neurocomputing, vol. 478, pp. 70-80, 2022.
[31] R. Padmanabhan and P. Seiler, "Analysis of Gradient Descent with Varying Step Sizes using Integral Quadratic Constraints", arXiv preprint arXiv : 2210.00644, 2022.
[32] M. Ravaut and S. Gorti, "Gradient descent revisited via an adaptive online learning rate", arXiv preprint arXiv:1801.09136, 2018.
[33] K. Zeng, J. Liu, Z. Jiang, and D. Xu, "A decreasing scaling transition scheme from Adam to SGD", Advanced Theory and Simulations, vol. 5, no. 7, p. 2100599, 2022.
[34] X. Wei and H. Huang, "A survey on several new popular swarm intelligence optimization algorithms", Research Square, 2023, doi: 10.21203/rs.3.rs-2450545/v1.
[35] F. Parandeh Motlagh, V. Khatibi Bardsiri, and A. Khatibi Bardsiri, "Human-Whale cooperation optimization (HWO) algorithm: A metaheuristic algorithm for solve optimization problems", International Journal of Nonlinear Analysis and Applications, vol. 14, no. 1, pp. 2279-2300, 2023.
[36] P. K. Gupta, B. Yadav, A. Kumar, and S. K. Himanshu, "Machine learning and artificial intelligence application in constructed wetlands for industrial effluent treatment: advances and challenges in assessment and bioremediation modeling", Bioremediation for Environmental Sustainability, pp. 403-414, 2021.
[37] N. H. Phong, A. Santos, and B. Ribeiro, "PSO-convolutional neural networks with heterogeneous learning rate", IEEE Access, vol. 10, pp. 89970-89988, 2022.
[38] D. A. Ejigu and X. Liu, "Gradient descent-particle swarm optimization based deep neural network predictive control of pressurized water reactor power", Progress in Nuclear Energy, vol. 145, p. 104108, 2022.
[39] S. Liu and X. Huang, "Multi- Objective Optimization for the Cross Brace of a Computer Numerical Control Gantry Machine Tool Based on Intelligent Algorithms", Advanced Intelligent Systems, vol. 5, no. 3, p. 2200368, 2023.
[40] A. Eleyan, M. S. Salman, and B. Al-Sheikh, "Application of optimization algorithms for classification problem", International Journal of Electrical and Computer Engineering, vol. 12, no. 4, p. 4373, 2022.
[41] A. Mehboodi, H. Nezamabadi-pour, M. Soleimanpour-moghadam, "Multi-robot Path Planning in a 3D Environment by Modified Particle Swarm Optimization Algorithm", Journal of Iranian Association of Electrical and Electronics Engineers 2022; 19 (3) :163-174. URL: http://jiaeee.com/article-1-1073-en.html
[42] M. Vakilifard, A. Sahafi, A. M. Rahmani, P. Sheikholharam Mashhadi, "FRA-PSO: A two-stage Resource Allocation Algorithm in Cloud Computing", Journal of Iranian Association of Electrical and Electronics Engineers 2023; 20 (1) :43-49. URL: http://jiaeee.com/article-1-1237-fa.html
[43] M. P. Deisenroth, A. A. Faisal, and C. S. Ong, Mathematics for machine learning. Cambridge University Press, 2020.
[44] F. A. Hashim, E. H. Houssein, K. Hussain, M. S. Mabrouk, and W. Al-Atabany, "Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems", Mathematics and Computers in Simulation, vol. 192, pp. 84-110, 2022.
[45] M. J. Goldanloo and F. S. Gharehchopogh, "A hybrid OBL-based firefly algorithm with symbiotic organisms

search algorithm for solving continuous optimization problems", The Journal of Supercomputing, vol. 78, no. 3, pp. 3998-4031, 2022.

[46] M. Dehghani, Š. Hubálovský, and P. Trojovský, "Tasmanian devil optimization: a new bio-inspired optimization algorithm for solving optimization algorithm", IEEE Access, vol. 10, pp. 19599-19620, 2022.

زیرنویس‌ها

[1] Cost Function
[2] Profit Function
[3] Saddle Points
[4] Hyper-Parameters
[5] Step Size
[6] Ant Colony Optimization (ACO)
[7] Artificial Bee Colony (ABC)
[8] Particle Swarm Optimization (PSO)
[9] Crow Search Algorithm (CSA)
[10] Grey Wolf Optimizer (GWO)
[11] Gravitational Search Algorithm (GSA)
[12] Genetic Algorithm (GA)
[13] Simulated Annealing (SA)
[14] Sine Cosine Algorithm (SCA)
[15] Immune Algorithm (IA)
[16] Tabu Search (TS)
[17] Variable Neighborhood Search (VNS)
[18] Valley
[19] Momentum
[20] Robbins-Monro
[21] Kesten
[22] Gaivoronski
[23] Lipschitz
[24] Benchmark Functions
[25] MATLAB
[26] Fitness