مجله محاسبات نرم، جلد 14، شماره 1 (بهار و تابستان 1404)، صفحه 184ـ203
https://doi.org/10.22052/scj.2025.254888.1244

مجله محاسبات نرم
SOFT COMPUTING JOURNAL
تارنمای مجله: scj.kashanu.ac.ir

دانشگاه کاشان
University of Kashan

حل عددی دستگاه معادلات غیرخطی با الگوریتم فراابتکاری ARO بهبودیافته✧

علی حمدی‌پور[1]، دانشجوی دکتری، عبدالعلی بصیری[1*]، دانشیار، مصطفی زارع خورمیزی[1]، استادیار

[1] دانشکده ریاضی و علوم کامپیوتر، دانشگاه دامغان، دامغان، ایران.

اطلاعات مقاله

تاریخچه مقاله:
دریافت 16 اردیبهشت ماه 1403
پذیرش 1 آبان ماه 1403

کلمات کلیدی:
دستگاه معادلات غیرخطی
آنالیز عددی
الگوریتم‌های فراابتکاری
الگوریتم ARO
بهینه‌سازی

چکیده

حل دستگاه معادلات غیرخطی یکی از سخت‌ترین مسائل در محاسبات عددی است. روش‌های عددی سنتی مانند روش‌های نیوتن و انواع آن نیاز به حدس اولیه خوب برای حل دستگاه معادلات غیرخطی دارند. حدس اولیه نامناسب می‌تواند تاثیر سوء در عملکرد و همگرایی این روش‌ها داشته باشد. در عمل، دست‌یابی به این حدس اولیه دشوار و از نظر زمانی پرهزینه خواهد بود. با هدف غلبه بر این مشکل، در این مقاله بهره‌گیری از الگوریتم فراابتکاری بهبودیافته (IARO) برای حل عددی دستگاه معادلات غیرخطی پیشنهاد شده است. از آنجا که حل دستگاه معادلات غیرخطی را می‌توان به حل یک مساله بهینه‌سازی تقلیل داد، الگوریتم فراابتکاری توانایی خوبی در پیدا کردن جواب آن خواهد داشت. الگوریتم فراابتکاری ARO از رفتار خرگوش‌ها در هنگام تغذیه الگو گرفته است و می‌تواند مسائل بهینه‌سازی پیچیده را در زمان مناسب حل کند. در روش پیشنهاد شده، الگوریتم ARO به کمک جدول حافظه بهبود یافته تا عملکرد مناسبی برای حل دستگاه معادلات غیرخطی داشته باشد. برای سنجش عملکرد روش پیشنهاد شده، جواب چندین دستگاه معادلات غیرخطی پیچیده توسط آن محاسبه شده که نتایج آن عملکرد خوب روش پیشنهادی را نمایش می‌دهد.

1. مقدمه

حل دستگاه‌های معادلات غیرخطی یکی از مهم‌ترین مسائل در محاسبات عددی به ویژه در طیف متنوعی از کاربردهای مهندسی است. بسیاری از مسائل کاربردی را می‌توان به حل دستگاه معادلات غیرخطی تقلیل داد که یکی از اساسی‌ترین مسائل در ریاضیات است و در بسیاری از زمینه‌های علمی‌کاربرد دارد [1]، [2]. بنابراین تلاش‌های زیادی برای حل کارای دستگاه معادلات غیرخطی توسط افراد مختلفی انجام شده است و نظریه‌ها و الگوریتم‌های مناسب زیادی برای حل دستگاه‌های معادلات غیرخطی پیشنهاد شده‌اند [3]. با این حال هنوز مسائلی در حل دستگاه‌های معادلات غیرخطی وجود دارد که برای اکثر روش‌های عددی سنتی مانند روش نیوتن، همگرایی و عملکرد مناسب می‌تواند به حدس اولیه راه‌حل بسیار حساس باشد. همچنین، انتخاب حدس اولیه معقول برای حل اکثر معادلات غیرخطی بسیار دشوار است. اگر حدس اولیه جواب نامناسب باشد، الگوریتم شکست خواهد خورد یا نتایج نامناسبی را به

✧ نوع مقاله: پژوهشی
* نویسنده مسئول
پست(های) الکترونیک: hamdipour.ali@std.du.ac.ir (حمدی‌پور)
basiri@du.ac.ir (بصیری)
mostafazaare@du.ac.ir (زارع خورمیزی)

نحوه ارجاع به مقاله: علی حمدی‌پور، عبدالعلی بصیری، مصطفی زارع خورمیزی، «حل عددی دستگاه معادلات غیرخطی با الگوریتم فراابتکاری ARO بهبودیافته»، مجله محاسبات نرم، جلد 14، شماره 1، ص184ـ203، بهار و تابستان 1404.

دست خواهد آورد [4].

بسیاری از ترکیب‌های مختلف از روش‌های عددی سنتی و الگوریتم‌های هوشمند برای حل دستگاه‌های معادلات غیرخطی استفاده می‌شوند که می‌تواند بر مشکل انتخاب حدس اولیه مناسب برای جواب غلبه کند [5]، اما زمانی که تعداد زیادی معادله در دستگاه معادلات غیرخطی برای حل وجود دارد، الگوریتم‌های موجود برای محاسبه بسیار پیچیده یا زمان‌بر هستند. همچنین بسیاری از الگوریتم‌های هوشمند بهبود یافته، مانند الگوریتم ازدحام ذرات[1] و الگوریتم ژنتیک[2]، برای حل دستگاه‌های معادلات غیرخطی پیشنهاد شده‌اند. اگرچه آنها بر مشکل انتخاب حدس اولیه معقول برای جواب غلبه می‌کنند، اما فاقد توانایی جستجوی پیچیده در منطقه محلی هستند، که ممکن است منجر به رکود همگرایی شود [6].

در طول چند دهه گذشته، رویکردهای بهینه‌سازی متعددی برای مقابله با مسائل بهینه‌سازی طراحی شده است. با این حال، در سال‌های اخیر، پیچیدگی مسائل بهینه‌سازی در دنیای واقعی به طور قابل ملاحظه‌ای با توسعه جامعه انسانی و فرآیندهای صنعت مدرن افزایش پیدا کرده است، که چالشی فزآینده برای تکنیک‌های بهینه‌سازی ایجاد می‌کند و باعث لزوم بهره‌گیری از روش‌های جدید برای حل این دستگاه‌ها می‌شود.

به طور کلی، تکنیک‌های بهینه‌سازی موجود را می‌توان به روش‌های قطعی و فراابتکاری دسته‌بندی کرد. الگوریتم‌های قطعی توابع ریاضی خاصی هستند و به صورت قطعی، تکراری و بدون هیچ‌گونه ماهیت تصادفی کار می‌کنند. در یک مساله معین، یک روش قطعی همیشه خروجی یکسانی را برای یک ورودی خاص به دست می‌آورد. گرادیان کاهشی[3] و روش‌های نیوتن دو دو نمونه کلاسیک از الگوریتم‌های قطعی هستند. اگرچه چنین الگوریتم‌هایی می‌توانند به طور موثر بهینه جهانی[4] را در حل برخی مسائل غیرخطی پیدا کنند، اما ممکن است به اطلاعات مشتق از مسائل نیاز داشته باشند و یا ممکن است در جواب‌های بهینه محلی به دام بیفتند. بنابراین، این روش‌ها هنگام برخورد با مسائل دارای قیود و پیچیده با پیک‌های متعدد، ناتوان هستند [7]. با توجه به موارد فوق، روش‌های فراابتکاری ظهور کرده‌اند تا جایگزین‌های خوبی برای روش‌های قطعی شوند. چنین روش‌هایی از عملگرهای مختلف به طور مکرر برای کاوش و بهره‌برداری از فضای جستجو براساس یک تابع حداقل‌سازی یا حداکثرسازی استفاده می‌کنند [8]. این الگوریتم‌ها می‌توانند بین بهره‌برداری[5] و اکتشاف[6] تعادل برقرار کنند [9]. در دو دهه گذشته، روش‌های فراابتکاری به شدت مورد بررسی قرار گرفته‌اند و می‌توانند در زمینه‌های مختلف اقتصاد و تجارت، مالی، انرژی، برنامه‌ریزی، پردازش تصویر، کنترل بهینه و طراحی مهندسی و غیره مورد استفاده قرار گیرند [10].

روش‌های فراابتکاری به دلیل تصادفی بودن و در نظر گرفتن مسائل در جعبه سیاه بسیار محبوب هستند. ماهیت تصادفی بودن روش‌های فراابتکاری، حساسیت مساله را نسبت به شرایط اولیه و جابجایی بین اکتشاف و بهره‌برداری کمتر می‌کند. ماهیت جعبه سیاه بودن، به ما این امکان را می‌دهد که بدون دانش ساختاری داخل الگوریتم فراابتکاری، روی ورودی و خروجی تمرکز کنیم. این شایستگی‌ها روش‌های فراابتکاری را قادر می‌سازد تا به طور موثر جواب بهینه جهانی را برای مسائل معینی که روش‌های قطعی به دلیل فقدان مشتق یا اطلاعات مرتبط دیگر نمی‌توانند حل کنند، بیابند. در شکل (1) طبقه‌بندی الگوریتم‌های فراابتکاری براساس ساختار عملکرد آنها آورده شده است [19].

در میان روش‌های فراابتکاری، الگوریتم‌های الهام‌گرفته از طبیعت بیشترین شتاب را در توسعه در سال‌های اخیر به دست آورده‌اند و به طور فزآینده‌ای برای مسائل مهندسی مختلف با مزایای زیادی استفاده می‌شوند [20]. الگوریتم‌های الهام گرفته از طبیعت به طور معمول فعالیت‌های بیولوژیکی موجودات زنده را تقلید می‌کنند و آنها را به مدل‌های ریاضی و روشی برای بهینه‌سازی تبدیل می‌کنند. تعداد زیادی از موجودات زنده مختلف در

[1] Particle swarm optimization
[2] Genetic algorithm
[3] Descent Gradient
[4] Global optimum

[5] Exploration
[6] Exploitation

طبیعت وجود دارند که برخی از آنها همیشه برای توسعه الگوریتم‌های بهینه‌سازی موثر به ما الهام می‌دهند. الگوریتم ژنتیک (GA)، یکی از الگوریتم‌های تکاملی[1] اولیه است و از دسته الگوریتم‌های بیولوژیکی می‌باشد که از انتخاب طبیعی برای حل مسائل پیچیده بهینه‌سازی بهره می‌گیرد [21]. نسخه اصلی GA از سه رفتار تکاملی تقلید می‌کند: انتخاب[2]، ترکیب[3] و جهش[4]. الگوریتم ژنتیک بر اساس جمعیتی از افراد عمل می‌کند که هر یک از آنها یک جواب کاندید را نشان می‌دهد. جمعیت با این سه عمل‌گر در طول زمان تکامل می‌یابد، و بهترین فرد برای تولید جمعیت جدید پس از تکرارهای متعدد به کار گرفته می‌شود. با توجه این‌که افراد متناسب با مقادیر تابع برازش[5] آنها انتخاب می‌شوند الگوریتم ژنتیک به بهینه جهانی همگرا می‌شود. GA نسبت به روش‌های قطعی در برخی موارد قوی‌تر است و نیازی به اطلاعات اضافی در مورد مسائل مورد نظر ندارد. این الگوریتم ظرفیت بهینه‌سازی خوبی را برای حل مسائل پیچیده نشان می‌دهد، با این حال، سرعت همگرایی این الگوریتم کم است. همچنین، عملکرد مناسب آن بیشتر به انتخاب نرخ ترکیب، جهش، انتخاب توابع برازش و اندازه جمعیت بستگی دارد [22].

الگوریتم‌های فراابتکاری	
الگوریتم‌های تکاملی	از جمله: الگوریتم‌ها ژنتیک، الگوریتم دیفرانسیلی تکاملی [11]، استراتژی تکاملی [12]
الگوریتم‌های بر پایه جمعیت	از جمله: الگوریتم ازدحام ذرات، الگوریتم کلونی مورچه‌ها، الگوریتم کلونی زنبور [13]
الگوریتم‌های بر پایه فیزیک	از جمله: الگوریتم جستجوی گرانشی [14]، الگوریتم جستجوی سیستم شارژ شده [15]
الگوریتم‌های بر پایه زیست	از جمله: کرم شب تاب [16]، الگوریتم فاخته [17]، الگوریتم بهینه‌سازی تغذیه باکتری [18]

شکل (1): طبقه‌بندی الگوریتم‌های فراابتکاری

[1] Evolutionary algorithms
[2] Selection
[3] Crossover
[4] Mutation
[5] Fitness function

بهینه‌سازی ازدحام ذرات (PSO) نیز یکی از محبوب‌ترین روش‌های الهام‌گرفته از طبیعت است که رفتارهای اجتماعی دسته پرندگان را تقلید می‌کند [23]. این فرآیند با جمعیتی از افراد تصادفی شروع می‌شود که موقعیت‌های آنها راه حلی برای یک مساله در نظر گرفته می‌شود. در هر تکرار این الگوریتم، موقعیت هر ذره به طور تصادفی براساس بهترین موقعیت جهانی به دست آمده و سپس موقعیت هر ذره و بهترین موقعیت برای خود ذره به روز می‌شود. تابع برازش برای اندازه‌گیری کیفیت یک ذره استفاده می‌شود. اگرچه PSO نرخ همگرایی خوبی دارد، اما به راحتی در بهینه محلی برای برخی مسائل با ابعاد بالا به دام می‌افتد و همچنین به نسبت به پارامترهای کنترلی خود حساس است [24]، [25].

بهینه‌سازی کلونی مورچه‌ها (ACO)[6]، یکی دیگر از الگوریتم‌های محبوب الهام گرفته شده از طبیعت است که رفتار اجتماعی مورچه‌ها را هنگام جستجوی غذا مدل می‌کند [26]. هنگامی که جمعیتی از مورچه‌ها فضای جستجو را کاوش می‌کنند، مسیرهای فرومونی[7] را برای هدایت یکدیگر به سمت اهداف می‌گذارند. هدف هر مورچه جستجوی کوتاه‌ترین مسیر بین لانه و منبع غذا از طریق مسیرهای فرومون است. اگر مورچه‌ای مسیر کوتاه‌تری پیدا کند، بقیه مورچه‌ها این مسیر را دنبال می‌کنند تا زمانی که بازخورد مثبت همه مورچه‌ها را به دنبال کردن یک مسیر هدایت کند. ACO با موفقیت برای حل مسائل مسیریابی وسیله نقلیه و سایر کاربردهای دینامیکی استفاده می‌شود، اما تجزیه و تحلیل نظری آن دشوار است و زمان همگرایی آن هنگام حل مسائل نامشخص است [27].

الگوریتم‌های مختلف دیگری که از طبیعت الهام گرفته اند در سال‌های اخیر معرفی شده اند که می‌توان به الگوریتم خفاش[8] (BA) [28]، بهینه‌سازی ازدحام کرم درخشنده[9] (BOA) [29]، بهینه‌ساز گرگ خاکستری[10] (GWO) [30]، بهینه‌سازی پروانه[11]

[6] Ant colony optimization
[7] Pheromone routes
[8] Bat algorithm
[9] Glowworm swarm optimization
[10] Gray wolf optimization
[11] Butterfly optimization algorithm

(BOA) [31] و نفرون-2[1] (NOA-2) [32] اشاره کرد.

یکی از الگوریتم فراابتکاری که از رفتار خرگوش‌ها هنگام تغذیه تقلید می‌کند، الگوریتم ARO[2] است این الگوریتم می‌تواند جواب مسائل بهینه‌سازی را در زمان مناسب به دست آورد [33]. در این مقاله الگوریتم تکاملی ARO برای حل دستگاه‌های معادلات غیرخطی بهبود داده شده است و نشان داده شده که می‌تواند جواب دستگاه معادلات غیرخطی را با دقت قابل قبول پیدا کند. همچنین این روش می‌تواند بر وابستگی به حدس اولیه منطقی راه حل غلبه کند و کارایی محاسباتی را بهبود بخشد.

ادامه مقاله به شرح زیر است: در بخش 2 به صورت خلاصه الگوریتم ARO شرح داده می‌شود. در بخش 3 ایده اصلی مقاله آورده شده است. در بخش 4 نتایج و آزمایش‌های صورت گرفته برای مقایسه مقاله آورده شده است. در بخش 5 نیز نتیجه‌گیری مقاله آورده شده است.

2. الگوریتم ARO

خرگوش‌ها در طبیعت رفتارهای خاصی برای تغذیه، فرار یا پنهان شدن از دست شکارچیان دارند. ARO الگوریتمی است که بر اساس این رفتارهای خرگوش‌ها طراحی شده است. خرگوش‌ها برای جلوگیری از تشخیص محل لانه خود، از علف‌های نزدیک لانه خود تغذیه نمی‌کنند. این رفتار که تغذیه انحرافی[3] نامیده می‌شود، برای مرحله اکتشاف الگوریتم ARO استفاده می‌شود. همچنین، خرگوش‌ها حفره‌هایی را در کنار لانه‌های خود حفر می‌کنند تا با حمله شکارچیان به آنها، به صورت تصادفی در یکی از آنها پنهان شوند. این استراتژی که پنهان شدن تصادفی[4] نامیده می‌شود، در مرحله بهره‌برداری برای الگوریتم ARO استفاده می‌شود. با توجه به این‌که انرژی مورد نیاز برای فرار از دست شکارچیان در خرگوش‌ها محدود است، بنابراین باید بین تغذیه انحرافی و مخفی شدن تصادفی تعادل ایجاد کنند زیرا هر چه از لانه خود دورتر باشند، برای فرار باید مسافت بیشتری را طی کنند تا هنگام حمله شکارچیان پنهان شوند. ARO از این الگو برای ایجاد تعادل بین مراحل اکتشاف و بهره‌برداری استفاده می‌کند.

1.2. مدل‌سازی ریاضی رفتار خرگوش‌ها

الگوریتم ARO عوامل جستجو (خرگوش‌ها) را در فضای d بعدی در نظر می‌گیرد و سپس موقعیت هر یک از عامل‌ها را با توجه به رفتارهای ذکر شده به روز می‌کند. یکی از رفتارهای خرگوش‌ها، تغذیه انحرافی است. برای شبیه‌سازی ریاضی این رفتار از معادلات (1) تا (6) استفاده می‌شود.

$$\vec{v}_i(t+1) = \vec{x}_j(t) + R.\left(\vec{x}_i(t) - \vec{x}_j(t)\right) + round\left(0.05(0.05r_1)\right).n_1 \quad (1)$$

$$i,j = 1,\dots,n \quad \text{and} \quad j \neq i$$

$$R = L \cdot c \quad (2)$$

$$L = (e - e^{(\frac{t+1}{T})^2}) \cdot sin(2\pi r_2) \quad (3)$$

$$c(k) = \begin{cases} 1 & if \quad k = g(l) \\ 0 & else \end{cases} \quad (4)$$

$$k = 1,\dots,d \quad and \quad l = 1,\dots,\lceil r_3, d \rceil$$

$$g = randperm(d) \quad (5)$$

$$n_1 \sim N(0,1) \quad (6)$$

که در آن $\vec{v_i}(t+1)$ یک متغیر موقت برای به‌روزرسانی موقعیت هر عامل جستجو (خرگوش) در فضای جستجو است. $\vec{x_i}(t)$، n و T به ترتیب موقعیت i-امین عامل جستجو (خرگوش) در زمان t، اندازه جمعیت خرگوش‌ها و حداکثر تعداد تکرار هستند. همچنین n_1، n_2 و n_3 اعداد تصادفی در محدوده (0,1) هستند. n_1 توزیع نرمال استاندارد است. تابع $randperm(k)$، جایگشت تصادفی اعداد صحیح بین 0 و k را می‌دهد. رابطه (1) تغذیه انحرافی را فراهم می‌کند.

در رابطه (2)، R با l رابطه مستقیم دارد. همچنین l که طول گام‌های خرگوش است، در ابتدا مقدار زیادی دارد و به تدریج

[1] Nephron Algorithm Optimization 2
[2] Artificial rabbits optimization
[3] Detour foraging
[4] Random hiding

کاهش می‌یابد. رفتار تغذیه انحرافی باعث می‌شود که الگوریتم ARO در نقاط بهینه محلی گیر نکند و به جواب جهانی همگرا شود.

همان‌طور که در ابتدای این بخش ذکر شد، یکی دیگر از استراتژی‌های خرگوش‌ها، پنهان شدن تصادفی است. خرگوش‌ها هنگام حمله شکارچیان در لانه‌هایی که در اطراف لانه خود حفر می‌کنند پنهان می‌شوند. برای شبیه‌سازی این رفتار، الگوریتم ARO تعداد d حفره در هر بعد فضای جستجو فرض می‌کند. برای تولید این حفره‌ها از روابط (7) تا (10) استفاده می‌شود.

$$\vec{b}_{i,j}(t) = \vec{x}_i(t) + H.g.\vec{x}_i(t) \quad i = 1,\dots,n \quad and \quad j = 1,\dots,d \tag{7}$$

$$H = \frac{T - t + 1}{T} \cdot r_4 \tag{8}$$

$$n_2 \sim N(0,1) \tag{9}$$

$$g(k) = \begin{cases} 1 & if \;\; k = j \\ 0 & else \end{cases} \quad k = 1,\dots,d \quad \text{and} \quad l = 1,\dots,\lceil r_3, d \rceil \tag{10}$$

که در آن $\overrightarrow{b_{i,j}}(t)$ برابر j-امین حفره برای i-امین خرگوش است. r_4 و n_2 به ترتیب اعداد تصادفی در محدوده (0,1) و توزیع نرمال استاندارد هستند. برای شبیه‌سازی پنهان شدن تصادفی خرگوش‌ها در حفره‌های تولید شده از رابطه (11) استفاده می‌شود.

$$\vec{v}_i(t+1) = \vec{x}_i(t) + R \cdot \left(r_5 \cdot \vec{b}_{i,j}(t) - \vec{x}_i(t)\right) \quad i = 1,\dots,n \tag{11}$$

که در آن r_5 عددی تصادفی در محدوده (0,1) است. پس از انتخاب یکی از استراتژی‌های تغذیه انحرافی یا مخفی شدن تصادفی و محاسبه مقدار مربوطه v_i، از رابطه زیر برای محاسبه موقعیت بعدی هر خرگوش استفاده می‌شود.

$$\vec{x}_i(t+1) = \begin{cases} \vec{x}_i(t) & f(\vec{x}_i(t)) \le f(\vec{v}_i(t+1)) \\ \vec{v}_i(t+1) & f(\vec{x}_i(t)) > f(\vec{v}_i(t+1)) \end{cases} \tag{12}$$

که در این رابطه $x_i(t+1)$ موقعیت i-امین خرگوش را در زمان $t+1$ نشان می‌دهد. الگوریتم ARO کاهش انرژی خرگوش‌ها را در حین دویدن مدل کرده تا بین فازهای اکتشاف و بهره‌برداری جابجا شود. برای انجام آن در تکرارهای اولیه باید تغذیه انحرافی بیشتری انجام شود و به تدریج تغذیه انحرافی کاهش یافته و پنهان شدن تصادفی افزایش می‌یابد. در هر تکرار ARO برای انتخاب استراتژی مورد نظر، از معادله زیر استفاده می‌شود.

$$A(t) = 4\left(1 - \frac{t}{T}\right) ln\frac{1}{r} \tag{13}$$

که در آن r یک عدد تصادفی در محدوده (0,1) است. شبه کد الگوریتم ARO در شکل (2) آورده شده است. برای بررسی بیشتر این الگوریتم می‌توانید از مرجع [33] بهره ببرید. در بخش بعدی این الگوریتم بهبود یافته و برای حل دستگاه معادلات غیرخطی مورد استفاده قرار گرفته است.

3. روش پیشنهادی

در این بخش بهبود الگوریتم ARO برای حل دستگاه معادلات غیرخطی شرح داده شده است. برای بهبود این الگوریتم جدول حافظه[1] به این الگوریتم اضافه شده است. هدف این جدول انتخاب هوشمندانه اندیس[2] j در رابطه (1) است. در الگوریتم ARO اندیس j به صورت تصادفی انتخاب می‌شود و هر بار عامل جستجو با توجه به یک عامل جستجوی تصادفی موقعیت خود را به روز می‌کند. در روش پیشنهادی اندیس j با توجه با جدول حافظه انتخاب می‌شود که منجر به بهبود عملکرد ARO می‌شود.

این جدول یک ماتریس دو بعدی با ابعاد $n \times n$ است که در آن n تعداد عامل جستجو (جمعیت) الگوریتم فرااابتکاری است. نحوه کار این جدول به این صورت است که در شروع الگوریتم، تمام عناصر جدول برابر صفر قرار می‌گیرد. برای بروزسانی موقیت عامل جستجوی i-ام در رابطه (1) در هر تکرار

[1] Memory table (MT)
[2] Index

الگوریتم، اندیس j از روش پیشنهادی به دست می‌آید. در این روش اندیس j برابر با شماره ستون بیشترین مقدار در ردیف i می‌باشد. اگر چند مورد بیشترین مقدار در یک ردیف وجود داشت، از بین آنها عاملی انتخاب می‌شود که مقدار تابع برازش کمتری دارد. پس از تعیین شدن اندیس j، عنصر i-ام با توجه به آن موقعیت خود را در رابطه (1) بروز می‌کند.

پس از مشخص شدن اندیس j، در عناصر جدول حافظه بروزرسانی‌هایی به شکل زیر صورت می‌گیرد. در جدول حافظه مقدار عنصر واقع در سطر i-ام و ستون j-ام برابر با صفر می‌شود و به دیگر عناصر در ردیف مربوطه (i-ام) یک واحد افزوده می‌شود. همچنین اگر مقدار تابع برازش جدید محاسبه شده ($F(v)$) کمتر از مقدار فعلی تابع برازش ($F(x)$) باشد، در ستون i-ام مقدار هر عنصر برابر با یکی بیشتر از بیشترین مقدار در سطر خودش می‌شود. همچنین بعد از این بروزرسانی‌های جدول حافظه، کنترل می‌شود که مقدار قطر اصلی این جدول برابر با $-\infty$ باشد زیرا هیچ عاملی نسبت به خودش موقعیتش را بروز نمی‌کند.

در ادامه برای شرح بیشتر، یک مثال برای جدول حافظه با چهار عامل جستجو (جمعیت برابر با چهار) در شکل (3) آورده شده است. در ابتدا مقدار تمام عناصر جدول حافظه صفر است (شکل (3)-الف). در این مرحله $i = 1$ است. بنابراین مقادیر اندیس j یکی از مقدارهای {2,3,4} است.

A random set of rabbits denoted as X_i is generated (population generation).
The X_i are evaluated
Fitness (Fit), and X_{best} are the optimal solution achieve thus far.
while the stop condition isn't satisfied do
    for each X_i in population do
        The factor A is calculated using Eq. (13).
        If $A > 1$ then
            A rabbit randomly is chosen from population.
            The R is calculated using Eq. (2).
            Detour foraging is performed using Eq. (1).
            The fitness (Fit_i) are calculated
            The position of the current X_i is updated using Eq. (12).
        else
            d burrows are generated using Eq. (7).
            One of burrows randomly is picked as hiding
            Random hiding is performed using Eq. (11).
            The fitness (Fit_i) are calculated
            The position of the X_i is updated using Eq. (12).
        end if
    The current optimal solution found thus far (X_{best}) is updated
    end for
end while
Return X_{best}

شکل (2): شبه کد الگوریتم ARO

	x_1	x_2	x_3	x_4
$F(v)$	8.1			

	x_1	x_2	x_3	x_4
$F(x)$	3.5	6.1	6	2

MT	x_j			
x_i	$-\infty$	0	0	0
	0	$-\infty$	0	0
	0	0	$-\infty$	0
	0	0	0	$-\infty$

(الف)

	x_1	x_2	x_3	x_4
$F(v)$	8.1	4.3		

	x_1	x_2	x_3	x_4
$F(x)$	3.5	6.1	6	2

MT	x_j			
x_i	$-\infty$	1	1	0
	0	$-\infty$	0	0
	0	0	$-\infty$	0
	0	0	0	$-\infty$

(ب)

	x_1	x_2	x_3	x_4
$F(v)$	8.1	4.3	8.9	

	x_1	x_2	x_3	x_4
$F(x)$	3.5	4.3	6	2

MT	x_j			
x_i	$-\infty$	2	1	0
	1	$-\infty$	1	0
	0	1	$-\infty$	0
	0	1	0	$-\infty$

(ج)

	x_1	x_2	x_3	x_4
$F(v)$	8.1	4.3	8.9	1.5

	x_1	x_2	x_3	x_4
$F(x)$	3.5	4.3	6	2

MT	xj			
x_i	$-\infty$	2	1	0
	0	$-\infty$	1	0
	1	0	$-\infty$	1
	0	1	0	$-\infty$

(د)

شکل (3): مثال از نحوه عملکرد جدول حافظه (MT: Memory Table)

با توجه به اینکه تمام مقادیر در ردیف $i = 1$ برابر صفر است. بنابراین اندیس j برابر با کمترین مقدار تابع برازش $(F(x))$ از بین آنها خواهد بود که در این مثال $j = 4$ می‌شود. سپس، اولین عامل جستجو $(i = 1)$ موقعیت خود را با توجه به عنصر چهارم $(j = 4)$ در رابطه (1) بروز می‌کند. اما چون مقدار جدید تابع برازش بیشتر از مقدار قبلی است در ستون اول جدول حافظه همه مقادیر به یکی بیشتر از، بیشترین مقدار در سطر مربوطه افزایش پیدا نمی‌کند. در شکل (3)-ب، $i = 2$ است. بنابراین در سطر دوم جدول حافظه اندیس ستون بیشترین مقدار را پیدا می‌کنیم. با توجه به این‌که چند مورد بیشترین مقدار وجود دارد. $j = 4$ را از میان آنها انتخاب می‌کنیم چون کمترین مقدار تابع برازش را دارد. بنابراین موقعیت عامل جستجوی دوم با توجه به عامل جستجوی چهارم در رابطه (1) بروز می‌شود. سپس مقدار عنصر سطر i-ام و ستون j-ام در جدول حافظه برابر صفر می‌شود.

بقیه موارد در سطر مربوطه یک واحد بیشتر می‌شوند و همچنین در ستون دوم نیز هر عنصر به یکی بیشتر از بیشترین مقدار در سطر خودش افزایش می‌یابد زیرا مقدار برازش جدید محاسبه شده کمتر از مقدار قبلی است. که در نتیجه آن (شکل (3)-ج) حاصل می‌شود. جدول حافظه در مورد بعدی (تبدیل از ج به د) نیز مانند این دو حالت بروز می‌شود. شبه کد روش پیشنهادی در شکل (4) آمده است.

در ادامه استفاده از روش پیشنهادی برای حل دستگاه معادلات غیرخطی شرح داده شده است. دستگاه معادلات غیرخطی زیر را در نظر بگیرید:

$$\begin{cases} f_1(x_1, x_2, x_3, \ldots, x_n) = 0 \\ f_2(x_1, x_2, x_3, \ldots, x_n) = 0 \\ \vdots \\ f_m(x_1, x_2, x_3, \ldots, x_n) = 0 \end{cases} \quad (14)$$

که در آن n تعداد متغیرها و m تعداد معادلات دستگاه معادلات غیرخطی است. برای حل دستگاه معادلات غیرخطی، این دستگاه را به صورت مساله بهینه‌سازی بدون قیود، به صورت یکی روابط (15) یا (16) بازنویسی می‌کنیم.

A random set of rabbits denoted as X_i is generated (population generation).
Create a 2-dimensional array for memory table (MT) with size (population size, population size)
The X_i are Evaluated
Fitness (Fit), and X_{best} are the optimal solution achieve thus far.
while the stop condition isn't satisfied do
    Setting the diameter of the MT array to negative infinity
    for each X_i in population do
        the factor A is calculated using Eq. (13).
        If $A > 1$ then
            Finding j index using memory table and fitness value
            The R is calculated using Eq. (2).
            Detour foraging is performed using Eq. (1).
            The fitness (Fit_i) are calculated
            The position of the current X_i is updated using Eq. (12).
            The current optimal solution found thus far (X_{best}) is updated
            If Fit of current optimal solution < Fit of X_{best}
                Increase by one in ith row of MT
                Set zero to selected agent in MT
                Increase by one corresponding column of the selected agent in the MT
            else
                Increase by one in ith row of MT
                Set zero to selected agent in MT
        else
            d burrows are generated using Eq. (7).
            One of burrows randomly is picked as hiding
            Random hiding is performed using Eq. (11).
            The fitness (Fit_i) is calculated
            The position of the X_i is updated using Eq. (12).
            The current optimal solution found thus far (X_{best}) is updated
        end if
    end for
end while
Return X_{best}

شکل (4): شبه کد الگوریتم IARO

که در آن R^n یک فضای جستجو برای جواب‌ها است. هر الگوریتم فراابتکاری نیاز به یک تابع برازش برای ارزیابی جواب‌های پیدا کرده خود در هر تکرار دارد. با تبدیل دستگاه معادلات غیرخطی به یک مساله مینیم‌سازی می‌توان پلی بین الگوریتم‌های فراابتکاری و حل دستگاه معادلات غیرخطی ایجاد کرد. برای حل دستگاه معادلات (14) آن را به شکل (15) یا (16) نوشته و از آن به عنوان توابع برازش برای روش پیشنهادی

$$\begin{cases} \text{Find:} & X = (x_1, x_2, \dots, x_n)^T, X \in R^n; \\ \text{Min:} & F(X) = \sum_{i=1}^{n} f_i^2 \end{cases} \quad (15)$$

یا

$$\begin{cases} \text{Find:} & X = (x_1, x_2, \dots, x_n)^T, X \in R^n; \\ \text{Min:} & F(X) = \sqrt{\sum_{i=1}^{n} f_i^2} \end{cases} \quad (16)$$

(الگوریتم IARO) استفاده می‌شود.

در آغاز این روش، موقعیت عامل‌های جستجو به صورت تصادفی مقداردهی می‌شوند. پس از آن، مقدار تابع برازش جمعیت اولیه با استفاده از توابع (15) یا (16) محاسبه می‌شود. سپس روند الگوریتم IARO شروع می‌شود. در خلال اجرای IARO اگر نیاز به محاسبه مقدار تابع برازش برای آن تکرار باشد، با توابع (16) یا (17) دوباره محاسبه می‌شود. در هر تکرار شرط پایان که حداکثر تعداد تکرار است کنترل می‌شود و اگر به حداکثر تعداد تکرار رسیده باشد، الگوریتم پایان می‌یابد و موقعیت بهترین عامل جستجو در تکرار را برمی‌گرداند که برابر جواب دستگاه معادلات غیرخطی خواهد بود.

4. آزمایش‌ها و نتایج

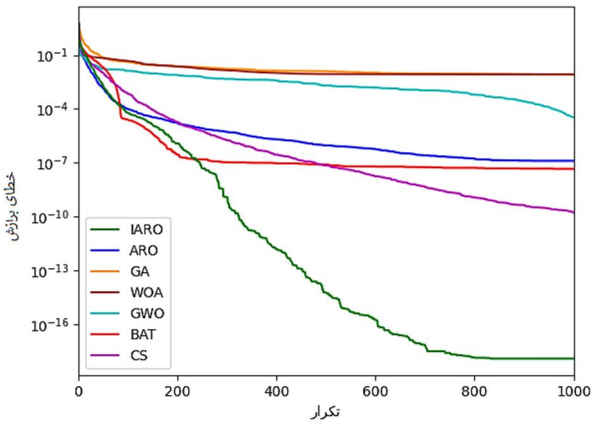
در این بخش، برای بررسی عملکرد روش پیشنهادی، 10 دستگاه معادلات با استفاده از این روش حل شده است. این دستگاه‌های معادلات غیرخطی در دنیای واقعی در زمینه‌های مهندسی شیمی مورد بررسی قرار گرفته است [34]، [35]. برای بررسی دقیق عملکرد الگوریتم پیشنهادی، 20 اجرای متوالی با روش پیشنهادی انجام گرفته و نتایج میانگین آن با جواب واقعی یا جواب‌های تقریبی به دست آمده در منابع قبلی مورد مقایسه قرار گرفته است. همچنین الگوریتم بهبود یافته (IARO) با الگوریتم‌های ARO، GA ، WOA [36]، GWO، BAT و CS [37]، مورد مقایسه قرار گرفته است. الگوریتم‌های ذکر شده از روش‌های پایه‌ای یا جدید بهینه‌سازی هستند.

برای پیاده‌سازی الگوریتم فراابتکاری از زبان برنامه‌نویسی پایتون استفاده شده است. همچنین پردازنده دستگاه مورد اجرا CPU Core i3 2.44 GHz با رم 4GB می‌باشد. منبع کدهای این مقاله در پیوند[1] پژوهش به صورت عمومی قابل دسترسی است. تعداد جمعیت الگوریتم فراابتکاری در همه موارد برابر 100 و تعداد تکرار الگوریتم‌ها[2] برابر 1000 در نظر گرفته شده است.

مورد 1: این مورد اغلب دستگاه معادلات غیرخطی براون [38] نامیده می‌شود و به شکل زیر است:

$$\begin{cases} 2x_1 + x_2 + x_3 + x_4 + x_5 = 6 \\ x_1 + 2x_2 + x_3 + x_4 + x_5 = 6 \\ x_1 + x_2 + 2x_3 + x_4 + x_5 = 6 \\ x_1 + x_2 + x_3 + 2x_4 + x_5 = 6 \\ x_1 x_2 x_3 x_4 x_5 = 1 \end{cases} \quad (17)$$

که در آن $-2 \le x_i \le 2, i = 1, \dots, 5$ است. همان‌طور که در جدول (1) مشاهده می‌شود، روش پیشنهادی توانسته مقدار تابع برازش را تا $1.2E\text{-}18$ کاهش دهد که بهترین عملکرد بین الگوریتم‌های مقایسه شده است. ES در ستون آخر نشان‌دهنده جواب دقیق دستگاه معادلات است. همچنین در شکل (5) مقدار تابع برازش در تکرارهای مختلف نمایش می‌دهد. در این شکل عملکرد خوب IARO مشاهده می‌شود.



شکل (5): نمودار مقدار خطای برازش دستگاه معادلات شماره یک در تکرارهای یک تا هزار برای الگوریتم‌های مورد بررسی

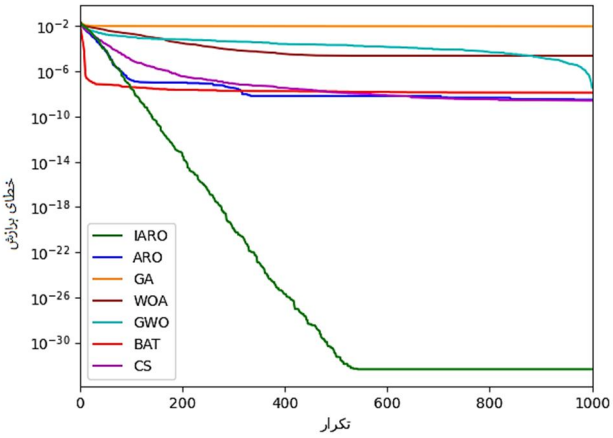
مورد 2: دستگاه معادلات زیر را با چهار مجهول تعریف می‌کنیم.

$$x_i - \cos\left(2x_i - \sum_{j=1}^{4} x_j\right) = 0 \quad 1 \le i \le 4 \quad (18)$$

که در آن $-100 \le x_i \le 100, i = 1, \dots, 4$ است. همان‌طور که در جدول (2) مشاهده می‌شود، روش پیشنهادی توانسته مقدار خطای برازش را تا $4.4E\text{-}33$ کاهش دهد که بهترین عملکرد بین الگوریتم‌های مقایسه شده است. پژوهش صورت گرفته در مرجع [39] روشی به نام «نیوتون وزندار کارا[3]» (EWN) را برای حل

[1] https://github.com/alihamdipour/IARO-NES
[2] Maximum Iterations
[3] Efficient weighted-Newton

دستگاه معادلات معرفی کرده است. با توجه به اینکه دستگاه معادلات غیرخطی بررسی شده جواب دقیق ندارد، جواب‌های تقریبی دستگاه معادلات با روش EWN نیز در جدول (2) آورده شده است. همچنین در شکل (6) عملکرد خوب IARO در مقایسه با دیگر الگوریتم‌های بررسی شده قابل مشاهده است.



شکل (6): نمودار مقدار خطای برازش دستگاه معادلات شماره دو در تکرارهای یک تا هزار برای الگوریتم‌های مورد بررسی

مورد 3: دستگاه معادلات زیر را با چهار مجهول تعریف می‌کنیم.

$$\begin{cases} x_1^2 + x_3^2 = 1 \\ x_2^2 + x_4^2 = 1 \\ x_5 x_3^3 + x_6 x_4^3 = 0 \\ x_5 x_1^3 + x_6 x_2^3 = 0 \\ x_5 x_1 x_3^2 + x_6 x_2 x_4^2 = 0 \\ x_5 x_3 x_1^2 + x_6 x_4 x_4^2 = 0 \end{cases} \quad (19)$$

که در آن $-10 \le x_i \le 10, i = 1, \dots, 6$ است. همان‌طور که در جدول (3) مشاهده می‌شود، روش پیشنهادی توانسته مقدار خطای برازش را تا $3.7E\text{-}14$ کاهش دهد که بهترین عملکرد بین الگوریتم‌های مقایسه شده است. همچنین در شکل (7) مقدار خطای برازش در تکرارهای مختلف نمایش می‌دهد. در این شکل عملکرد خوب IARO مشاهده می‌شود.

مورد 4: دستگاه معادلات زیر را با ده مجهول تعریف می‌کنیم.

$$\begin{cases} x_1 - 0.25428722 - 0.18324757 x_4 x_3 x_9 = 0 \\ x_2 - 0.37842197 - 0.16275449 x_1 x_{10} x_6 = 0 \\ x_3 - 0.27162577 - 0.16955071 x_1 x_2 x_{10} = 0 \\ x_4 - 0.19807914 - 0.15585316 x_7 x_1 x_6 = 0 \\ x_5 - 0.44166728 - 0.19950920 x_7 x_6 x_3 = 0 \\ x_6 - 0.14654113 - 0.18922793 x_8 x_5 x_{10} = 0 \\ x_7 - 0.42937161 - 0.21180486 x_2 x_5 x_8 = 0 \\ x_8 - 0.07056438 - 0.17081208 x_1 x_7 x_6 = 0 \\ x_9 - 0.34504906 - 0.19612740 x_{10} x_6 x_8 = 0 \\ x_{10} - 0.42651102 - 0.21466544 x_4 x_8 x_1 = 0 \end{cases} \quad (20)$$

جدول (1): نتایج به دست آمده از حل دستگاه معادلات شماره یک (VAR: VARIABLE, ES:EXACT SOLUTION)

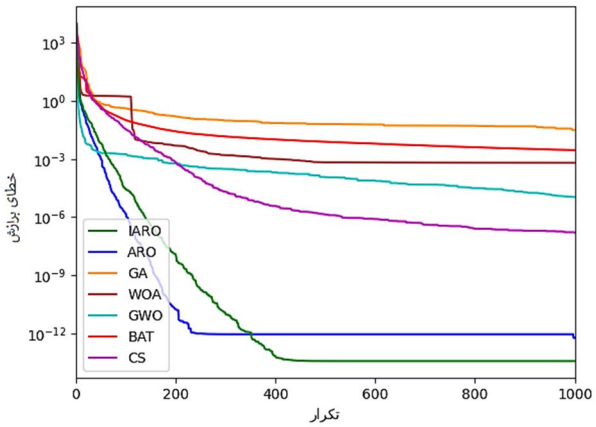
Var	IARO	ARO	GA	WOA	GWO	BAT	CS	ES
x_1	1	0.99995842	0.99499711	1.0787193	0.99945614	1.00021335	0.99499711	1
x_2	1	0.9999571	0.94757832	0.9907849	0.99927208	1.00011296	0.94757832	1
x_3	1	0.9999445	0.93808635	1.03625201	0.9998472	1.00009496	0.93808635	1
x_4	1	0.99994348	0.78982767	1.02597104	0.99918784	1.0002526	0.78982767	1
x_5	1	1.00023525	0.96599193	0.84346796	1.00254621	0.99917949	0.96599193	1
f	1.2770633E-18	1.2884102E-07	7.9754840E-03	8.4313287E-03	3.4365138E-05	4.6162474E-08	1.7097578E-10	0

جدول (2): نتایج به دست آمده از حل دستگاه معادلات شماره دو

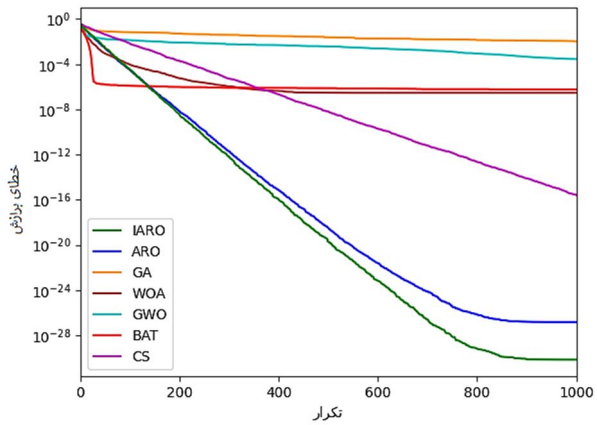
Var	IARO	ARO	GA	WOA	GWO	BAT	CS	EWN
x_1	0.51493326	0.51493352	0.58480692	0.51242665	0.5147858	0.51486608	0.39216094	0.51493326
x_2	0.51493326	0.51493352	0.41199934	0.51304262	0.51503343	0.51505878	0.48620573	0.51493326
x_3	0.51493326	0.51493352	0.47076047	0.52146724	0.51485113	0.51488535	0.65701088	0.51493326
x_4	0.51493326	0.51493352	0.43630517	0.51275569	0.51500014	0.5149187	0.47772335	0.51493326
f	4.4681575E-33	3.1301764E-09	9.8500520E-03	2.4871264E-05	3.7962488E-08	1.3453352E-08	2.6160838E-09	-

جدول (3): نتایج به دست آمده از حل دستگاه معادلات شماره سه

Var	IARO	ARO	GA	WOA	GWO	BAT	CS
x_1	0.86592896	8.87390218E-01	0.88297998	-9.61127510E-01	0.14778358	-0.95318194	0.8163401
x_2	0.9904602	-4.46170492E-01	0.2535923	-8.46879815E-02	0.14781548	-0.53215372	0.0167982
x_3	0.50007801	4.61019089E-01	0.62833815	-2.76125807E-01	-0.9889309	-0.92684248	0.51176833
x_4	0.13323532	8.94947983E-01	0.91608268	-9.96444958E-01	-0.98905092	-1.75210087	0.8143668
x_5	-0.0611113	4.31877427E-42	0.30043738	-0.30127510E-01	-2.57657861	9.26710673	0.31846923
x_6	0.05581324	2.20118609E-41	0.0625095	-8.9688815E-02	2.57574712	1.72921022	0.03644078
f	3.78919573E-14	5.88633622E-13	3.16679253E-02	6.39473536E-04	1.09311551E-05	2.88106683E-03	0.002881066



شکل (7): نمودار مقدار خطای برازش دستگاه معادلات شماره سه در تکرارهای یک تا هزار برای الگوریتم‌های مورد بررسی



شکل (8): نمودار مقدار خطای برازش دستگاه معادلات شماره چهار در تکرارهای یک تا هزار برای الگوریتم‌های مورد بررسی

در معادلات (20)، $-2 \le x_i \le 2, i = 1, \dots, 10$ است. همان‌طور که در جدول (4) مشاهده می‌شود، روش پیشنهادی توانسته مقدار خطای برازش را تا $7.1E\text{-}31$ کاهش دهد که بهترین عملکرد بین الگوریتم‌های مقایسه شده است. همچنین در شکل (8) مقدار خطای برازش در تکرارهای مختلف را نمایش می‌دهد. در این شکل عملکرد خوب IARO مشاهده می‌شود. در این نمودارها محور عمودی نشان‌دهنده مقدار خطای برازش میانگین (میانگین 20 بار اجرا) است. بنابراین نمودار پایین‌تر نشان‌دهنده جواب بهتر است. اگر الگوریتم بتواند جواب دقیق را پیدا کند مقدار خطای برازش برابر صفر خواهد شد.

مورد 5: دستگاه معادلات زیر را با هشت مجهول تعریف می‌کنیم.

$$\begin{cases} 4.731 \times 10^{-3} x_1 x_3 - 0.3578 x_2 x_3 - 0.1238 x_1 + x_7 - 1.637 \times 10^{-3} x_2 - 0.9338 x_4 = 0.3571 \\ 0.2238 x_1 x_3 + 0.7623 x 2 x_3 + 0.2638 x_1 - x_7 - 0.07745 x_2 - 0.6734 x_4 = 0.6022 \\ x_6 x_8 + 0.3578 x_1 + 4.731 \times 10^{-3} x_2 = 0 \\ -0.7623 x_1 + 0.2238 x_2 = -0.3461 \\ x_1^2 + x_2^2 = 1 \\ x_3^2 + x_4^2 = 1 \\ x_5^2 + x_6^2 = 1 \\ x_7^2 + x_8^2 = 1 \end{cases} \quad (21)$$

که در آن $-1 \le x_i \le 1, i = 1, \dots, 8$ است. همان‌طور که در جدول (5) مشاهده می‌شود، روش پیشنهادی توانسته مقدار خطای برازش را تا $7.8E\text{-}25$ کاهش دهد که بهترین عملکرد بین الگوریتم‌های مقایسه شده است.

جدول (4): نتایج به دست آمده از حل دستگاه معادلات شماره چهار

Var	IARO	ARO	GA	WOA	GWO	BAT	CS	FASA
x_1	0.25783339	0.25783339	0.0125144	0.25797826	0.2579793	0.25800869	0.0125144	0.2578334
x_2	0.38109715	0.38109715	0.16286594	0.38117908	0.38134709	0.38145555	0.16286594	0.3810972
x_3	0.27874502	0.27874502	0.69335026	0.27884696	0.27892543	0.27839438	0.69335026	0.2787450
x_4	0.20066896	0.20066896	0.23189673	0.20080711	0.20079768	0.20078828	0.23189673	0.2006690
x_5	0.44525142	0.44525142	0.62170693	0.44512692	0.44532774	0.44482351	0.62170693	0.4452514
x_6	0.14918392	0.14918392	0.31181227	0.14920295	0.14923197	0.14930236	0.31181227	0.1491839
x_7	0.4320097	0.4320097	0.54014274	0.43212387	0.43220142	0.43177881	0.54014274	0.4320097
x_8	0.07340278	0.07340278	0.15748612	0.0736315	0.07351296	0.07350713	0.15748612	0.0734028
x_9	0.34596683	0.34596683	0.30485177	0.34574249	0.34610746	0.34602261	0.30485177	0.3459668
x_{10}	0.42732628	0.42732628	0.44583852	0.42722305	0.42753504	0.42740308	0.44583852	0.4273263
f	7.1796980E-31	1.5340244E-27	1.0763711E-02	2.9629424E-07	2.9073140E-04	5.9359123E-07	2.6712270E-16	-

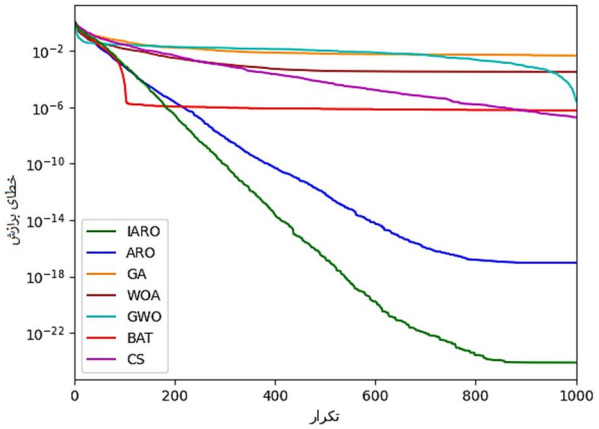
جدول (5): نتایج به دست آمده از حل دستگاه معادلات شماره پنج

Var	IARO	ARO	GA	WOA	GWO	BAT	CS	FASA
x_1	0.67155426	0.67155426	0.71586707	0.65995277	0.67085728	0.67176862	0.71586707	0.2578334
x_2	0.74095538	0.74095538	0.65735549	0.75471194	0.74194709	0.74084692	0.65735549	0.3810972
x_3	0.95189275	0.95189275	0.91386614	0.95803792	0.95205429	0.95218588	0.91386614	0.278745
x_4	-0.3064313	-0.3064313	0.21095073	-0.2973689	-0.3069533	-0.3063365	0.21095073	0.200669
x_5	0.9638107	0.96381077	0.40922854	0.96114208	0.96460567	0.96358805	0.40922854	0.4452514
x_6	-0.2665873	-0.2665873	0.14006579	-0.2714958	-0.2645041	0.26687355	0.14006579	0.1491839
x_7	0.40464139	0.40464139	0.59394807	0.41662879	0.40463018	0.40467647	0.59394807	0.4320097
x_8	0.91447545	0.91447545	0.47515345	0.90841482	0.91448521	-0.9145035	0.47515345	0.0734028
f	7.8914907E-25	9.2965905E-18	4.6537353E-03	3.2649249E-04	2.8032127E-06	6.0005824E-07	1.9333144E-07	-

پژوهش صورت گرفته در مرجع [40]، روشی به نام «الگوریتم تبلیقی تبرید شبیه‌سازی‌شده فـازی[4]» (FASA) را بـرای حـل دستگاه معادلات معرفی کرده است. با توجه به اینکـه دسـتگاه معادلات غیرخطی بررسی شده جواب دقیق ندارد، جواب‌های تقریبی دستگاه معادلات با روش FASA نیز در جدول (5) آورده شده است. با توجه به اینکه دستگاه معادلات ممکن است بیش از یک جواب داشته باشد بنابراین فقـط یـک جـواب دسـتگاه معادلات در جدول (5) آورده شده است ولی برای مقایسه عملکرد از مقدار خطای برازش استفاده شده که میانگین مقدار خطـای برازش 20 اجرا گرفته شده است. دلیل اختلاف جواب‌هـا در این جدول داشتن جواب‌های متعدد این دستگاه معادلات است. بنابراین با توجه به تعدد جواب‌های دستگاه معادلات، میـانگین جواب‌ها گرفته نشده است و فقط میانگین خطای برازش گرفته شده است. این نکته باعث می‌شود که در برخی دستگاه‌های مورد بررسی مانند مورد اول، با اینکه جواب دقیـق بـرای الگـوریتم پیشنهادی در جدول برای آن مورد آورده شده است ولی خطای برازش صفر نیست که نشان‌دهنده این است که در برخی موارد

[4] Fuzzy adaptive simulated annealing

از 20 اجرای گرفته شده الگوریتم پیشنهادی جواب تقریبی به دست آورده است. بنابراین مبنای مقایسه میانگین خطای برازش مورد بررسی قرار گرفته است. همچنین در شکل (9) مقدار خطای برازش در تکرارهای مختلف نمایش می‌دهد که عملکرد خوب IARO مشاهده می‌شود.



شکل (9): نمودار مقدار خطای برازش دستگاه معادلات شماره پنج در تکرارهای یک تا هزار برای الگوریتم‌های مورد بررسی

مورد 6: دستگاه معادلات زیر را با سه مجهول تعریف می‌کنیم.

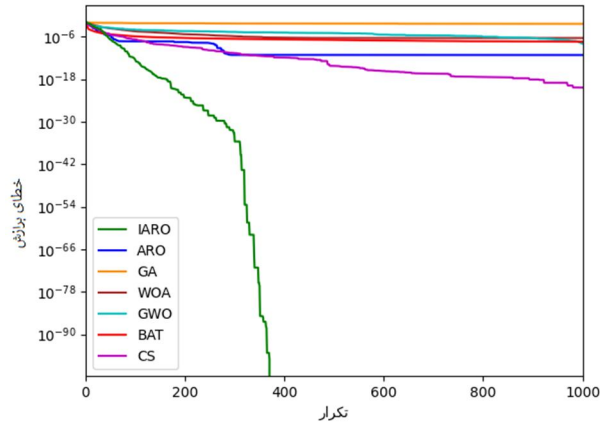
$$\begin{cases} exp(x_1^2) - 8x_1\, sin(x_2) = 0 \\ x_1 + x_2 = 1 \\ (x_3 - 1)^3 = 0 \end{cases} \quad (22)$$

که در آن $-10 \le x_i \le 10, i = 1, \dots, 3$ است. همان‌طور که در جدول (6) مشاهده می‌شود، روش پیشنهادی توانسته مقدار خطای برازش را تا صفر کاهش دهد که بهترین عملکرد بین الگوریتم‌های مقایسه شده است. همچنین در شکل (10)، مقدار خطای برازش در تکرارهای مختلف نمایش می‌دهد. در این شکل عملکرد خوب IARO مشاهده می‌شود.

مورد 7: دستگاه معادلات زیر را با دو مجهول تعریف می‌کنیم.

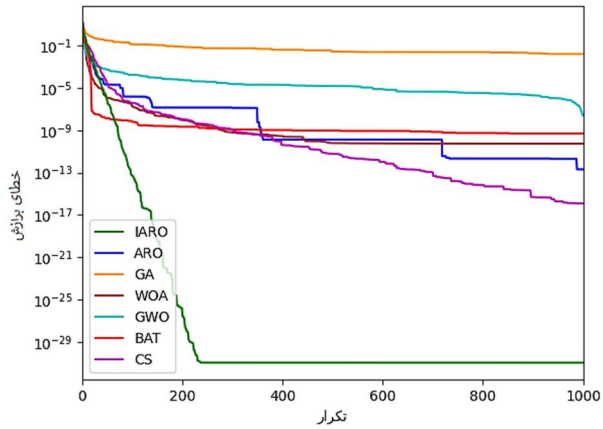
$$\begin{cases} x_1^3 - 3x_1x_2^2 = 1 \\ 3x_2x_1^2 - x_2^3 = -1 \end{cases} \quad (23)$$

که در آن $-10 \le x_i \le 10, i = 1,2$ است. همان‌طور که در جدول (7) مشاهده می‌شود، روش پیشنهادی توانسته مقدار خطای برازش را تا $1.08E\text{-}31$ کاهش دهد که بهترین عملکرد بین الگوریتم‌های مقایسه شده است.



شکل (10): نمودار مقدار خطای برازش دستگاه معادلات شماره شش در تکرارهای یک تا هزار برای الگوریتم‌های مورد بررسی

پژوهش صورت گرفته در مرجع [41]، روشی به نام «الگوریتم ترکیبی وال و جمعیت گل[5]» (HWF) را برای حل دستگاه معادلات معرفی کرده است. با توجه به اینکه دستگاه معادلات غیرخطی بررسی شده جواب دقیق ندارد، جواب‌های تقریبی دستگاه معادلات با روش HWF نیز در جدول (7) آورده شده است. همچنین در شکل (11) مقدار خطای برازش در تکرارهای مختلف نمایش می‌دهد، این شکل عملکرد خوب IARO مشاهده می‌شود. همان‌طوری که مشاهده می‌شود، روش پیشنهادی در تکرارهای کمتر توانسته جواب دقیق‌تری را به دست آورد.



شکل (11): نمودار مقدار خطای برازش دستگاه معادلات شماره هفت در تکرارهای یک تا هزار برای الگوریتم‌های مورد بررسی

[5] Hybridizing whale optimization algorithm and flower pollination Algorithm

جدول (6): نتایج به دست آمده از حل دستگاه معادلات شماره شش

Var	IARO	ARO	GA	WOA	GWO	BAT	CS	FASA
x_1	0.17559892	0.17559548	0.17559548	0.17569517	0.1755909	0.17560379	0.17559548	0.175599
x_2	0.82440108	0.82441236	0.81539023	0.82402421	0.82439636	0.82439268	0.81539023	0.824401
x_3	1	1.00000971	0.8265568	0.90703918	0.9406832	0.94160713	0.8265568	1
f	0	6.46476289E-12	0.00403971	4.08040104E-7	1.05177479E-8	3.67639377E-8	4.32899427E-21	-

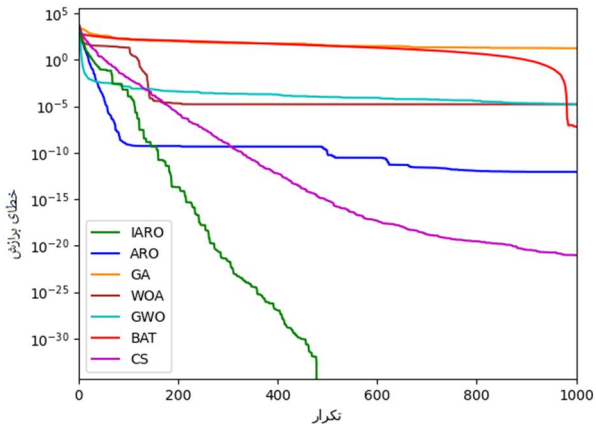
جدول (7): نتایج به دست آمده از حل دستگاه معادلات شماره هفت

Var	IARO	ARO	GA	WOA	GWO	BAT	CS	HWF
x_1	-0.2905145	-0.2905145	-1.5157396	-0.2905142	-0.2905053	-0.2905185	-0.3986028	-0.2905145
x_2	1.08421508	1.08421508	0.90445397	1.08421501	1.08418532	1.08421564	0.60390547	1.08421508
f	1.0846837E-31	2.0126928E-13	1.6224529E-02	5.3373718E-11	2.5162371E-08	4.8935839E-10	1.2533792E-16	-

مورد 8: دستگاه معادلات زیر را با سه مجهول تعریف می‌کنیم.

$$\begin{cases} 3x_1 - cos(x_2x_3) = 0.5 \\ x_1^2 - 625x_2^2 = 0.25 \\ exp(-x_1x_2) + 20x_3 + \dfrac{10\pi - 3}{3} = 0 \end{cases} \quad (24)$$

که در آن $-10 \le x_i \le 10, i = 1,\dots,3$ است. همان‌طور که در جدول (8) مشاهده می‌شود، روش پیشنهادی توانسته مقدار خطای برازش را تا صفر کاهش دهد که بهترین عملکرد بین الگوریتم‌های مقایسه شده است. همچنین شکل (12) مقدار خطای برازش در تکرارهای مختلف نمایش می‌دهد. در این شکل عملکرد خوب IARO مشاهده می‌شود.

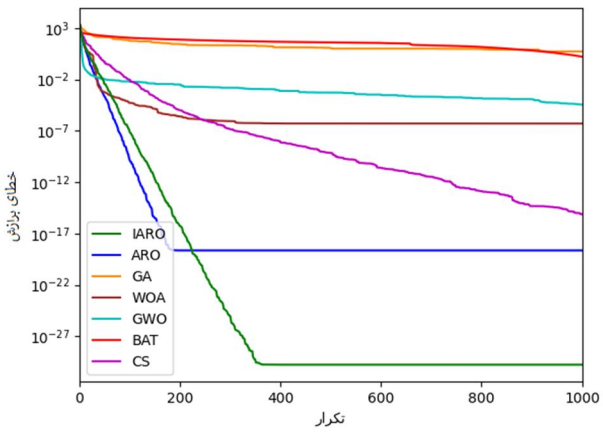


شکل (12): نمودار مقدار خطای برازش دستگاه معادلات شماره هشت در تکرارهای یک تا هزار برای الگوریتم‌های مورد بررسی

مورد 9: دستگاه معادلات زیر را با سه مجهول تعریف می‌کنیم.

$$\begin{cases} 3x_1 - cos(x_2x_3) = 0.5 \\ x_1^2 - 81(x_2 + 0.1)^2 + sin(x_3) + 1.06 = 0 \\ exp(-x_1x_2) + 20x_3 + \dfrac{10\pi - 3}{3} = 0 \end{cases} \quad (25)$$

که در آن $-10 \le x_i \le 10, i = 1,\dots,3$ است. همان‌طور که در جدول (9) مشاهده می‌شود، روش پیشنهادی توانسته مقدار خطای برازش را تا $1.7E\text{-}30$ کاهش دهد که بهترین عملکرد بین الگوریتم‌های مقایسه شده است. همچنین در شکل (13) مقدار خطای برازش در تکرارهای مختلف نمایش می‌دهد. در این شکل عملکرد خوب IARO مشاهده می‌شود.



شکل (13): نمودار مقدار خطای برازش دستگاه معادلات شماره نه در تکرارهای یک تا هزار برای الگوریتم‌های مورد بررسی

جدول (8): نتایج به دست آمده از حل دستگاه معادلات شماره هشت

Var	IARO	ARO	GA	WOA	GWO	BAT	CS	FASA
x_1	5.000000E-01	5.000000E-01	9.85042268	0.50135204	0.50035267	5.0001564E-01	9.85042268	0.5
x_2	-4.76460E-11	5.5022725E-08	-0.2530489	-0.0046484	0.00226559	-1.269818E-04	-0.2530489	8.9236228E-11
x_3	-5.23599E-01	-5.235988E-01	-3.0733969	-0.5237138	-0.5235437	-5.236009E-01	-3.0733969	−5.235988E-01
f	0	8.8425891E-13	1.7790688E+01	1.6440982E-05	1.5868346E-05	6.5892494E-08	9.3637419E-22	-

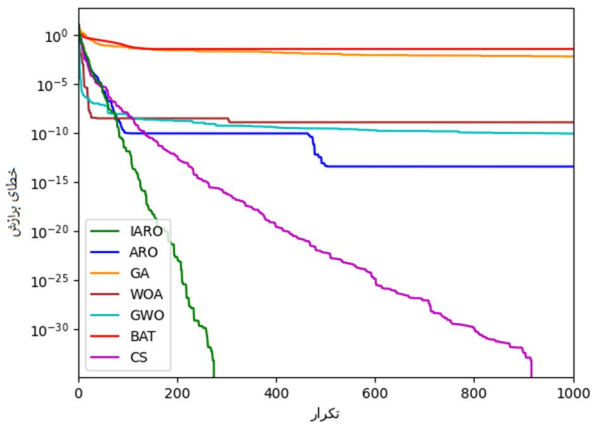
جدول (9): نتایج به دست آمده از حل دستگاه معادلات شماره نه

Var	IARO	ARO	GA	WOA	GWO	BAT	CS	ES
x_1	0.500000000	0.50000000	3.73369126	0.49816585	0.50041828	0.49990648	3.73369126	0.5
x_2	-2.9649677E-18	-4.8461989E-19	-0.5518468	-0.19962032	2.4208896E-04	2.1153626E-05	-0.5518468	0
x_3	-0.52359877	-0.52359877	0.7353487	-0.52882293	-0.5235929	-0.52359957	0.7353487	0.5
f	1.7410407E-30	2.3115346E-19	5.37458130	5.1422368E-07	3.6062630E-05	1.66056884	7.8708676E-16	0

مورد 10: دستگاه معادلات زیر را با دو مجهول تعریف می‌کنیم.

$$\begin{cases} exp(x_1) + x_1 x_2 = 1 \\ sin(x_1 x_2) + x_1 + x_2 = 1 \end{cases} \quad (26)$$

که در این دستگاه معادلات $-10 \le x_i \le 10, i = 1,2$ است. همان‌طور که در جدول (10) مشاهده می‌شود، روش پیشنهادی توانسته مقدار خطای برازش را تا صفر کاهش دهد که بهترین عملکرد بین الگوریتم‌های مقایسه شده است. همچنین شکل (14) مقدار خطای برازش در تکرارهای مختلف را نمایش می‌دهد. در این شکل عملکرد خوب IARO مشاهده می‌شود.



شکل (14): نمودار مقدار خطای برازش دستگاه معادلات شماره ده در تکرارهای یک تا هزار برای الگوریتم‌های مورد بررسی

مورد 11: دستگاه معادلات زیر را با پنج مجهول تعریف می‌کنیم.

$$\begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 = 0 \\ x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_1x_5 = 0 \\ x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_5 + \\ \qquad x_1x_2x_5 + x_1x_4x_5 = 0 \\ x_1x_2x_3x_4 + x_2x_3x_4x_5 + x_1x_3x_4x_5 + \\ \qquad x_1x_2x_4x_5 + x_1x_2x_3x_5 = 0 \\ x_1x_2x_3x_4x_5 = 1 \end{cases} \quad (27)$$

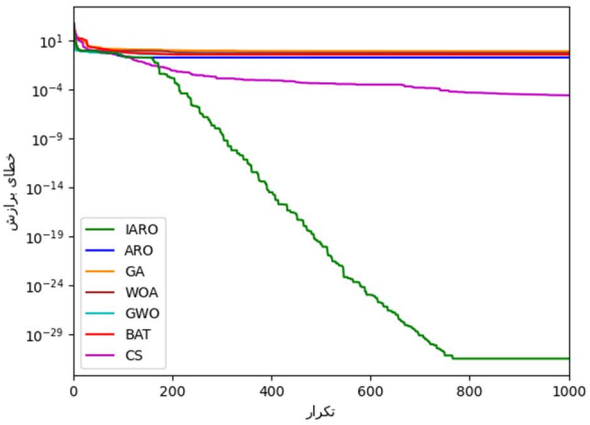
که در آن $-10 \le x_i \le 10, i = 1, \dots, 5$ است. همان‌طور که در جدول (11) مشاهده می‌شود، روش پیشنهادی توانسته مقدار خطای برازش را تا $3.14E\text{-}3$ کاهش دهد که بهترین عملکرد بین الگوریتم‌های مقایسه شده است. همچنین در شکل (15) مقدار خطای برازش در تکرارهای مختلف نمایش می‌دهد. در این شکل عملکرد خوب IARO مشاهده می‌شود.

یکی از شاخص‌های مهم برای ارزیابی عملکرد الگوریتم‌های متفاوت برای یک مساله خاص، زمان اجرای آن الگوریتم‌ها برای حل مساله مربوطه است. به همین منظور، زمان اجرای هر یک از الگوریتم‌ها برای هر کدام از دستگاه معادلات در جدول (12) امده است. برای رتبه‌بندی کلی الگوریتم‌ها، مجموع زمان اجرا برای همه 11 مورد در جدول (12) محاسبه شده است. بهبود مورد نظر به صورت کلی باعث افزایش جزئی در زمان اجرا برای الگوریتم پیشنهادی شده است. ولی با توجه به کارایی بالایی

روش پیشنهادی می‌توان این اختلاف جزئی را نادیده گرفت. به صورت کلی زمان اجرا برای همه موراد مقایسه شده نزدیک به هم است که ناشی از یکسان بودن پیچیدگی زمانی هر هفت الگوریتم مورد مقایسه است. پیچیدگی زمانی برای همه هفت الگوریتم برابر است با $O(N \times I \times E)$ است که مولفه‌های آن به صورت زیر است.

- N: تعداد جمعیت الگوریتم فراابتکاری
- I: تعداد دورهای الگوریتم فراابتکاری
- E: پیچیدگی زمانی ارزیابی یک جواب کاندید

اندازه جدول حافظه در طول اجرای الگوریتم ثابت است. با توجه به مشخص بودن اندازه جدول حافظه، میزان مصرف حافظه قابل محاسبه است. جدول حافظه آرایه‌ای دوبعدی با طول و عرض یکسان است که برابر تعداد جمعیت جستجوی روش پیشنهادی است. با توجه به این‌که نیاز به ذخیره اعداد صحیح است، بنابراین آرایه دوبعدی را از نوع صحیح (Int) تعریف می‌کنیم. پس هر درایه جدول حافظه برابر 32 بیت (اندازه Int) است. اگر اندازه جمعیت الگوریتم پیشنهادی را n در نظر بگیریم، آنگاه میزان مصرف حافظه برابر $32n^2$ می‌شود. در آزمایش‌های صورت گرفته در پژوهش حاضر، تعداد جمعیت 100 در نظر گرفته شده است. بنابراین اندازه حافظه مصرفی برابر 40 کیلو بایت می‌شود. با توجه به افزایش روزافزون میزان حافظه کامپیوترها میزان مصرف حافظه ناچیز بوده و می‌توان از آن صرف نظر کرد.



شکل (15): نمودار مقدار خطای برازش دستگاه معادلات شماره ده در تکرارهای یک تا هزار برای الگوریتم‌های مورد بررسی

جدول (10): نتایج به دست آمده از حل دستگاه معادلات شماره ده

Var	IARO	ARO	GA	WOA	GWO	BAT	CS	ES
x_1	0	5.3576560E-17	0.6674262	0	0	-2.0518470E-06	0.6674262	0
x_2	1	1	2.21797683	1	1.00000015	1.00000225	2.21797683	1
f	0	3.8550502E-14	6.6208657E-03	1.2652397E-09	8.8176965E-11	3.8237376E-02	0	0

جدول (11): نتایج به دست آمده از حل دستگاه معادلات شماره یازده

VAR	IARO	ARO	GA	WOA	GWO	BAT	CS
x_1	-2.61803399	-2.61803399	-2.00327256	-2.65062843	-0.24901475	-2.6180458	-2.00327256
x_2	-0.38196601	-0.38196601	0.77941159	-0.37685439	-0.20915116	0.99993027	0.77941159
x_3	1	1	0.17555257	1.02038926	-0.07804303	1.00014719	0.17555257
x_4	1	1	0.05794047	0.99779137	0.59722808	0.99999066	0.05794047
x_5	1	1	4.61015403	0.98217831	-0.06127774	-0.38201569	4.61015403
f	3.1431177E-32	0.19999999	0.89831497	0.60105296	0.39998952	0.40000029	2.72744533

جدول (12): زمان اجرا برای الگوریتم‌های مورد بررسی

Case	IARO	ARO	GA	WOA	GWO	BAT	CS
1	25.58	25.94	24.89	33.05	35.36	34.56	26.87
2	26.06	26.49	26.61	29.70	31.92	31.19	24.91
3	26.01	26.15	26.81	37.89	40.83	38.14	31.34
4	27.05	27.6	26.41	53.77	60.42	54.03	48.13
5	26.66	27.1	25.57	46.56	51.90	46.39	40.08
6	24.13	24.25	24.80	21.78	24.63	24.97	20.25
7	22.5	21.79	24.99	15.16	16.84	17.30	15.16
8	24.25	23.77	26.13	21.73	24.70	25.74	20.00
9	28.35	25.27	26.68	23.77	25.97	27.42	20.77
10	23.9	22.7	25.17	15.19	17.14	19.74	15.87
11	27.64	27.77	27.52	34.77	37.13	35.63	30.61
Sum	282.13	278.83	285.58	333.37	366.84	355.11	293.99
Assigned rank	2th	1th	3th	5th	7th	6th	4th

5. نتیجه‌گیری

در این مقاله الگوریتم فراابتکاری ARO با استفاده از جدول حافظه بهبود داده شد. سپس، از الگوریتم بهبود داده شده (IARO) با خود الگوریتم ARO و چند الگوریتم فراابتکاری دیگر برای حل ده دستگاه معادلات غیرخطی مختلف مورد محک قرار گرفت. نتایج نشان داد که الگوریتم بهبودیافته، به خوبی توانایی حل عددی دستگاه معادلات غیرخطی را بدون داشتن هیچ گونه حدس اولیه دارا است. با توجه به این نکته که یک دستگاه معادله ممکن است چندین جواب داشته باشد، ملاک مقایسه مقدار خطای برازش قرار داده شد. با در نظرگرفتن این نکته روش پیشنهاد شده در همه دستگاه معادلات مورد مطالعه در مقایسه با سایر الگوریتم‌ها به صورت میانگین بهترین جواب (میانگین کمترین مقدار خطای برازش) را به دست آورده است. برای مطالعه‌های بعدی می‌توان عملکرد الگوریتم بهبود یافته مطرح شده در این مقاله را روی مسائل دیگر عددی در ریاضیات بررسی کرد.

سپاسگزاری

از جناب آقای دکتر سید علی میرجلیلی که با مشاوره و نظارت بر پژوهش ما یاری نمودند، تشکر ویژه داریم.

تعارض منافع: نویسندگان اعلام می‌کنند که هیچ تعارض منافعی ندارند.

مراجع

[1] B.M. Barbashov, V.V. Nesterenko, and A.M. Chervyakov, "General solutions of nonlinear equations in the geometric theory of the relativistic string," Commun. Math. Phys., vol. 84, no. 4, pp. 471-481, 1982, doi: 10.1007/BF01209629.

[2] S.R. Bickham, S.A. Kiselev, and A.J. Sievers, "Stationary and moving intrinsic localized modes in one-dimensional monatomic lattices with cubic and quartic anharmonicity," Phys. Rev. B, vol. 47, no. 21, pp. 14206-14211, 1993, doi: 10.1103/PhysRevB.47.14206.

[3] J.M. Ortega and R.W. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, SIAM, 2000.

[4] Y. Li, Y. Wei, and Y. Chu, "Research on solving systems of nonlinear equations based on improved PSO," Math. Probl. Eng., vol. 2015, pp. 1-13, 2015, doi: 10.1155/2015/727218.

[5] Y. Mo, H. Liu, and Q. Wang, "Conjugate direction particle swarm optimization solving systems of nonlinear equations," Comput. Math. Appl., vol. 57, no. 11-12, pp. 1877-1882, 2009, doi: 10.1016/j.camwa.2008.10.005.

[6] Y.-Z. Luo, G.-J. Tang, and L.-N. Zhou, "Hybrid approach for solving systems of nonlinear equations using chaos optimization and quasi-Newton method," Appl. Soft Comput., vol. 8, no. 2, pp. 1068-1073, 2008, doi: 10.1016/j.asoc.2007.05.013.

[7] W. Zhao, L. Wang, and S. Mirjalili, "Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications," Comput. Methods Appl. Mech. Eng., vol. 388, p. 114194, 2022, doi: 10.1016/j.cma.2021.114194.

[8] Z. Beheshti and S.M. Shamsuddin, "A review of population-based meta-heuristic algorithm," Int. J. Adv. Soft Comput. Appl., vol. 5, pp. 1-35, 2013.

[9] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization," ACM Comput. Surv., vol. 35, no. 3, pp. 268-308, 2003, doi: 10.1145/937503.937505.

[10] T. Mantere and J.T. Alander, "Evolutionary software engineering, a review," Appl. Soft Comput., vol. 5, no. 3, pp. 315-331, 2005, doi: 10.1016/j.asoc.2004.08.004.

[11] K.V. Price, "Differential evolution," in Handbook of Optimization, 2013, pp. 187-214, doi: 10.1007/978-3-642-30504-7_8.

[12] N. Hansen, "The CMA evolution strategy: A tutorial," arXiv preprint, arXiv:1604.00772, 2016.

[13] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," J. Global Optim., vol. 39, no. 3, pp. 459-471, 2007, doi: 10.1007/s10898-007-9149-x.

[14] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A gravitational search algorithm," Inf. Sci., vol. 179, no. 13, pp. 2232-2248, 2009, doi: 10.1016/j.ins.2009.03.004.

[15] A. Kaveh and S. Talatahari, "A novel heuristic optimization method: charged system search," Acta Mech., vol. 213, no. 3-4, pp. 267-289, 2010, doi: 10.1007/s00707-009-0270-4.

[16] N.F. Johari, A.M. Zain, M.H. Noorfa, and A. Udin, "Firefly algorithm for optimization problem," Appl. Mech. Mater., vol. 421, pp. 512-517, 2013, doi: 10.4028/www.scientific.net/AMM.421.512.

[17] X.S. Yang and S. Deb, "Engineering optimisation by cuckoo search," Int. J. Math. Model. Numer. Optim., vol. 1, no. 4, p. 330, 2010, doi: 10.1504/IJMMNO.2010.035430.

[18] K.M. Passino, "Bacterial foraging optimization," Int. J. Swarm Intell. Res., vol. 1, no. 1, pp. 1-16, 2010, doi: 10.4018/jsir.2010010101.

[19] A. Gogna and A. Tayal, "Metaheuristics: review and application," J. Exp. Theor. Artif. Intell., vol. 25, no. 4, pp. 503-526, 2013, doi: 10.1080/0952813X.2013.782347.

[20] W. Zhao, C. Du, and S. Jiang, "An adaptive multiscale approach for identifying multiple flaws based on XFEM and a discrete artificial fish swarm algorithm," Comput. Methods Appl. Mech. Eng., vol. 339, pp. 341-357, 2018, doi: 10.1016/j.cma.2018.04.037.

[21] J.H. Holland, "Genetic algorithms," Sci. Am., vol. 267, no. 1, pp. 66-73, 1992.

[22] E. Afarande and R. Hosseini, "An automatic Model for Managing Uncertainty and Rule Extraction in Form of Fuzzy Rules using Genetic Algorithm," Soft Comput. J., vol. 9, no. 1, pp. 14-25, 2020, doi: 10.22052/scj.2021.111449 [In Persian].

[23] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proc. ICNN'95 - Int. Conf. Neural Networks, 1995, pp. 1942-1948, doi: 10.1109/ICNN.1995.488968.

[24] S. Esfandyari and V. Rafe, "Using the Particle Swarm Optimization Algorithm to Generate the Minimum Test Suite in Covering Array with Uniform Strength," Soft Comput. J., vol. 8, no. 2, pp. 66-79, 2020, doi: 10.22052/8.2.66 [In Persian].

[25] Y. Zhang, S. Wang, and G. Ji, "A comprehensive survey on particle swarm optimization algorithm and its applications," Math. Probl. Eng., vol. 2015, pp. 1-38, 2015, doi: 10.1155/2015/931256.

[26] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 26, no. 1, pp. 29-41, 1996, doi: 10.1109/3477.484436.

[27] M.N. Ab Wahab, S. Nefti-Meziani, and A. Atyabi, "A comprehensive review of swarm optimization algorithms," PLoS One, vol. 10, no. 5, p. e0122827, May 2015, doi: 10.1371/journal.pone.0122827.

[28] X. Yang and A. Hossein Gandomi, "Bat algorithm: a novel approach for global engineering optimization," Eng. Comput., vol. 29, no. 5, pp. 464-483, Jul. 2012, doi: 10.1108/02644401211235834.

[29] K.N. Krishnanand and D. Ghose, "Detection of multiple source locations using a glowworm metaphor with applications to collective robotics," in Proc. 2005 IEEE Swarm Intell. Symp., 2005, pp. 84-91, doi: 10.1109/SIS.2005.1501606.

[30] S. Mirjalili, S.M. Mirjalili, and A. Lewis, "Grey wolf optimizer," Adv. Eng. Softw., vol. 69, pp. 46-61, 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[31] S. Arora and S. Singh, "Butterfly optimization algorithm: a novel approach for global optimization," Soft Comput., vol. 23, no. 3, pp. 715-734, 2019, doi: 10.1007/s00500-018-3102-4.

[32] R. Behmanesh and N. Majma, "Nephron-2 Meta-Heuristic Algorithm (NOA-2), to Solve Optimization Problems," Soft Comput. J., vol. 11, no. 2, pp. 62-71, 2023, doi: 10.22052/scj.2023.248427.1104 [In Persian].

[33] L. Wang, Q. Cao, Z. Zhang, S. Mirjalili, and W. Zhao, "Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems," Eng. Appl. Artif. Intell., vol. 114, p. 105082, 2022, doi: 10.1016/j.engappai.2022.105082.

[34] N. Henderson, W.F. Sacco, and G.M. Platt, "Finding more than one root of nonlinear equations via a polarization technique: An application to double retrograde vaporization," Chem. Eng. Res. Des., vol. 88, no. 5-6, pp. 551-561, 2010, doi: 10.1016/j.cherd.2009.11.001.

[35] W.F. Sacco and N. Henderson, "Finding all solutions of nonlinear systems using a hybrid metaheuristic with Fuzzy Clustering Means," Appl. Soft Comput., vol. 11, no. 8, pp. 5424-5432, 2011, doi: 10.1016/j.asoc.2011.05.016.

[36] S. Mirjalili and A. Lewis, "The whale optimization algorithm," Adv. Eng. Softw., vol. 95, pp. 51-67, 2016, doi: 10.1016/j.advengsoft.2016.01.008.

[37] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in Proc. 2009 World Cong. Nature Biol. Inspired Comput. (NaBIC), 2009, pp. 210-214, doi: 10.1109/NABIC.2009.5393690.

[38] O.E. Turgut, M.S. Turgut, and M.T. Coban, "Chaotic quantum behaved particle swarm

optimization algorithm for solving nonlinear system of equations," Comput. Math. Appl., vol. 68, no. 4, pp. 508-530, 2014, doi: 10.1016/j.camwa.2014.06.013.

[39] J.R. Sharma and H. Arora, "On efficient weighted-Newton methods for solving systems of nonlinear equations," Appl. Math. Comput., vol. 222, pp. 497-506, 2013, doi: 10.1016/j.amc.2013.07.066.

[40] H.A. e Oliveira and A. Petraglia, "Solving nonlinear systems of functional equations with fuzzy adaptive simulated annealing," Appl. Soft Comput., vol. 13, no. 11, pp. 4349-4357, 2013, doi: 10.1016/j.asoc.2013.06.018.

[41] M.A. Tawhid and A.M. Ibrahim, "Solving nonlinear systems and unconstrained optimization problems by hybridizing whale optimization algorithm and flower pollination algorithm," Math. Comput. Simul., vol. 190, pp. 1342-1369, 2021, doi: 10.1016/j.matcom.2021.07.010.