ACPSO: یک الگوریتم جدید بهینه‌سازی گروه ذرات تعاونی با قابلیت به روزرسانی تطبیقی پارامترها

الهام شکرانی‌پور[۱]، امیرمسعود افتخاری مقدم[۲]
۱- دانشگاه آزاد اسلامی- واحد قزوین- باشگاه پژوهشگران جوان - قزوین- ایران– el_shokranipour@yahoo.com
۲- دانشکده مهندسی برق رایانه و فن‌آوری اطلاعات- دانشگاه آزاد اسلامی- قزوین- ایران– eftekhari@qiau.ac.ir

چکیده: **این مقاله یک الگوریتم بهینه‌سازی تعاونی PSO[1] به نام ACPSO[2] را ارائه می‌دهد که پارامترهای وزن اینرسی و ضرایب شتاب را برای هر بعد در هر حلقه به صورت تطبیقی به‌روزرسانی می‌کند. ACPSO با ترکیب دو الگوریتم APSO[3] و CPSO[4] از مزایای ساختار تطبیقی APSO و ساختار تعاونیCPSO به طور هم‌زمان بهره می‌برد. ساختار تطبیقی ACPSO باعث می‌شود در هر مرحله از اجرای الگوریتم پارامترها با مناسب‌ترین مقدار خود، معادله سرعت را به‌روزرسانی کنند تا در نهایت، الگوریتم در تکرارهای کمتری به جواب رسیده و سرعت هم‌گرایی افزایش یابد. ساختار تعاونی ACPSO باعث می‌شود: ۱) برای حل مسأله‌های با ابعاد بالا مفید باشد؛ ۲) با افزایش تنوع جمعیت، از گیر افتادن در بهینه محلی جلوگیری کرده و نرخ هم‌گرایی را بهبود بخشد و ۳) بر خلاف روش‌های دیگر که ممکن است برخی از مؤلفه‌ها را بدتر و بقیه را بهتر کنند، ACPSO در هر مرحله کلیه ابعاد مسأله را بهبود دهد. ACPSO در مقایسه با سایر روش‌ها که ساختار غیرتعاونی دارند و پارامترها را به صورت ثابت، متغیر با زمان یا تطبیقی مقداردهی می‌کنند، در بهینه‌سازی توابع محک استاندارد تک‌قله‌ای و چندقله‌ای به پاسخ‌های بسیار بهتری رسیده است. همچنین مقایسه روند تغییرات پاسخ‌ها نشان می‌دهد که ACPSO سریع‌تر از سایر روش‌ها حتی روش تعاونی غیر تطبیقی هم‌گرا می‌شود.**

واژه‌های کلیدی: **بهینه‌سازی گروه ذرات، به‌روزرسانی تطبیقی پارامترها، بهینه‌سازی گروه ذرات تعاونی.**

ACPSO: a new cooperative particle swarm optimization algorithm with adaptive updating parameters

Elham Shokranipour, Amir Masood Eftekhari Moghadam

Abstract: This paper presents a particle swarm optimization algorithm named ACPSO that adaptively updates inertia weight and acceleration coefficients of each dimension for every particle in each cycle. ACPSO simultaneously uses the advantages of adaptive structure of APSO and cooperative structure of CPSO by combining them.

Adaptive structure of ACPSO allows updating velocity equation with most appropriate value of parameters in each cycle so that response will produce in less iteration. Cooperative structure of ACPSO has caused to 1) be useful to solve high dimensional problems; 2) prevents trapping in local optimum using increases the population diversity; 3) improves all dimensions of problem in each cycle unlike other methods that may be in some of the components are better and in others worse; application of the new ACPSO algorithm on several benchmark optimization problems shows a marked improvement of performance rather than other methods that have non cooperative structure with constant, time varying or adaptive parameters initializing. Compare of convergence speed shows that convergence speed of ACPSO is faster than other methods even non adaptive cooperative method.

Keywords: Particle Swarm Optimization, adaptive updating parameters, Cooperative Particle Swarm Optimization.

تاریخ ارسال مقاله: ۱۳۸۹/۶/۹
تاریخ اصلاح مقاله: ۱۳۸۹/۱۲/۱۴
تاریخ پذیرش مقاله: ۱۳۸۹/۱۲/۲۲
نام نویسنده مسئول: الهام شکرانی‌پور
نشانی نویسنده مسئول : قزوین – بلوار نخبگان – دانشگاه آزاد اسلامی واحد قزوین – دانشکده مهندسی برق رایانه و فن‌آوری اطلاعات

۱- مقدمه

اصطلاح تکاملی که در الگوریتم‌های تکاملی استفاده شده است، می‌تواند به چندین نوع مختلف از گونه‌هایی که در یک محیط زندگی می‌کنند اطلاق شود. زمانی‌که بیش از یک نوع موجود زنده وجود داشته باشد آن‌ها می‌توانند دو نوع رابطه رقابتی یا تعاونی با یکدیگر داشته باشند. طراحی الگوریتم تکاملی رقابتی معمولا با طراحی تابع شایستگی رقابتی آغاز می‌شود. چنین تابعی میزان کیفیت یک پاسخ را زمانی‌که با دیگر افراد رقابت می‌کند، اندازه‌گیری می‌نماید اما زمانی‌که افزایش شایستگی یک گونه منجر به افزایش شایستگی گونه‌های دیگر شود، رابطه آن‌ها هم‌زیستی تعاونی گفته می‌شود. یک مثال طبیعی از این نوع می‌تواند رابطه میان بعضی از حشرات و گل‌ها باشد. کلیِرواترِ و همکارانش تعاون را به صورت زیر بیان کردند [۳۱]:

"تعاون با یک مجموعه از عامل‌ها درگیر است که به‌وسیله تبادل اطلاعات با یکدیگر در زمان حل مسأله دنبال می‌شود اما اطلاعاتی که میان عامل‌ها مبادله می‌شود ممکن است غلط باشد و باید بعضی اوقات رفتار عامل دریافت کننده آن را تغییر دهد."

بر اساس تعریف فوق، الگوریتم‌های بسیاری در گروه الگوریتم‌های تعاونی دسته‌بندی می‌شوند. در ادامه به برخی از الگوریتم‌های تعاونی اشاره می‌شود.

۱-۱- الگوریتم‌های تکاملی تعاونی

یک الگوریتم ژنتیک استاندارد به جای این‌که یک الگوریتم رقابتی در نظر گرفته شود می‌تواند با استفاده از تعریف کلیِرواتِر به عنوان یک الگوریتم تعاونی دیده شود. اگر افراد در یک جمعیت الگوریتم ژنتیک به عنوان عامل‌ها در نظر گرفته شده و عملگر بازترکیبی به عنوان تبادل اطلاعات باشد؛ آنگاه الگوریتم ژنتیک به طور قطعی یک الگوریتم تعاونی است. این دیدگاه به‌وسیله محققان دیگر نیز به خوبی دنبال شده و نشان می‌دهد که تحت تنظیم پارامترهای مشخص، الگوریتم ژنتیک به راستی یک یادگیر تعاونی است[۳۲].

از یک دیدگاه خاص، الگوریتم بهینه‌سازی گروه ذرات (PSO) شکل دیگری از الگوریتم ژنتیک است[۳۲]. بنابراین الگوریتم بهینه‌سازی گروه ذرات استاندارد نیز می‌تواند به عنوان یک الگوریتم تعاونی در نظر گرفته شود که عامل‌ها در آن همان ذرات جمعیت هستند و تبادل اطلاعات میان عامل‌ها از طریق ترکیب هر ذره با بهترین ذره گروه در هنگام به‌روزرسانی سرعت آن ذره انجام می‌شود. با این وجود برای الگوریتم‌های ژنتیک و بهینه‌سازی گروه ذرات نسخه‌های تعاونی آن‌ها ارائه شده است که ساختار آن‌ها در ادامه آورده شده است.

۱-۲- الگوریتم‌های ژنتیک تعاونی

الگوریتم‌های ژنتیک تعاونی اولین بار به وسیله پاتِر معرفی شدند [۳۰ و ۲۶]. در مدل او راه‌حل‌های پیچیده باید به شکل زیرمؤلفه‌های انفعالی قابل تطبیق با یکدیگر و به صورت یک جمعیت جداگانه در الگوریتم ژنتیک خصوصی خودشان باز شوند. در حقیقت شایستگی یک زیرمؤلفه نمی‌تواند ارزیابی شود مگر این‌که بعضی از مفاهیم پیوندی وجود داشته باشد. مفاهیم به‌وسیله انتخاب اعضای نماینده از دیگر زیرجمعیت‌ها و ساختن یک بردار قالب پاسخ، ساخته می‌شود که یک زیر مؤلفه را ندارد. با استفاده از این قالب هر فرد از یک گونه خاص می‌تواند در محتوای بردار قالب پاسخ ارزیابی شود تا شایستگی آن محاسبه گردد. پاتِر دو نوع الگوریتم تکاملی تعاونی ژنتیک ($CCGA^5$) به نام‌های CCGA-1 و CCGA-2 را مطرح کرد[۳۰ و ۲۶] که در اولی مؤلفه‌های بردار قالب پاسخ را بهترین اعضای زیرجمعیت‌های معادل تشکیل می‌دهند و در نوع دوم این امکان وجود دارد که مؤلفه‌ها به صورت تصادفی از زیرجمعیت مربوط انتخاب شوند.

۱-۳- الگوریتم‌های بهینه‌سازی گروه ذرات تعاونی

الگوریتم بهینه‌سازی گروه ذرات تعاونی CPSO یک نسخه جدید از الگوریتم PSO است که از یک رفتار تعاونی برای بهبود کارآیی الگوریتم استاندارد PSO بهره می‌برد. PSO استاندارد از یک جمعیت با بردارهای n-بعدی استفاده می‌کند. این بردارها در الگوریتم CPSO به n دسته از بردارهای ۱-بعدی قسمت‌بندی می‌شوند. هر دسته تلاش می‌کند تا یک مؤلفه یکتا از بردار جواب را بهینه کند؛ که در اصل یک مسأله بهینه‌سازی یک بعدی است.

اگر بعضی از مؤلفه‌ها وابسته باشند، آن‌ها باید با استفاده از یک روش بخش‌بندی قراردادی در دسته‌های یکسانی هم‌گروه شوند؛ زیرا تغییرات مستقلی که به وسیله دسته‌های مختلف اعمال می‌شود، اثر مخربی بر روی متغیرهای وابسته دارد. نتیجه این کار وجود بعضی از دسته‌ها با بردارهای یک بعدی و مابقی دسته‌ها با بردارهای c-بعدی ($c<n$) است. متاسفانه مؤلفه‌های وابسته همیشه قابل تشخیص نیستند.

به کار بردن روش پاتِر بر روی PSO منجر به تولید دو مدل PSO تعاونی جدید به نام‌های $CPSO\text{-}S_K$ و $CPSO\text{-}H_K$ شده است [۱۷]. الگوریتم $CPSO\text{-}S_K$ یک بردار را به K بخش تجزیه می‌کند.

بخش‌بندی فضای جستجو یک اثرجانبی به نام شبه کمینه را ایجاد می‌کند. شبه کمینه به نقطه‌ای از فضای جستجو اطلاق می‌شود که یک کمینه محلی در تمام زیر فضاهای از پیش تعریف شده باشد؛ اما یک کمینه محلی در یک فضای جستجوی کامل n-بعدی نباشد. الگوریتم $CPSO\text{-}S_K$ ممکن است در شبه کمینه گیر کند، به همین دلیل الگوریتم دیگری به نام $CPSO\text{-}H_K$ معرفی شد که الگوریتم $CPSO\text{-}S_K$ را با الگوریتم استاندارد PSO ترکیب می‌کرد [۱۷].

افصحی و میبدی الگوریتمی به نام FCPSO-H را ارائه کرده‌اند [۷] که از منطق فازی برای کنترل ضرایب شتاب در الگوریتم $CPSO\text{-}H_K$ استفاده می‌کند.

در [۱] الگوریتمی تحت عنوان EGPSO ارائه شده است که از یک چارچوب مبتنی بر گروه برای ترکیب کردن عملگرهای بازترکیبی و جهش با بهینه‌سازی گروه ذرات استفاده می‌کند. EGPSO گروه‌های

مختلفی را بر اساس شایستگی ذرات برای انتخاب والدین، به‌روزرسانی ذرات و جایگزینی آن‌ها به صورت پویا تشکیل می‌دهد. گروه‌ها در هر تکرار به صورت تطبیقی تشکیل یافته و تعداد آن‌ها در هر تکرار ممکن است متفاوت باشد.

در ادامه این مقاله، مسأله مورد بررسی و رویکرد روش پیشنهادی ارائه شده است. الگوریتم‌های PSO استاندارد و تعاونی در بخش ۳ معرفی شده و پارامترهای مختلف آن‌ها مورد بررسی قرار گرفته است. در بخش ۴، الگوریتم پیشنهادی ACPSO مطرح شده است. در بخش ۵ این الگوریتم شبیه‌سازی شده و با سایر روش‌های مشابه مقایسه گردیده و از نظر کیفیت و سرعت رسیدن به پاسخ، مورد بررسی قرار گرفته‌است.

۲- مسأله مورد بررسی و رویکرد روش پیشنهادی

همان‌طور که در بخش ۱-۳ گفته شد یک الگوریتم n-بعدی PSO تعاونی به چندین الگوریتم ۱-بعدی PSO تجزیه می‌شود، بنابراین سرعت اجرای یک الگوریتم PSO تعاونی چند برابر سرعت اجرای یک الگوریتم PSO استاندارد خواهد شد. در این مقاله رویکرد ACPSO پیشنهاد شده است تا سرعت رسیدن به جواب را در یک الگوریتم PSO تعاونی بهبود دهد. برای این کار در هر حلقه از تکرار الگوریتم، برای هر دسته یک بعدی، وضعیت تکاملی ذرات آن دسته بررسی می‌شود. سپس با توجه به این‌که جمعیت ذرات در کدام وضعیت تکاملی قرار داشته باشد، پارامترهای وزن اینرسی و ضرایب شتاب به صورت تطبیقی برای آن دسته مقداردهی می‌شوند. به این ترتیب در هر مرحله از اجرای الگوریتم، پارامترها با مناسب‌ترین مقدار خود معادله سرعت را به‌روزرسانی می‌کنند؛ در نتیجه هر ذره با سرعت مناسبی به سمت هدف حرکت می‌کند. این امر در نهایت باعث می‌شود الگوریتم در تکرارهای کمتری به پاسخ رسیده و سرعت رسیدن به پاسخ افزایش یابد. علاوه بر این، ساختار تعاونی ACPSO از مزایای ویژه‌ای برخودار است؛ از جمله:

اول: کارآیی الگوریتم ACPSO برای حل مسائل با ابعاد بالا بسیار بهتر از PSO استاندارد است؛ زیرا ابعاد را جداگانه مورد بررسی قرار می‌دهد. بسیاری از الگوریتم‌های بهینه‌سازی تصادفی (PSOها و الگوریتم‌های ژنتیک) از *معضل ابعاد*[۶] رنج می‌برند به طوری‌که با افزایش ابعاد مسأله، کارآیی آن‌ها کاهش می‌یابد. احتمال تولید یک نمونه داخل ناحیه بهینه برابر است با حجم ناحیه بهینه تقسیم بر حجم کل فضای جستجو. این احتمال با افزایش ابعاد فضای جستجو به صورت نمایی کاهش می‌یابد. با این تعریف واضح است که پیدا کردن بهینه عمومی برای یک مسأله با ابعاد بالا در مقایسه با یک مسأله با ابعاد پایین و همان ساختار، بسیار مشکل‌تر است. یک راه غلبه بر معضل ابعاد، بخش‌بندی فضای جستجو به زیرفضاهایی با ابعاد کوچکتر است بنابراین ما در الگوریتم ACPSO ابعاد را از هم جدا کردیم تا الگوریتم بهینه‌سازی بتواند جستجوی تمام نواحی ممکن در فضای جستجو را تضمین کند.

دوم: در PSO استاندارد به بهبود کل بردار پاسخ توجه می‌شود حتی اگر بعضی از مؤلفه‌های پاسخ بدتر شده باشند. در چنین حالتی اگر اثر مؤلفه‌های بهبود یافته بر اثر مؤلفه‌هایی که بدتر شده‌اند، غلبه کند. آنگاه شایستگی بهتری برای بردار در نظر گرفته می‌شود. اما در ACPSO چنین حالتی اتفاق نمی‌افتد؛ زیرا فرکانس ارزیابی بردار پاسخ بسیار بیشتر است به طوری که با هر بار جایگزینی یک زیر مؤلفه در بردار پاسخ، نرخ خطا ارزیابی می‌شود.

سوم: الگوریتم ACPSO با افزایش تنوع جمعیت نسبت به PSO استاندارد، هم‌گرایی را بهبود می‌بخشد. زمانی که ابعاد در روش تعاونی از یکدیگر جدا می‌شوند، تنوع جمعیت زیاد می‌شود؛ زیرا جایگزینی ذرات مختلف از دسته‌های متفاوت، ترکیب‌های بسیار زیادی را ایجاد می‌کند که شانس الگوریتم را در پیدا کردن پاسخ بهینه افزایش می‌دهد.

۳- الگوریتم‌های PSO و CPSO

PSO برای اولین بار به‌وسیله کِندی و اِبرهارت در سال ۱۹۹۵ مطرح شد[۲۹ و ۲۸]. یک روش بهینه‌سازی مبتنی بر جمعیت است. مفهومی ساده و پارامترهای کمی دارد که به راحتی قابل پیاده‌سازی است و برای توابع غیرخطی پیوسته، شبکه‌های عصبی، مسائل بهینه‌سازی غیرخطی و غیره با موفقیت به کار برده شده است [۱۶]. PSO از گروه پرندگان یا ماهیان الهام گرفته شده است. دانشمندان سناریو جستجوی پرندگان برای غذا را شبیه‌سازی کردند. آن‌ها دریافتند که هر پرنده سرعتش را بر اساس دو عامل مشخص می‌کند: بهترین تجربه قبلی خودش و بهترین تجربه همه پرندگان دیگر. این کار مشابه رفتار انسان‌ها در تصمیم گیری است. این الگوریتم به اختصار در ادامه بیان شده است.

۳-۱- الگوریتم PSO

الگوریتم PSO یک جمعیت با M ذره d بعدی را مقداردهی اولیه می‌کند [۲۹]. هر ذره دارای یک بردار موقعیت:

$$\vec{X}_i = \left(X_i^1,\ X_i^2, \dots, X_i^d\right) \quad (۱)$$

یک بردار سرعت:

$$\vec{V}_i = \left(V_i^1,\ V_i^2, \dots, V_i^d\right) \quad (۲)$$

و یک حافظه از بهترین وضعیت قبلی خودش (*pbest*) است.

$$\vec{P}_i = \left(P_i^1,\ P_i^2, \dots, P_i^d\right) \qquad i = 1, 2, \dots, M \quad (۳)$$

بردار $\vec{P}_g$ نیز به طریق مشابه برای بهترین موقعیت در همسایگی به دست می‌آید. در نسخه PSO عمومی، بهترین ذره (*gbest*) در کل جمعیت محاسبه می‌شود؛ اما در نسخه محلی آن، برای هر ذره، بهترین ذره در همسایگی آن ذره محاسبه می‌شود. در هر تکرار، $\vec{P}_g$ (بهترین تجربه قبلی گروه) و بردار $\vec{P}_i$ ذره فعلی (بهترین تجربه قبلی فرد) با هم

```
Define: b(j, z) = [P1.gbest, P2.gbest, ..., Pj-1.gbest, z, Pj+1.gbest, ..., Pn.gbest]
Create and initialize n one-dimensional PSOs: Pj, j = 1… n
While stopping condition is false Do
        For each swarm j = 1 .. n
          For each particle i = 1 .. s
            If fitness (b (j, Pj.xi)) < fitness (b (j, Pj.pbesti))
              Then Pj.pbesti = Pj.xi
            End_If
            If fitness (b (j, Pj.pbesti)) < fitness (b (j, Pj.gbest))
                Then Pj.gbest = Pj.pbesti
      End_If
          End_For
          Update each Pj using eqs. (5-6)
        End_For
End_While
b (j, Pj.gbest) and fitness (b (j, Pj.gbest)) are outputs
```

شکل (۱): شبه کد الگوریتم تعاونی CPSO [۱۷]

ترکیب می‌شوند تا سرعت ذره را در امتداد هر بعد تطبیق دهند. آن بخش از تطبیق سرعت که تحت تاثیر بهترین موقعیت قبلی است به عنوان بخش شناخت[۷] و آن بخش که تحت تاثیر بهترین در همسایگی است به عنوان بخش اجتماعی[۸] بررسی می‌شود. معادله (۴) فرمول تغییر بعد dام بهترین وضعیت قبلی ذره را در یک کار کمینه‌سازی نشان می‌دهد که f در آن تابع شایستگی است.

$$P_i^d(t+1)=\begin{cases} P_i^d(t) & if\ f\left(\vec{X}_i(t+1)\right)\geq f(\vec{P}_i(t)) \\ X_i^d(t+1) & if\ f\left(\vec{X}_i(t+1)\right)< f(\vec{P}_i(t)) \end{cases} \quad (۴)$$

در تکرار t، بُعد dام سرعت و موقعیت ذره i به ترتیب با معادلات (۵) و (۶) تغییر می‌کنند [۲۴]. که در آن w وزن اینرسی، c_1 و c_2 ضرایب شتاب نام دارند که هر سه پارامترهای ثابت حقیقی غیر منفی هستند$r_{1,i}^d(t)$ و $r_{2,i}^d(t)$ دو عدد تصادفی مستقل با توزیع یکنواخت در محدوده [0 1] هستند:

$$V_i^d(t+1)=wV_i^d(t)+c_1r_{1,i}^d(t)(P_i^d(t)-X_i^d(t))+c_2r_{2,i}^d(t)(P_g^d(t)-X_i^d(t)) \quad (۵)$$

$$X_i^d(t+1)=X_i^d(t)+V_i^d(t+1) \quad (۶)$$

تا زمان رسیدن به شرط توقف به ترتیب بردارهای $pbest$،$gbest$ و در امتداد آن سرعت و موقعیت به‌روزرسانی خواهند شد و در نهایت نیز بردار $gbest$ جواب مسأله خواهد بود.

۳-۲- الگوریتم CPSO

پیش از این (بخش ۱-۳) در مورد الگوریتم بهینه‌سازی گروه ذرات تعاونی گفته شد که برای حل یک مسأله n-بعدی، CPSO از n الگوریتم ۱-بعدی PSO استفاده می‌کند. هر PSO یک بعدی، تلاش می‌کند تا یک مؤلفه یکتا از بردار جواب را بهینه کند که در اصل یک مسأله بهینه‌سازی ۱-بعدی است. یک پیچیدگی این تنظیم، در حقیقت تابعی است که باید کمینه شود، f به یک بردار n-بعدی به عنوان ورودی نیازمند است. اگر هر دسته تنها یک بعد واحد از فضای جستجو را نمایش دهد، واضح است که محاسبه مستقیم شایستگی هر ذره جمعیت اصلی امکان‌پذیر نیست. یک بردار مفهوم[۹] نیاز است تا هر ذره جمعیت بتواند در آن ارزیابی شود [۱۱]. نمای ساده چنین بردار مفهومی برای ساختن ذره بهینه عمومی این طور است که ذره بهینه عمومی هر کدام از n دسته گرفته شده و به شکل یک بردار n-بعدی به یکدیگر متصل می‌شوند. برای محاسبه شایستگی تمام ذرات در دسته j، زمانی‌که مؤلفه jام بردار مفهوم یکی یکی با هر ذره دسته jام جابجا می‌شود، n-1 مؤلفه دیگر بردار مفهوم ثابت نگاه داشته می‌شوند (با مقادیری که برای ذرات بهینه عمومی از n-1 دسته دیگر تخصیص داده شده‌اند.). شکل ۱ شبه کد الگوریتم CPSO را نشان می‌دهد.

۳-۳- پارامترهای PSO

معمولا PSO نیاز دارد تا ضرایب عددی آن شامل مقدار بیشینه سرعت، وزن اینرسی (w)، عامل شناخت (c_1)، عامل اجتماعی (c_2)، اندازه جمعیت و ساختار آن از قبل مشخص شود. توانایی بهینه‌سازی عمومی به میزان قابل توجهی به تنظیم این پارامترها بستگی دارد [۱۸]. در ادامه نقش هر یک از این پارامترها در هم‌گرایی PSO بیان شده است.

بیشینه سرعت بر روی توانایی گذر کردن از بهینه محلی و بهینه-یابی عمومی مجدد تاثیر می‌گذارد. در این مقاله همان‌طور که در [۲۱]. نشان داده شده است بیشینه سرعت برابر با ۲۰٪ محدوده جستجوی اولیه، مقداردهی شده است.

اندازه جمعیت میان بهینه‌سازی عمومی و هزینه محاسباتی تعادل ایجاد می‌کند.

ساختار، توانایی به اشتراک گذاری اطلاعات و هزینه ارتباط را بررسی می‌نماید.

وزن اینرسی به‌وسیله شای و اِبرهارت به عنوان یک مکانیسم کنترل توانایی‌های اکتشاف[۱۰] و استخراج[۱۱] ذرات معرفی شد[۲۴]. یک وزن اینرسی بزرگ، جستجوی عمومی را و یک وزن اینرسی کوچک،

جستجوی محلی را آسان می‌نماید. ایجاد تعادل میان جستجوی عمومی و محلی در خلال یک اجرا، نقشی بحرانی در موفقیت یک الگوریتم بهینه‌سازی ایفا می‌کند. تمام الگوریتم‌های تکاملی از روش‌های مختلفی برای این کار استفاده می‌کنند، برای مثال: اندازه قدم‌های جهش نرمال در استراتژی تکاملی و پارامتر درجه حرارت در گرماسازی شبیه‌سازی[۱۲] شده [۳۳] نمونه‌هایی از پارامترهای کنترلی تعادل هستند. سه روش مختلف برای مقداردهی پارامتر وزن اینرسی در الگوریتم‌های مختلف PSO وجود دارد:

۱. *وزن‌های اینرسی ثابت و تصادفی*. معمولا در پیاده‌سازی‌ها پارامتر w را در محدوده[۱ ۰]. مقداردهی می‌کنند. شای و همکارانش مقدار پارامتر w را در محدوده[۱/۲ ۰/۸] پیشنهاد دادند[۲۵]. تا الگوریتم PSO بتواند بهینه عمومی را در خلال تعداد معقولی تکرار پیدا کند. وزن اینرسی تصادفی برای دنبال کردن بهینه در یک محیط پویا پیشنهاد شده است [۲۰].

۲. *وزن‌های اینرسی متغیر با زمان*. در این روش مقدار وزن اینرسی در هر تکرار بر اساس شماره آن تکرار مشخص می‌شود. تغییرات وزن اینرسی نسبت به زمان (تکرارها) می‌تواند خطی یا غیرخطی و کاهشی یا افزایشی باشد. در فرآیند جستجوی بهینه عمومی، بهتر است در ابتدای کار، جستجو عمومی باشد و بعد از آن‌که محدوده تقریبی بهینه کشف شد، جستجو به صورت محلی دنبال شده تا بتوان بهینه عمومی را استخراج کرد. به این ترتیب به نظر می‌رسد تغییرات کاهشی وزن اینرسی نسبت به تغییرات افزایشی آن موفق‌تر باشد. در[۲] یک رویکرد موثر براساس بهینه‌سازی گروه ذرات با کدشدگی صحیح مطرح شده است که از تغییرات کاهشی خطی وزن اینرسی استفاده می‌کند. در[۳] نیز از یک الگوریتم بهینه‌سازی گروه ذرات برای مدل‌سازی استفاده شده است که وزن اینرسی را به صورت خطی کاهش می‌دهد. همچنین مقدار وزن اینرسی در [۲۳و ۹ و ۵] به صورت خطی و در [۱۵] به صورت غیر خطی کاهش یافته است. در [۴] دو پارامتر انتقال و تاخیر را معرفی شده است تا بتوان وزن اینرسی را برای تمام ذرات به صورت غیر خطی کاهش داد.

۳. *وزن‌های اینرسی تطبیقی*. در این روش الگوریتم با استفاده از برخی پارامترهای فیدبک، محیط را بازبینی کرده و مقدار مناسب وزن اینرسی را مقدار دهی می‌کند. در [۲۲ و ۱۴] از بهترین شایستگی تا کنون یافت شده برای تطبیق وزن اینرسی به‌وسیله یک شیوه فازی استفاده شده است. در[۱۰] با استفاده از شایستگی تکرارهای فعلی و قبلی، محیط بازبینی شده اما در [۸] این کار را با استفاده از ترتیب ذرات انجام شده است.

عامل شناخت، ذره را به طرف بهترین موقعیت گذشته خود سوق می‌دهد، به جستجوی محلی قله‌ها کمک کرده و تنوع جمعیت را حفظ می‌کند. در [۱۹] از یک استراتژی خودکار برای تغییر با زمان ضرایب شتاب PSO استفاده شده است تا عامل شناخت را به صورت خطی کاهش دهد. هدف آن‌ها بهبود جستجوی عمومی در قسمت ابتدایی بهینه‌سازی و تشویق ذرات به هم‌گرایی به سمت بهینه عمومی در پایان جستجو است. در [۱۲]. یک PSO با به‌روزرسانی پویا مطرح شده است که در آن از یک تکامل با تغییرات زمانی غیرخطی کاهشی برای عامل شناخت استفاده شده است. موضوع اصلی در روش آن‌ها، رسیدن به سرعت هم‌گرایی بیشتر و دقت جواب بهتر با سربار محاسباتی کمتر است.

عامل اجتماعی، تاثیر گروه را بازنمایی کرده، جمعیت را به سمت بهترین ناحیه عمومی فعلی هم‌گرا کرده و سرعت هم‌گرایی را بهبود می‌بخشد. در[۲۶] پارامتر اجتماعی c_2 به صورت خطی افزایش یافته اما در[۱۲] این افزایش به صورت غیر خطی انجام شده است.

در بخش ۳ الگوریتم‌های استاندارد و تعاونی PSO مطرح شدند و اثر هر یک از پارامترهای الگوریتم مورد بررسی قرار گرفتند. به نظر می‌رسد مقداردهی بهینه هر یک از این پارامترها می‌تواند در کیفیت و سرعت رسیدن به پاسخ، موثر باشد. در بخش ۴ استفاده از یک روش تطبیقی را برای مقداردهی بهینه پارامترهای وزن اینرسی، عامل شناخت و عامل اجتماعی بر روی نسخه تعاونی PSO پیشنهاد شده است.

۴- الگوریتم پیشنهادی بهینه‌سازی گروه ذرات تعاونی تطبیقی ACPSO

در بخش ۳ نشان داده شد که یک الگوریتم n-بعدی PSO تعاونی، از n الگوریتم ۱-بعدی PSO تشکیل شده است. بنابراین برای حل یک مسأله n-بعدی، سرعت اجرای یک الگوریتم PSO تعاونی چندین برابر سرعت اجرای یک الگوریتم PSO خواهد بود. این مقاله به دنبال راه‌کاری است تا این نقص الگوریتم PSO تعاونی را جبران کرده و سرعت اجرای الگوریتم را افزایش دهد. به نظر می‌رسد که اگر برای هر ذره مقدار مناسبی به هر یک از پارامترهای به‌روزرسانی سرعت آن ذره اختصاص داده شود، آنگاه آن ذره با سرعت بهتری به سمت بهینه مطلق حرکت خواهد کرد تا جایی که حتی ممکن است از بهینه‌های محلی سر راه خود عبور کرده و در نهایت به پاسخ بهتری دست یابد. به این ترتیب الگوریتم در تکرارهای کمتری به پاسخ خواهد رسید و سرعت اجرای آن افزایش پیدا خواهد کرد.

در این مقاله به منظور مقداردهی بهینه پارامترهای سرعت در الگوریتم PSO تعاونی، از روش ارائه شده در [۶] استفاده شده است (لازم به ذکر است این کار را می‌توان با سایر روش‌های به‌روزرسانی تطبیقی پارامترها نیز انجام داد.) و به صورت تطبیقی پارامترهای وزن اینرسی، عامل شناخت و عامل اجتماعی را در الگوریتم CPSO به‌روزرسانی شده و الگوریتم PSO تعاونی جدید ACPSO نام‌گذاری شده است. شکل ۲ شبه کد الگوریتم ACPSO را نشان می‌دهد.

به طور کلی، یک الگوریتم PSO از ابتدای اجرا تا مرحله رسیدن به پاسخ از چهار وضعیت تکاملی زیر عبور می‌کند:

```
Define: b(j, z) = [P1.gbest, P2.gbest, ..., Pj-1.gbest, z, Pj+1.gbest, ..., Pn.gbest]
Create and initialize n one-dimensional PSOs: Pj, j = 1… n
While stopping condition is false Do
    For each swarm j = 1 … n
        For each particle i = 1 … s
            If fitness (b (j, Pj.xi)) < fitness (b (j, Pj.pbesti))
                Then Pj.pbesti = Pj.xi
        End_If
            If fitness (b (j, Pj.pbesti)) < fitness (b (j, Pj.gbest))
                Then Pj.gbest = Pj.pbesti
    End_If
            If fitness (b (j,Pj.xi)) < fitness (b (j,Pj.xcur_best))
                Then cur_best = i
    End_If
        End_For
        F = Evolutionary Factor()
        Estimate the Evolutionary State from F
        [wnew c1new c2new] = Adaptive Parameters (State)
        Update each Pj .velocity with new parameters using eq. (5)
        Restrict each Pj .velocity between [-vmax vmax]
        Update each Pj .x with new parameters using eq. (6)
    End_For
End_While
b (j, Pj.gbest) and fitness (b (j, Pj.gbest)) are outputs.
```

شکل (۲): شبه کد الگوریتم ACPSO

۱) **اکتشاف:** در تکرارهای آغازین الگوریتم رخ می‌دهد. در این وضعیت تمام ذرات در سراسر محیط به طور یکنواخت پخش هستند.

۲) **استخراج:** وضعیتی است که ذرات تقریبا گروه‌بندی شده‌اند.

۳) **هم‌گرایی:** وضعیتی است که تمام ذرات در اطراف بهینه عمومی جمع شده و همگی در یک‌جا متمرکز هستند.

۴) **پرش:** وضعیتی است که بهینه عمومی جابه‌جا شده در حالی‌که تمام ذرات در اطراف آن جمع شده بودند.

این چهار وضعیت به ترتیب و پشت سر هم رخ می‌دهند. در ACPSO مقدار وزن اینرسی و ضرایب شتاب بر اساس این‌که الگوریتم در کدام یک از این چهار وضعیت باشد تغییر می‌کند بنابراین نیازمند شناسایی وضعیت تکاملی الگوریتم هستیم. این کار با بررسی اطلاعات توزیع جمعیت در هر تکرار انجام می‌شود. این اطلاعات به‌وسیله محاسبه عامل تکامل به دست می‌آید. برای محاسبه عامل تکامل باید میانگین فاصله اقلیدسی موقعیت فعلی هر ذره با تمام ذرات دیگر در دسته مورد نظر محاسبه شود [۶].

$$d_{\mathrm{i}} = \frac{1}{N-1}\sum_{j=1,j\neq i}^{N}\sqrt{(x_i - x_j)^2} \qquad (۷)$$

N بیانگر اندازه جمعیت است (دقت شود که در این فرمول ذرات x یک بعدی هستند). سپس میانگین فاصله تمام ذرات با یکدیگر مقایسه شده تا بیشترین میانگین فاصله ($d_{\max}$) و کمترین میانگین فاصله ($d_{\min}$) مشخص شوند. $d_{\mathrm{cur\_best}}$ نشان دهنده میانگین فاصله بهترین ذره دسته است. *عامل تکامل* به روش زیر محاسبه می‌شود [6]:

$$f = \frac{d_{cur_{best}} - d_{min}}{d_{max} - d_{min}} \in [0\ 1] \qquad (۸)$$

این مرحله تحت عنوان تابع *Evolutionary Factor*() در شکل ۲ نشان داده شده است.

با استفاده از عامل تکامل می‌توان تخمین زد که الگوریتم در کدام یک از چهار وضعیت تکاملی قرار دارد. f در یکی از مجموعه‌های S_1, S_2, S_3 و S_4 به صورت فازی کلاسه‌بندی شده که به ترتیب نشان دهنده وضعیت‌های اکتشاف، استخراج، هم‌گرایی و پرش هستند. بنابراین مجموعه‌های S_1, S_2, S_3 و S_4 به توابع عضویت فازی نسبت داده می‌شوند. نمودار شکل ۳ مقدار تابع عضویت را بر اساس عامل تکامل (f) نشان می‌دهد. محاسبه مقدار توابع عضویت به روش زیر انجام می‌شود:

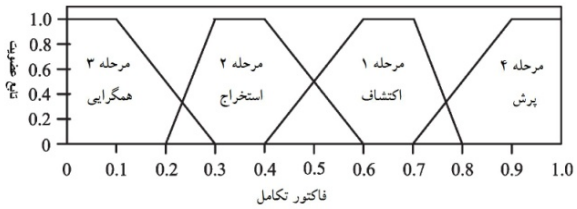
- اکتشاف: یک مقدار متوسط تا بزرگ f نشان دهنده مجموعه S_1 است. معادله ۹ تابع عضویت در وضعیت اکتشاف را نشان می‌دهد.

$$\mu_{S_1}(f) = \begin{cases} 0 & 0 \le f \le 0.4 \\ 5\times f - 2 & 0.4 < f \le 0.6 \\ 1 & 0.6 < f \le 0.7 \\ -10\times f + 8 & 0.7 < f \le 0.8 \\ 0 & 0.8 < f \le 1 \end{cases} \qquad (۹)$$

- استخراج: یک مقدار کوچک f نشان دهنده مجموعه S_2 است. معادله ۱۰ تابع عضویت در وضعیت استخراج را نشان می‌دهد.

$$\mu_{S_2}(f) = \begin{cases} 0 & 0 \le f \le 0.2 \\ 10\times f - 2 & 0.2 < f \le 0.3 \\ 1 & 0.3 < f \le 0.4 \\ -5\times f + 3 & 0.4 < f \le 0.6 \\ 0 & 0.6 < f \le 1 \end{cases} \qquad (۱۰)$$

- هم‌گرایی: کمترین مقدار f نشان دهنده مجموعه S_3 است. معادله ۱۱ تابع عضویت در وضعیت هم‌گرایی را نشان می‌دهد.



شکل (۳): نمودار تغییرات تابع عضویت به ازای فاکتور تکامل

$$\mu_{S_3}(f)=\begin{cases}1 & 0\le f\le 0.1\\ -5\times f+1.5 & 0.1<f\le 0.3\\ 0 & 0.3<f\le 1\end{cases} \quad (۱۱)$$

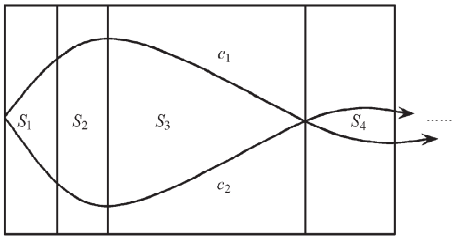
- پرش: زمانی‌که الگوریتم در وضعیت پرش از بهینه محلی است بهترین ذره عمومی در فاصله بسیار دوری از خوشه ذرات دیگر قرار دارد بنابراین f بیشترین مقدار خود را خواهد داشت. معادله ۱۲ تابع عضویت در وضعیت پرش را نشان می‌دهد.

$$\mu_{S_4}(f)=\begin{cases}1 & 0\le f\le 0.7\\ 5\times f-3.5 & 0.7<f\le 0.9\\ 1 & 0.9<f\le 1\end{cases} \quad (۱۲)$$

مقدار بیشینه توابع عضویت، نشان می‌دهد که f به کدام مجموعه تعلق دارد. اما تصمیم گیری در نواحی گذر به همین سادگی نیست. در این نواحی علاوه بر مقدار تابع عضویت، وضعیت قبلی الگوریتم نیز تاثیر گذار است. مراحل الگوریتم به ترتیب $S_1 >> S_2 >> S_3 >> S_4$ ... $>> S_1$ >> تغییر می‌کنند. حال اگر مقدار عامل تکامل برابر با 0.45 باشد. مقدار تابع عضویت در مجموعه‌های S_1 و S_2 را خواهد داشت و نشان می‌دهد که الگوریتم در حال گذر میان S_1 و S_2 است. در این حالت وضعیت فعلی بر اساس وضعیت قبلی مشخص می‌شود. اگر حالت قبلی S_4 باشد، حالت فعلی S_1 خواهد بود. اگر وضعیت قبلی S_1 باشد، آنگاه به منظور پایداری کلاسه‌بندی، حالت فعلی نیز S_1 مشخص می‌شود. اگر وضعیت قبلی S_2 یا S_3 باشد، آنگاه وضعیت فعلی S_2 خواهد بود. بعد از این‌که وضعیت تکاملی جمعیت مشخص شد، نوبت به تغییر پارامترهای وزن اینرسی و ضرایب شتاب می‌رسد. این تغییرات به شرح زیر انجام می‌شود:

تطبیق وزن اینرسی: عامل تکامل f بعضی از مشخصه‌ها را با وزن اینرسی w به اشتراک می‌گذارد به طوری‌که f در وضعیت اکتشاف به نسبت بزرگ است و در وضعیت همگرایی به نسبت کوچک می‌شود. در وضعیت پرش یا اکتشاف، f یا w بزرگ برای جستجوی عمومی مفید خواهد بود. بر عکس زمانی که f کوچک است، یک وضعیت استخراج یا همگرایی مشخص می‌شود و بنابراین w کاهش می‌یابد؛ چون جستجوی محلی مفید است. بنابراین از یک تابع سیگموید[۲۳] برای نگاشت f به w استفاده شده است [6].

$$w(f)=\frac{1}{1+1.5e^{-2.6f}}\in[0.4\ 0.9]\quad \forall f\in[0\ 1] \quad (۱۳)$$



شکل (۴): روند تغییرات ضرایب شتاب در وضعیت‌های مختلف تکامل

- **کنترل ضرایب شتاب**: مقادیر ضرایب شتاب c_1 و c_2 بر اساس این‌که الگوریتم در کدام وضعیت تکامل باشد، تغییر خواهد کرد.
- **وضعیت اکتشاف:** افزایش c_1 و کاهش c_2. افزایش c_1 و کاهش c_2 می‌تواند به جستجوی محلی ذرات کمک کرده تا بهترین موقعیت‌های گذشته خودشان را به دست آورند به جای این‌که اطراف بهترین ذره فعلی را شلوغ کنند.
- **وضعیت استخراج:** افزایش مقطعی c_1 و کاهش مقطعی c_2. در این وضعیت ذرات از اطلاعات محلی استفاده می‌کنند و در اطراف بهینه محلی که به‌وسیله بهترین موقعیت قبلی هر ذره مشخص شده است، جمع می‌شوند؛ بنابراین افزایش تدریجی c_1 و حفظ یک مقدار نسبتا بزرگ می‌تواند جستجو و استخراج اطراف $pbest_i$ را تقویت کند. در میانه زمان، همیشه بهترین ذره عمومی هنوز در بهترین ناحیه عمومی قرار نگرفته است؛ بنابراین، کاهش آهسته c_2 و حفظ یک مقدار کوچک می‌تواند از گیر افتادن در یک بهینه محلی جلوگیری کند. به علاوه یک وضعیت استخراج در اغلب موارد بعد از یک وضعیت اکتشاف و قبل از وضعیت همگرایی رخ می‌دهد. بنابراین تغییر جهت برای c_1 و c_2 باید به صورت مقطعی از وضعیت اکتشاف به وضعیت همگرایی انجام شود.
- **وضعیت همگرایی:** افزایش مقطعی c_1 و افزایش مقطعی c_2. در وضعیت همگرایی به نظر می‌رسد که جمعیت ناحیه بهینه عمومی را پیدا کرده است؛ بنابراین تاثیر c_2 باید برجسته شود تا ذرات دیگر را به ناحیه بهنیه عمومی محتمل هدایت کند. بنابراین مقدار c_2 باید افزایش یابد. از طرف دیگر مقدار c_1 باید کاهش یابد تا به گروه اجازه همگرایی سریع دهد. نتیجه این است که گروه به شدت بهترین ناحیه فعلی را دنبال می‌کند که موجب همگرایی زودرس خواهد شد که اگر بهترین ناحیه فعلی یک بهینه محلی باشد، بسیار خطرناک است. برای جلوگیری از این امر هر دو پارامتر به صورت مقطعی افزایش می‌یابند.

جدول (۱): توابع محک مورد آزمایش

شماره تابع	نام تابع	ویژگی تابع	محدوده جستجو	بهینه مطلق	فرمول تابع
$f1$	Sphere	تک‌قله‌ای	[-30 30]	0	$f1(x)=\sum_{i=1}^{D} x_i^2$
$f2$	Quadric	تک‌قله‌ای	[-100 100]	0	$f2(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2)$
$f3$	Rosenbrock	تک‌قله‌ای	[-2.048 2.048]	0	$f3(x)=\sum_{i=1}^{n/2}(100(x_{2i}-x_{2i-1}^2)^2+(1-x_{2i}-1)^2)$
$f4$	Rastrigin	چندقله‌ای	[-5.12 5.12]	0	$f4(x)=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$
$f5$	Ackley	چندقله‌ای	[-30 30]	0	$f5(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D\sum_{i=1}^{D}x^2}}\right)-\exp(1/D\sum_{i=1}^{D}cos2\pi x_i)+20+e$
$f6$	Griewank	چندقله‌ای	[-600 600]	0	$f6(x)=1/4000\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos(x_i/\sqrt{i})+1$

- **وضعیت پرش:** کاهش c_1 و افزایش c_2. زمانی‌که بهترین ذره عمومی از بهینه محلی به سمت یک بهینه محلی بهتر پرش می‌کند، در حقیقت از خوشه پر ازدحام دور می‌شود. به محض این‌که این ناحیه جدید به‌وسیله یک ذره کشف می‌شود، ذرات گروه باید این ذره را دنبال کنند و با بیشترین سرعت ممکن به سمت این ناحیه جدید حرکت کنند. یک مقدار بزرگ c_2 به همراه یک مقدار به نسبت کوچک c_1 برای رسیدن به این هدف مناسب است.

مرزهای ضرایب شتاب: نرخ تغییر ضرایب شتاب به صورت تصادفی یکنواخت در محدوده [۰/۱ ۰/۰۵] مشخص می‌شود و برای تغییرات مقطعی این مقدار ۰/۵ برابر می‌شود. در این مقاله ضرایب شتاب با ۲ مقداردهی اولیه شده‌اند. همچنین مقدار هر یک از این ضرایب در بازه[۲/۵ ۱/۵] و مجموع آن‌ها در بازه [۴ /۳] محدود شده‌اند. اگر مجموع ضرایب شتاب کمتر از 3 شود سپس هر دو ضریب به روش زیر نرمال می‌شوند.

$$c_i=\frac{c_i}{c_1+c_2}*3 \qquad (۱۴)$$

و اگر این مجموع بزرگتر از 4 باشد، نرمال‌سازی به روش زیر انجام می‌شود.

$$c_i=\frac{c_i}{c_1+c_2}*4 \qquad (۱۵)$$

که در آن‌ها $i=1,2$ می‌باشد.

در هر مرحله از اجرای الگوریتم پارامترهای وزن اینرسی و ضرایب شتاب مطابق روش بیان شده در این بخش به روزرسانی می‌شوند. این مرحله در شکل ۲ با تابع *AdaptiveParametrs* نمایش داده شده است.

شکل ۴ روند تغییرات ضرایب شتاب را در مراحل مختلف تکامل نشان می‌دهد. لازم به ذکر است که ضریب c_1 در وضعیت هم‌گرایی به صورت مقطعی افزایش می‌یابد اما این تغییر در شکل ۴ به صورت کاهشی نشان داده شده است، این امر به دلیل نرمال‌سازی اتفاق می‌افتد. در بخش بعد نتایج شبیه‌سازی این الگوریتم آورده شده است.

۵- ارزیابی کارآیی الگوریتم ACPSO

الگوریتم پیشنهادی ACPSO در محیط MATLAB پیاده‌سازی و بر روی توابع محک استاندارد برای حل مسائل ۱۰ بعدی، آزمایش شد. به منظور بررسی دقیقتر در ابعاد بالاتر، تمام آزمایشات برای حل مسائل ۳۰ بعدی نیز تکرار شد. مشخصات توابع مورد آزمایش در جدول ۱ آورده شده است. همچنین ACPSO با سایر روش‌های مشابه مقایسه گردیده که شرح آن‌ها در ذیل آورده شده است:

- الگوریتم PSO استاندارد: این الگوریتم از شیوه وزن اینرسی و ضرایب شتاب ثابت استفاده می‌کند.
- الگوریتم Dec_IWPSO : این الگوریتم، وزن اینرسی را متغیر با زمان به صورت غیر خطی و مطابق معادله (۱۶) کاهش می‌دهد [۱۵]. در این معادله *iteration* شماره تکرار حلقه اصلی الگوریتم را نشان می‌دهد. پارامتر u مطابق پیاده‌سازی انجام شده در [۱۵] با مقدار ۱/۰۰۰۰۲ مقدار دهی شده است.

$$w_{new} = w * u^{-iteration} \quad (۱۶)$$

- الگوریتم CPSO: این الگوریتم، PSO تعاونی است [۱۷] که برای حل مسأله n-بعدی، n دسته ۱-بعدی را ایجاد می‌نماید و در حقیقت نسخه غیر تطبیقی الگوریتم پیشنهادی است (رجوع شود به بخش ۳- ۲).
- الگوریتم APSO: این الگوریتم، به صورت تطبیقی پارامترهای PSO را کنترل می‌کند [۶]. تطبیق پارامترهای وزن اینرسی و ضرایب شتاب در آن همانند ACPSO انجام می‌شود و در حقیقت نسخه غیر تعاونی الگوریتم پیشنهادی است.

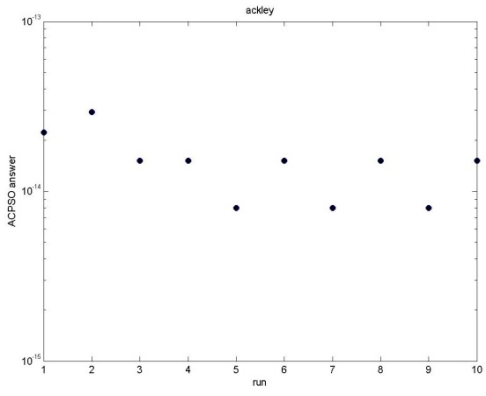
در الگوریتم‌های PSO و CPSO پارامترها مقادیر ثابت دارند، همچنین ضرایب شتاب نیز در الگوریتم Dec_IWPSO نیز ثابت هستند. جدول ۲ مقدار پارامترهای ثابت را در آزمایشات انجام شده نشان می‌دهد (مطابق با مقادیر ارائه شده در[۱۷]).

جدول (۲): مقدار پارامترهای ثابت

پارامتر	مقدار
وزن اینرسی (w)	۰.۷۹
عامل شناخت (c_1)	۱.۴۹
عامل اجتماعی (c_2)	۱.۴۹
شرط توقف	برای ۵۰۰ تکرار متوالی شایستگی *gbest* ثابت باشد.

۵-۱- بررسی کیفی پاسخ الگوریتم ACPSO

آزمایشات ۱۰ بار تکرار شده‌اند تا از کیفیت پاسخ‌های به‌دست آمده، اطمینان حاصل شود. بهترین شایستگی *gbest* بعد از میانگین‌گیری از ۱۰ اجرای مختلف به همراه انحراف معیار این ۱۰ جرا برای حل مسائل ۱۰بعدی و ۳۰ بعدی در جدول ۳ نشان داده شده است. شکل 5 یک نمونه از نمودار پایداری را برای تابع $f4$در حالت ۱۰ بعدی برای ۱۰ اجرای مختلف نشان می‌دهد.



شکل (۵): نمودار پایداری برای تابع Rastrigin

شکل (۵): نمودار پایداری تابع آکلی در حالت ۱۰ بعدی

f1 یک تابع تک‌قله‌ای است که بهینه عمومی آن در نقطه 0 قرار دارد. به طور کلی الگوریتم PSO بر روی توابع تک‌قله‌ای خوب عمل کرده و از نقاط قوت PSO محسوب می‌شوند. پاسخ‌ها در حالت ۱۰ بعدی نشان می‌دهند که پاسخ الگوریتم Dec_IWPSO، تقریبا ۱۶۰ برابر بهتر از پاسخ الگوریتم PSO استاندارد است. تفاوت این دو الگوریتم تنها در نحوه مقداردهی پارامتر وزن اینرسی در معادله سرعت است. این امر اهمیت مقداردهی پارامتر وزن اینرسی را نشان می‌دهد. همچنین پاسخ الگوریتم APSO نسبت به پاسخ Dec_IWPSO، تقریبا ۱۵۵ برابر بهتر شده است. APSO نسبت به Dec_IWPSO علاوه بر وزن اینرسی، پارامترهای ضرایب شتاب را نیز تغییر می‌دهد. به این ترتیب اهمیت ضرایب شتاب نیز مشخص می‌شود. هر دو الگوریتم تعاونی CPSO و ACPSO در حالت ۱۰ بعدی با موفقیت به بهینه مطلق دست یافته‌اند. می‌توان نتیجه‌گیری کرد که الگوریتم‌های تعاونی موجب ارتقای کیفیت پاسخ حتی در حل مسائل ابعاد پایین می‌شوند. پاسخ‌های به دست آمده در حالت ۳۰ بعدی بر روی این تابع نشان می‌دهد که الگوریتم‌های PSO و Dec_IWPSO که به ترتیب دارای وزن‌های اینرسی ثابت و نزولی غیر خطی هستند، در حالت ۳۰ بعدی به جواب‌هایی نزدیک به هم رسیده و Dec_IWPSO کمی بهتر از PSO عمل کرده است. همچنین الگوریتم‌های APSO و CPSO نیز به جواب‌های مشابهی دست یافتند که اختلاف زیادی با دسته اول دارد و بسیار بهتر شده است. الگوریتم دیگر ACPSO است، به بهینه عمومی دست یافته و به این ترتیب برتری مطلق الگوریتم پیشنهادی را در ابعاد بالا بر روی تابع *f1* نشان می‌دهد.

f2 یک تابع تک‌قله‌ای است که بر خلاف انتظار، PSO در هر دو حالت ۱۰ بعدی و ۳۰ بعدی به جواب قابل قبولی در آن نرسیده است. این اتفاق برای الگوریتم‌های Dec_IWPSO و CPSO هم رخداده است. از میان روش‌های مورد بررسی تنها الگوریتم‌های APSO و ACPSO هستند که به روش تطبیقی، پارامترها را مقداردهی کرده و بر روی این تابع به جواب‌های قابل قبولی دست یافته‌اند. بنابراین می‌توان نتیجه‌گیری کرد که مقداردهی مناسب پارامترها در رسیدن به پاسخ بر روی این تابع از اهمیت ویژه‌ای برخوردار است.

f3 یک تابع تک‌قله‌ای است که هر سه الگوریتم APSO, PSO و Dec_IWPSO که از ساختار یکپارچه استفاده می‌کنند، در حالت ۱۰ بعدی به جواب‌های مشابهی رسیده‌اند و در یک سطح از بهینه محلی گیر افتاده‌اند، البته با بیشتر شدن ابعاد مسأله در حالت ۳۰ بعدی، کارآیی این الگوریتم‌ها افت کرده است. هر دو الگوریتم ACPSO و CPSO که ابعاد را جداگانه مورد بررسی قرار می‌دهند توانسته‌اند به جواب‌های بسیار بهتری نسبت به الگوریتم‌های یکپارچه برسند و کارآیی خود را با بالا رفتن ابعاد مسأله از ۱۰ بعد به ۳۰ بعد همچنان حفظ کنند.

f4 یک تابع چندقله‌ای است و از آنجا که PSO بر روی توابع چندقله‌ای عملکرد ضعیفی داشته و معمولا در بهینه محلی گیر می‌کند. انتظار می‌رود PSO به جواب مناسبی بر روی این تابع نرسد. همان‌طور که در جدول ۳ مشاهده می‌شود سه الگوریتم PSO، Dec_IWPSO و APSO که ساختار غیر تعاونی دارند بر روی این

جدول (۳): نتایج ۱۰ اجرای مختلف برای بهینه‌سازی توابع محک استاندارد

بعد	تابع محک		PSO	Dec-IWPSO	APSO	CPSOS	ACPSO
۱۰ بعدی	*f*1	میانگین	2.11e-008	1.48e-168	1.98e-323	0	0
		انحراف معیار	7.44e-019	0	0	0	0
	*f*2	میانگین	2.41e+000	3.00e+003	1.48e-323	1.11e+003	2.57e-322
		انحراف معیار	6.70e+000	4.21e+003	0	2.44e+003	0
	*f*3	میانگین	6.18e-005	1.25e+000	1.76e-005	1.97e-028	2.59e-027
		انحراف معیار	1.32e-004	2.45 e+000	5.58e-005	4.89e-028	2.53e-028
	*f*4	میانگین	1.21e+001	1.61e+001	1.35e+001	0	0
		انحراف معیار	6.28e+000	1.14e+001	15.65e+000	0	0
	*f*5	میانگین	1.60e-001	2.22e+000	8.11e+001	2.26e-014	1.51e-014
		انحراف معیار	5.20e-001	6.24e+000	8.78e-001	1.16e-014	6.69e-014
	*f*6	میانگین	1.40e-001	9.88e-002	1.02e-001	0	0
		انحراف معیار	7.60e-002	4.72e-002	5.02e-002	0	0
۳۰ بعدی	*f*1	میانگین	3.95e-019	7.79e-015	2.57e-322	4.94e-324	0
		انحراف معیار	7.44e-019	1.46e-014	0	0	0
	*f*2	میانگین	5.90e+003	3.02e+004	3.46e-321	4.07e+004	3.28e-321
		انحراف معیار	2.72e+003	1.71e+004	0	4.01e+004	0
	*f*3	میانگین	1.02e-002	8.63e+001	1.17e-001	2.27e-026	8.08e-027
		انحراف معیار	1.81e-002	1.77e+002	2.36e-001	7.06e-026	8.25e-028
	*f*4	میانگین	1.00e+002	1.15e+002	5.77e+001	0	0
		انحراف معیار	2.42e+001	3.13e+001	1.33e+001	0	0
	*f*5	میانگین	4.09e+000	1.02e+001	3.18e+000	7.34e-014	4.95e-014
		انحراف معیار	2.08e+000	6.58e+000	9.38e-001	4.61e-014	2.04e-014
	*f*6	میانگین	2.77e-001	3.61e+001	4.57e-002	0	0
		انحراف معیار	3.91e-001	6.31e+001	5.07e-002	0	0

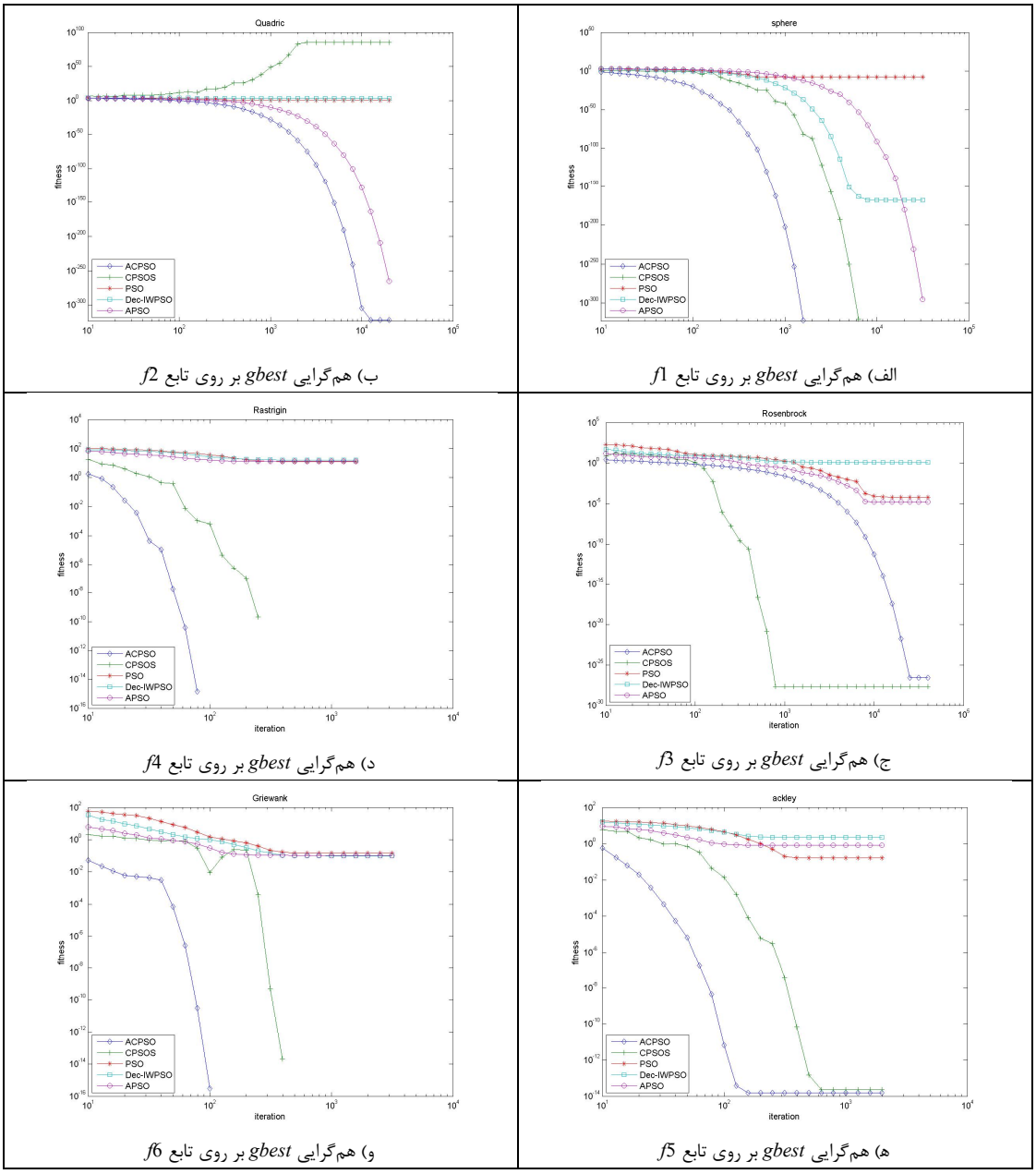
تابع جواب مناسبی نگرفتند و در مقابل هر دو تابع تعاونی CPSO و ACPSO به بهینه مطلق دست یافتند. می‌توان نتیجه گیری کرد که ساختار تعاونی بر روی این تابع چندقله‌ای از موفقیت کامل برخوردار است زیرا در هر دو حالت ۱۰ بعدی و ۳۰ بعدی تنها الگوریتم‌هایی که ساختار تعاونی دارند به بهینه مطلق این تابع چندقله‌ای رسیده‌اند.

*f*5 نیز یک تابع چندقله‌ای است که تنها یک کمینه عمیق در نقطه صفر دارد و در سطوح بالاتر پر از کمینه‌های محلی است. در هر دو حالت ۱۰ بعدی و ۳۰ بعدی سه الگوریتم PSO، Dec_IWPSO و APSO در سطوح یکسانی از بهینه محلی گیر افتاده و الگوریتم‌های CPSO و ACPSO هستند که با موفقیت از این سطوح گذر کرده و به سطح عمیق‌تری دست یافته و به جواب قابل قبول رسیده‌اند.

*f*6 نیز یک تابع چندقله‌ای است. همان‌طور که انتظار می‌رود الگوریتم PSO بر روی این تابع در بهینه محلی گیر کرده است. همچنین در هر دو حالت ۱۰ بعدی و ۳۰ بعدی الگوریتم‌های Dec_IWPSO و APSO هم در موقعیت‌های مشابهی گیر کرده و به جواب مناسب نرسیده‌اند. اما هر دو الگوریتم CPSO و ACPSO با موفقیت کامل به بهینه مطلق دست یافته‌اند.

مقایسه پاسخ‌های به دست آمده در حالت ۱۰ بعدی با پاسخ‌های به دست آمده در حالت ۳۰ بعدی نشان می‌دهد که هر سه روش PSO، Dec_IWPSO و APSO که از ساختار بردارهای یکپارچه استفاده می‌کنند، با بالا رفتن ابعاد مسأله دچار یک افت نسبی در کیفیت پاسخ می‌شوند و در عوض الگوریتم‌های CPSO و ACPSO که از یک ساختار تعاونی بهره می‌برند، کیفیت پاسخ خود را با بالا رفتن ابعاد مسأله حفظ می‌کنند.

همچنین می‌توان نتیجه‌گیری کرد که بر روی توابع تک‌قله‌ای تنها الگوریتم ACPSO است که به بهترین پاسخ دست یافته است. در برخورد با توابع چندقله‌ای الگوریتم‌های PSO ، Dec_IWPSO و APSO که از ساختار بردارهای یک‌پارچه استفاده می‌کنند در بهینه محلی گیر کرده و به جواب مناسبی نمی‌رسند اما الگوریتم‌های CPSO و ACPSO که ابعاد را جداگانه مورد بررسی قرار می‌دهند، موفق شده‌اند تا از بهینه‌های محلی عبور کرده و به پاسخ‌های قابل قبول و حتی بهینه مطلق دست یابند. با توجه به این‌که بر روی توابع چندقله‌ای، الگوریتم ACPSO از یک برتری نسبی برخوردار بوده است اما هر دو الگوریتم‌های CPSO و ACPSO به جواب‌های مشابهی دست یافته‌اند بنابراین لازم است تا پاسخ‌ها از نظر سرعت هم‌گرایی نیز مورد بررسی قرار گیرند. به همین منظور در بخش بعد الگوریتم ACPSO از نظر سرعت رسیدن به جواب با دیگر روش‌ها مقایسه شده است.



الف) همگرایی *gbest* بر روی تابع *f1*

ب) همگرایی *gbest* بر روی تابع *f2*

ج) همگرایی *gbest* بر روی تابع *f3*

د) همگرایی *gbest* بر روی تابع *f4*

ه) همگرایی *gbest* بر روی تابع *f5*

و) همگرایی *gbest* بر روی تابع *f6*

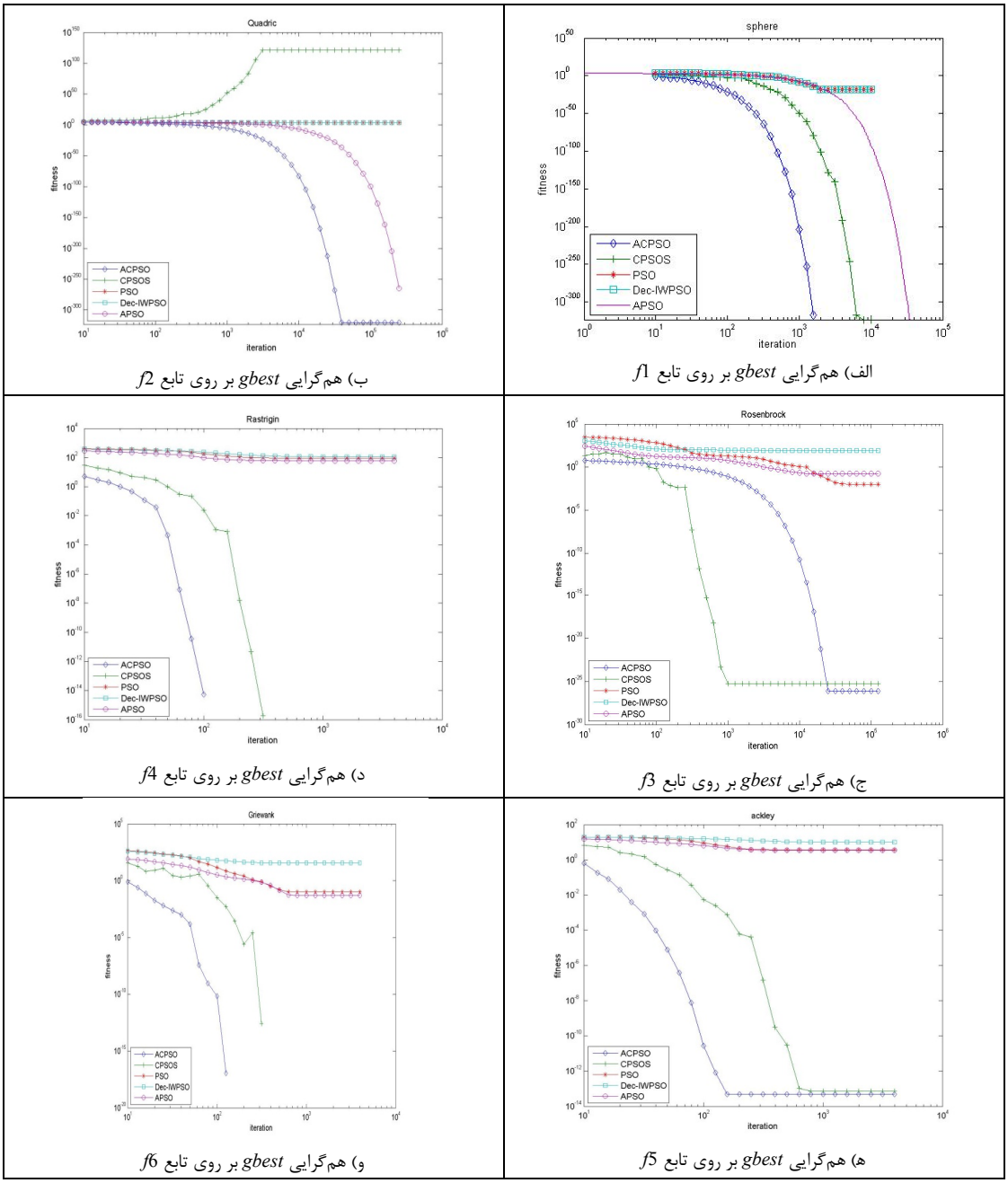
شکل (۶): همگرایی *gbest* برای حالت ۱۰ بعدی

۵-۲- تحلیل سرعت الگوریتم ACPSO

به منظور بررسی‌های دقیقتر لازم است سرعت رسیدن به جواب در الگوریتم‌های مورد آزمایش تحلیل شود. بنابراین روند تغییرات پاسخ الگوریتم‌های مورد بررسی بر روی توابع مختلف برای میانگین ۱۰ اجرای مختلف برای هر دو حالت ۱۰ بعدی و ۳۰ بعدی به ترتیب در شکل‌های ۶ و ۷ نشان داده شده است. مقایسه این دو شکل نشان می‌دهد روند تغییرات پاسخ برای هر تابع در هر دو حالت ۱۰ بعدی و ۳۰ بعدی تقریبا مشابه است.

شکل ۶-الف روند همگرایی در روش‌های مورد بررسی را در حالت ۱۰ بعدی بر روی تابع *f1* نشان می‌دهد. همان‌طور که در شکل پیداست الگوریتم ACPSO سریع‌تر از بقیه روش‌ها همگرا شده است. شکل ۷-الف نشان می‌دهد که روند تغییرات شایستگی بهترین ذره بر روی تابع *f1* در حالت ۳۰ بعدی برای دو الگوریتم PSO و Dec_IWPSO به طور کامل مشابه یکدیگر است. الگوریتم‌های APSO و CPSO به جواب‌های مشابهی در حالت ۳۰ بعدی جدول ۳ رسیدند اما شکل ۷-الف نشان می‌دهد که سرعت رسیدن به همان پاسخ در CPSO بیشتر بوده است. همان‌طور که در این شکل قابل ملاحظه است الگوریتم ACPSO سریع‌تر از سایر روش‌های دیگر همگرا شده است.

روند تغییرات پاسخ بر روی تابع *f2* برای حالت‌های ۱۰بعدی (شکل ۶-ب) و ۳۰ بعدی (شکل ۷-ب) کاملا مشابه یکدیگر است. باز



الف) هم‌گرایی *gbest* بر روی تابع *f1*

ب) هم‌گرایی *gbest* بر روی تابع *f2*

ج) هم‌گرایی *gbest* بر روی تابع *f3*

د) هم‌گرایی *gbest* بر روی تابع *f4*

ه) هم‌گرایی *gbest* بر روی تابع *f5*

و) هم‌گرایی *gbest* بر روی تابع *f6*

شکل (۷): هم‌گرایی *gbest* برای حالت ۳۰ بعدی

هم دو الگوریتم PSO و Dec_IWPSO رفتارهای مشابهی را از خود نشان داده‌اند. بر خلاف انتظار تغییرات شایستگی *gbest* در الگوریتم CPSO به صورت افزایشی بوده است. در جدول ۳ بهترین پاسخ‌ها بر روی تابع *f2* متعلق به الگوریتم‌های تطبیقی APSO و ACPSO است اما شکل‌های ۶-ب و ۷-ب نشان می‌دهند که ACPSO سریع‌تر از APSO به جواب مورد نظر دست یافته است.

روند تغییرات شایستگی بر روی تابع *f3* در هر دو حالت ۱۰ بعدی و ۳۰ بعدی (شکل‌های ۶-ج و ۷-ج) نزدیک به یکدیگر است. توابع CPSO و APSO به کمینه‌هایی بسیار عمیق‌تر از سایر الگوریتم‌ها رسیده‌اند. همان‌طور که در شکل قابل مشاهده است این‌بار الگوریتم CPSO است که سریعتر از ACPSO هم‌گرا شده است.

بر روی تابع *f4* تنها الگوریتم‌های تعاونی CPSO و ACPSO هستند که به جواب مطلوب رسیده‌اند اما در شکل‌های شکل‌های ۶-د و ۷-د کاملا مشهود است که ACPSO در تکرارهای کمتری، بهینه مطلق را فتح کرده است.

روند تغییرات پاسخ تابع *f5* در هر دو حالت ۱۰ بعدی و ۳۰ بعدی از الگوی مشابهی پیروی می‌کنند. همان‌طور که شکل‌های ۶-ه و ۷-ه نشان می‌دهند روند تغییرات پاسخ در الگوریتم‌های PSO، Dec_IWPSO و APSO بر روی تابع *f5* نزدیک به یکدیگر است و

هر سه این الگوریتم در بهینه محلی گیر افتاده‌اند. همچنین تغییرات شایستگی در الگوریتم‌های تعاونی CPSO و ACPSO از الگوی مشابهی پیروی می‌کند و حتی در نهایت به پاسخ‌های مشابهی دست یافته‌اند اما فاصله زمانی قابل توجهی میان این دو نمودار وجود دارد به طوری‌که ACPSO سریع‌تر از CPSO عمل کرده است.

روند تغییرات نمودارها بر روی تابع *f6* در شکل‌های ۶-و و ۷-و از ساختاری مشابه تابع *f5* پیروی می‌کند. این بار نیز ACPSO در مقایسه با دیگر الگوریتم‌ها توانسته است در کمترین تکرار به بهترین پاسخ برسد.

نمودارهای بررسی شده نشان می‌دهند که روند تغییرات شایستگی بر روی توابع چندقله‌ای در الگوریتم‌های CPSO و ACPSO که از ساختار تعاونی بهره می‌برند، مشابه یکدیگر است. همچنین سه الگوریتم PSO، Dec_IWPSO و APSO نیز بر روی این تابع‌ها ساختاری نزدیک به هم را دنبال می‌کنند. تنها الگوریتم‌های ACPSO و CPSO توانستند بر روی توابع چندقله‌ای به پاسخ‌های قابل قبولی برسند اما همانطور که از نمودارها پیداست سرعت هم‌گرایی ACPSO در ۵ مورد از ۶ مورد آزمایش انجام شده بیشتر از CPSO بوده است. به طور کلی می‌توان در مورد سرعت هم‌گرا شدن نتیجه‌گیری کرد که ACPSO موفق شده است تا در تکرارهای کمتری نسبت به سایر روش‌های مورد بررسی به جواب مطلوب دست یابد، پس از سرعت هم‌گرایی بیشتری برخوردار است.

۶- نتیجه گیری و راهکارهای آتی

۶-۱- نتیجه‌گیری

این مقاله یک الگوریتم PSO جدید به نام ACPSO ارائه می‌کند که در یک ساختار تعاونی، پارامترهای وزن اینرسی (w)، عامل شناخت (c_2) و عامل اجتماعی (c_1) را به صورت تطبیقی برای هر ذره در هر تکرار برای هر بعد به روزرسانی می‌کند. از آنجا که یک ساختار تعاونی از چندین ساختار کوچکتر تشکیل شده است؛ بنابراین سرعت اجرای یک الگوریتم تعاونی چند برابر سرعت اجرای یک الگوریتم استاندارد است. تطبیقی کردن پارامترها در الگوریتم تعاونی PSO باعث شده است تا هر کدام از پارامترها با مناسب‌ترین مقدار خود معادله سرعت را به روزرسانی کنند، این کار در نهایت منجر به افزایش سرعت کلی الگوریتم می‌شود زیرا الگوریتم در تکرارهای کمتری به پاسخ می‌رسد.

ساختار تعاونی الگوریتم ACPSO سه مزیت زیر را به همراه دارد؛ ۱) برای حل مسائل با ابعاد بالا موفق است؛ زیرا ساختارهای پیچیده را به مؤلفه‌های اولیه خود شکسته و تبدیل به مسأله‌های ساده‌تر می‌کند. ۲) زمانی که مؤلفه‌های گسسته در کنار یکدیگر قرار می‌گیرند تا شایستگی را حساب کنند، ترکیب‌های بسیار زیادی را به وجود می‌آورند که موجب افزایش تنوع در کل جمعیت اصلی می‌شود. افزایش تنوع رویکردی است که جمعیت را از هم‌گرایی زودرس در بهینه محلی نجات می‌دهد. ۳) محاسبه شایستگی در هنگام تغییر هر مؤلفه از بردار پاسخ از بروز حالت‌هایی که در آن بعضی از مؤلفه‌ها بدتر و برخی دیگر بهتر می‌شوند، جلوگیری می‌کند.

آزمایش‌های انجام شده برای بهینه‌سازی توابع محک استاندارد در دو حالت ۱۰ بعدی و ۳۰ بعدی نشان داد که ACPSO به جواب بهتری نسبت به روش‌های دیگری که پارامترهای ثابت (PSO)، متغیر با زمان (Dec_IWPSO) یا متغیر تطبیقی (APSO) دارند، رسیده است. کیفیت و سرعت رسیدن به پاسخ در ACPSO از CPSO که ساختار تعاونی غیرتطبیقی دارد بهتر است. به طوری‌که ACPSO توانست در سه مورد از پنج مورد آزمایش، بهینه مطلق را در مسائل ۱۰ بعدی و ۳۰ بعدی کشف کند. نمودارهای تغییرات شایستگی *gbest* نشان می‌دهند که ACPSO با فاصله زمانی قابل توجهی می‌تواند سریع‌تر از سایر روش‌ها هم‌گرا شود.

۶-۲- کارهای آتی

در این مقاله برای تطبیقی کردن پارامترهای معادله سرعت از روش ارائه شده در[۶] استفاده شده است اما می‌توان این کار را با روش‌های دیگر نیز انجام داد. در ادامه این کار قصد داریم از سایر روش‌های دیگر برای مقداردهی خودکار این پارامترها استفاده کنیم و نتایج را با نتایجی که در این مقاله به دست آورده‌ایم مقایسه کنیم.

مراجع

[1] C. F. Juang and Y. C. Chang, "Evolutionary Group-Based Particle Swarm-Optimized Fuzzy Controller with Application to Mobile Robot Navigation in Unknown Environments", IEEE Trans On Fuzzy Systems, Issue 99, doi. 10.1109/tfuzz.2011.2104364, 2011.

[2] W. C. Wu and M. C. Tsai, "Application of Enhanced Integer Coded Particle Swarm Optimization for Distribution System Feeder Reconfiguration", IEEE Tran on Power Systems, doi. 10.1109/TPWRS.2010.2094212, 2011.

[3] K. Y. Chan, T. S. Dillon and C. K. Kwong, "Modeling of a Liquid Epoxy Modeling Process Using a Particle Swarm Optimization-Based Fuzzy Regression Approach", IEEE Tran on Industrial Informatics, Vol. 7, No. 1, Feb 2011.

[4] Y. H. Wang, K. W. Chan and T. Y. G. Pong, "Dynamic Voltage Security Constrained Optimal Coordinated Voltage Control Using Enhanced Particle Swarm Optimization", IET Generation, Transmission & Distribution, Vol. 5, Issue. 2, pp. 239–248, doi. 10.1049/iet-gtd.2010.0157, 2011.

[5] S.-B. Peng, J. Xu, S. B. Peng and J. B. Xiang, "Parametric Inverse Synthetic Aperture Radar Manoeuvring Target Motion Compensation Based on Particle Swarm Optimizer", IET Radar Sonar Navig, Vol. 5, Issue. 3, pp. 305–314, doi. 10.1049/iet-rsn.2010.0175, 2011.

[6] Z. H. Zhan, J. Zhang, Y. Li and S. H. Chung, "Adaptive Particle Swarm Optimization", IEEE Trans on Systems

Man and Cybernetics Part B: Cybernetics, Vol. 39, No. 6, pp. 1362-1381, Dec 2009.
[7] Z. Afsahi and M. Meybodi, "Improving Cooperative PSO Using Fuzzy Logic", Research and Development in Intelligent Systems XXVI, pp. 219-232, Oct 2009.
[8] B. K. Panigrahi, V. R. Pandi and S. Das, "Adaptive Particle Swarm Optimization Approach for Static and Dynamic Economic Load Dispatch", Energy Conversion and Management, Vol. 49, pp. 1407–1415, 2008.
[9] Z. Ma, V. Volski and G. A. E. Vandenbosch, "Optimal Design of a Highly Compact Low-cost and Strongly Coupled 4 Element Array for WLAN", IEEE Trans on Antennas and Propagation, Issue 99, doi. 10.1109/TAP.2010.2103029, 2011.
[10] X. Yang, J. Yuan, J. Yuan and H. Mao, "A Modified Particle Swarm Optimizer with Dynamic Adaptation", Applied Mathematics and Computation, Vol. 189, pp. 1205-1213, 2007.
[11] M. Maitra and A. Chatterjee, A Hybrid Cooperative Comprehensive Learning Based PSO Algorithm for Image Segmentation Using Multilevel Thresholding, Expert Systems with Applications, 2007.
[12] C. N. Ko, Y. P. Chang and C. J. Wu, An Orthogonal-Array-Based Particle Swarm Optimizer with Nonlinear Time-Varying Evolution, Applied Mathematics and Computation, 2007.
[13] F. Van den Bergh, An Analysis of Particle Swarm Optimizers, Ph.D Thesis, University of Pretoria, Pretoria, 2006.
[14] A. Y. Saber, T. Senjyu, N. Urasaki and T. Funabashi T, Okinawa Unit Commitment Computation A Novel Fuzzy Adaptive Particle Swarm Optimization Approach, Power Systems Conference and Exposition, pp. 1820-1828, 2006.
[15] B. Jiao, Z. Lian and X. Gu, "A Dynamic Inertia Weight Particle Swarm Optimization Algorithm", Chaos Solitons & Fractals, Vol. 37, Issue 3, pp. 698-705, 2008.
[16] R. C. Eberhart; and Y. Shi, "Guest Editorial Special Issue on Particle Swarm Optimization", IEEE Trans on Evolutionary Computation, pp. 201-203, 2004.
[17] F. Van den Bergh and P. Engelbrecht, "A Cooperative Approach to Particle Swarm Optimization", IEEE Trans on Evolutionary Computation, Vol. 8, No. 3, pp. 225-239, Jun 2004.
[18] M. P. Song and G. C. Gu, Research on Particle Swarm Optimization: A Review, Proc. of the Third International Conference on Machine Learning and Cybernetics, Shanghai, pp. 26-29, Aug 2004.
[19] A. Ratnaweera, S. K. Halgamuge and H. C. Watson, "Self-Organizing Hierarchical Particle Swarm Optimizer with Time-Varying Acceleration Coefficients", IEEE Trans on Evolutionary Computation, pp. 240-255, 2004.
[20] R. C. Eberhart and Y. H. Shi, Particle swarm optimization: Developments applications and resources, Proc. IEEE Congress on Evolutionary Computation, Korea, pp. 81-86, 2001.
[21] R. C. Eberhart and Y. H. Shi, Tracking and Optimizing Dynamic Systems with Particle Swarms, Proc. Congress on Evolutionary Computation, Korea, 2001.
[22] Y. H. Shi and R. C. Eberhart, Fuzzy Adaptive Particle Swarm Optimization, Proc. Congress on Evolutionary Computation, Seoul, Korea, 2001.
[23] R. C. Eberhart and Y. Shi, Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization, IEEE Cong on Evolutionary Computation, pp. 84–88, 2000.
[24] Y. H. Shi and R. C. Eberhart, Empirical Study of Particle Swarm Optimization, Cong. on Evolutionary Computation, Washington DC, USA, Jul 1999.
[25] Y. Shi Y and R. C. Eberhart, A Modified Particle Swarm Optimizer, Proc. of the IEEE Congress on Evolutionary Computation, pp. 69-73, May 1998.
[26] M. A. Potter, The Design and Analysis of a Computational Model of Cooperative Coevolution, Ph.D Thesis, George Mason University, Fairfax, Virginia, USA, 1997.
[27] E. Cantu-Paz, "A Summary of Research on Parallel Genetic Algorithms", IlliGAL Report No. 95007, Illinosis Genetic Algorithms Laboratory (IlliGAL), Urbana-Chamaign, IL, USA, Jul 1995.
[28] R. C. Eberhart and J. Kennedy, A New Optimizer Using Particle Swarm Theory, Proc. of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp. 39-43, 1995.
[29] J. Kennedy and R. C. Eberhart, Particle Swarm Optimization, Proc. of IEEE international Conference on Neural Networks, Perth, Australia, Vol. 4, pp. 1942-1948, 1995.
[30] M. A. Potter and K. A. de Jong, A Cooperative Coevolutionary Approach to Function Optimization, in the Third Parallel Problem Solving from Nature, Jerusalem, Israel, pp. 249-257, 1994.
[31] S. H. Clearwater, T. Hogg and B. A. Huberman, Cooperative Problem Solving, In Computation: The Micro and Macro View, Singapore: World Scientific, pp. 33-70, 1992.
[32] H. G. Cobb, Is the Genetic Algorithm a Cooperative Learner?, In Fouvdations of Genetic Algorithms 2, pp. 277-297, 1992.
[33] S. Kirkpatrick and C. D. Gelatt; and M. P. Vecchi, "Optimization by Simulated Annealing", Science, Vol. 220, No. 4598, pp. 671-680, 1983.

زیرنویس‌ها

1. Particle Swarm Optimization
2. Adaptive Cooperative Particle Swarm Optimizer
3. Adaptive Particle Swarm Optimization
4. Cooperative Particle Swarm Optimization
5. Cooperative Coevolutionary Genetic Algorithm
6. Curse of Dimensionality
7. Cognition
8. Social
9. Context Vector
10. Exploration
11. Exploitation
12. Simulated Annealing