بهینه سازی گروه ذرات مشارکتی فازی

محمد حسین نورزوی بیرامی
دانشکده مهندسی برق، رایانه و فناوری اطلاعات
دانشگاه آزاد اسلامی
قزوین ایران
mh.noroozi@gmail.com

محمدرضا میبدی
دانشکده مهندسی کامپیوتر و فناوری اطلاعات
دانشگاه صنعتی امیرکبیر
تهران ایران
mmeybodi@aut.ac.ir

چکیده: بهینه سازی گروه ذرات یک تکنیک بهینه سازی مبتنی بر جمعیت می باشد که بر اساس قوانین احتمال کار می کند. در این روش هر یک از ذرات سعی می کنند به سمتی حرکت کنند که بهترین تجربه های فردی و گروهی در آن نقاط روی داده است. دو مشکل عمده الگوریتم PSO استاندارد، افتادن در دام بهینگی محلی و پایین بودن سرعت همگرایی آن می باشد. یکی روش برای حل مشکلات عنوان شده ترکیب PSO استاندارد با منطق فازی می باشد. در این مقاله در ابتدا یک الگوریتم ترکیبی مبتنی بر منطق فازی که مشکلات فوق را حل میکند پیشنهاد میگردد و سپس آن در بهینه سازی گروه ذرات مشارکتی بکار گرفته میشود. به منظور ارزیابی، الگوریتمهای پیشنهادی بر روی تعدادی از توابع محک استاندارد آزمایش شده اند و نتایج حاصل با نتایج بدست آمده با الگوریتمهای موجود مقایسه گردیده است. نتایج مقایسه نشان از برتری الگوریتمهای پیشنهادی دارد.

کلمات کلیدی: بهینه سازی گروه ذرات، بهینه سازی گروه ذرات مشارکتی، بهینگی محلی، منطق فازی

Cooperative Fuzzy Particle Swarm Optimization

M. H. Noroozi Beyrami
Electrical and Computer Engineering and Information Technology Department
Islamic Azad University
Qazvin, Iran
mh.noroozi@gmail.com

M. R. Meybodi
Computer Engineering and Information Technology Department
Amirkabir University of Technology
Tehran Iran
mmeybodi@aut.ac.ir

Abstract: Particle swarm optimization is a population based optimization technique that is based on probability rules. In this technique each particle moves toward their best individual and group experience had occurred. Fundamental problems of standard PSO algorithm are the falling into the trap of local optimum and its low speed of convergence. One approach for solving the above problems is to combine fuzzy logics with PSO. In this paper we first use fuzzy logic to improve standard PSO and then use it in cooperative particle swarm optimization. For evaluation purpose, the proposed algorithms are tested on number of standard optimization functions and the results obtained are compared with the results for the existing algorithms. The results of comparisons have shown the superiority of the proposed algorithm over existing algorithms.

Keywords: Particle Swarm Optimization, Cooperative PSO, Local Optimum, Fuzzy Logic

۱. مقدمه

در بسیاری از مسائل بهینه سازی بویژه مسائل بزرگ، انتخاب بهترین جواب از طریق جستجوی همه جانبه اگر چه غیر ممکن نیست، ولی کاری بسیار مشکل است. هدف مسئال بهینه سازی کاهش زمان این جستجوی می باشد. روش های ابتکاری[1] راه حل های خوبی برای یافتن جواب بهینه می باشد، ولی تضمینی برای یافتن جواب بهینه نمی دهند. اما امروزه با بزرگ و پیچیده شدن مسائل، استقبال از روش

[1] - Heuristic

های ابتکاری بطور چشم گیری افزایش یافته است[2].

بهینه سازی گروه ذرات[1] یکی از تکنیک های بهینه سازی ابتکاری می باشد که بر مبنای جمعیت کار می کند. ایده اصلی این روش ابتدا در سال ۱۹۹۵ توسط دکتر کندی و دکتر ابرهارت[3] مطرح گردید که از رفتار دسته جمعی ماهی ها و پرندگان برای یافتن غذا الهام می گیرد. گروهی از پرندگان و ماهی ها در یک فضای تصادفی دنبال غذا می گردند و تنها یک تکه غذا وجود دارد و هیچ یک از پرندگان از محل غذا اطلاعی ندارد و فقط فاصله خود تا غذا را می داند ، یکی از بهترین استراتژی ها دنبال کردن پرنده ای می باشد که به غذا نزدیک تر است، این تئوری، استراتژی اصلی الگوریتم PSO می باشد.

در PSO هر پرنده یک جواب ممکن در فضای مسئله می باشد که ذره[2] نامیده می شود. هر ذره دارای یک مقدار شایستگی می باشد که توسط تابع شایستگی مسئله محاسبه می شود. ذره ای که به جواب نزدیک تر باشد، شایستگی بیشتری دارد. این الگوریتم ماهیت پیوسته ای دارد و در کاربرد های متعددی کارایی خود را اثبات کرده است[1]. در این مقاله یک الگوریتم ترکیبی فازی با PSO به نام های FPSO1 ارائه شده است که اولا مشکلات الگوریتم استاندارد را تا حد زیادی حل می کند و از طرفی باعث افزایش سرعت همگرایی آن می شود. برای اینکه بتوان امکان جستجوی موازی به الگوریتم PSO استاندارد اضافه شود، الگوریتم های بهینه سازی گروه ذرات مشارکتی ارائه شده است[18,24]. این الگوریتم ها با تقسیم فضای مسئله در بین دسته های ذرات مختلف، اولا امکان جستجوی موازی را به PSO استاندارد اضافه می کنند و از طرف دیگر امکان همکاری بین دسته های ذرات مختلف را برای همگرایی بهتر به وجود می آورد. در این مقاله ما یک الگوریتم بهینه سازی گروه ذرات مشارکتی ارائه می کنیم که از یک طرف امکان جستجوی موازی را به وجود می آورد و از طرف دیگر برای مقدار دهی اولیه ذرات از بهینه های محلی فضای مسئله استفاده می کند. این ویژگی های الگوریتم پیشنهادی باعث می شود که سریع تر بهینه های محلی را تشخص داده و از افتادن ذرات در آنها جلوگیری نماید. برای مقایسه الگوریتم پیشنهادی با توابع موجود نیز از توابع محک استاندارد استفاده خواهیم کرد.

بخش های بعدی این مقاله به این صورت سازماندهی شده است. در بخش دوم الگوریتم PSO استاندارد مورد بحث قرار گرفته است. در بخش سوم پیش زمینه ای از منطق فازی و کاربرد های آن ارائه می شود. در بخش چهارم الگوریتم های ترکیبی PSO با منطق فازی را توضیح خواهیم داد. در بخش پنجم الگوریتم های PSO مشارکتی توضیح داده شده و در بخش ششم الگوریتم پیشنهادی مورد بحث قرار خواهد گرفت. در بخش هفتم ارزیابی های انجام گرفته برای الگوریتم پیشنهادی با روش های موجود را بیان می کنیم و در نهایت در بخش هشتم نتیجه گیری را بیان خواهیم کرد.

۲. الگوریتم PSO استاندارد

PSO از جمله الگوریتم های جستجوی موازی مبتنی بر جمعیت است که با یک گروه از جواب های تصادفی (ذره ها) شروع به کار می کند، سپس برای یافتن جواب بهینه در فضای مسئله با به روز کردن مکان ذره ها به جستجو ادامه می دهد. هر ذره به صورت چند بعدی (بسته به نوع مسئله) با دو بردار X_{id} و V_{id} که به ترتیب معرف موقعیت مکانی و سرعت بعد d ام از i امین ذره هستند، مشخص می شود. در هر مرحله از حرکت جمعیت، مکان هر ذره با دو مقدار بهترین به روز می شود. اولین مقدار، بهترین تجربه ای است که خود ذره تا به حال بدست آورده است و با p_best نشان داده می شود. دومین مقدار، بهترین تجربه ای است که در بین تمامی ذره ها بدست آمده است و با g_best نشان داده می شود. در برخی ویرایش های PSO، ذره قسمتهایی از جمعیت را که همسایگان توپولوژیکی اش هستند، انتخاب می کند و تنها آنها را در اعمال خود دخیل می کند. در این صورت بهترین راه حلی محلی استفاده می شود و از l_best به جای g_best استفاده می شود و مدل های مختلفی برای بدست آوردن همسایگان توپولوژیکی هر ذره ارائه شده است[4,5]. در هر تکرار، الگوریتم بعد از یافتن دو مقدار بالا، سرعت و موقعیت جدید ذره را بر اساس معادلات (۱) و (۲) بروز رسانی می کند.

---

[2] - Particle Swarm Optimization - PSO

2 - Particle

$$v_{id}(t+1) = w.v_{id}(t) + c_1.rand_1(p\_best_{id}(t) - x_{id}(t)) + c_2.rand_2(g\_best_{id}(t) - x_{id}(t)) \quad (۱)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (۲)$$

در معادله (۱) ، w ضریب اینرسی است که به صورت خطی کاهش می یابد که معمولا در بازه [۱-۰] می باشد. c_1 و c_2 ضرایب یادگیری یا شتاب می باشد که در بازه [۲-۰] انتخاب می شود که در بیشتر موارد برای هر دوی آنها از مقدار ۱,۴۹و یا ۲ استفاده شده است[3,6,7]. دو عدد$rand_1$ و $rand_2$ نیز اعداد تصادفی در بازه [۱-۰] می باشد. همچنین مقدار نهایی سرعت هر ذره برای جلوگیری از واگرایی الگوریتم به یک بازه محدود می شود. $v_{id} \in [-v_{\max}, v_{\max}]$. شرط خاتمه الگوریتم همگرایی تا حد معین و یا توقف بعد از تعداد معینی تکرار می باشد. معادله (۲) نیز بردار موقعیت فعلی ذره را با توجه به سرعت جدید آن بروز رسانی می کند.

سمت راست معادله (۱) از سه قسمت تشکیل شده است، قسمت اول ضریبی از سرعت فعلی ذره می باشد، قسمت دوم برای تغییر سرعت و چرخش ذره به سمت بهترین تجربه شخصی می باشد و قمست سوم نیز باعث تغییر سرعت و چرخش ذره به طرف بهترین تجربه گروهی می باشد[2]. در واقع حرکت بهینه سازی گروه ذرات بدون قسمت اول معادله (۱)، فرایندی خواهد بود که طی آن فضای جستجو به تدریج کوچک می شود و جستجوی محلی حول بهترین ذره شکل می گیرد، اما در مقابل قسمت اول معادله (۱) باعث حرکت ذرات در مسیر عادی خود خواهد بود تا به دیواره محدوده جستجو برسد و به نوعی جستجوی سراسری انجام می دهد. با ترکیب این دو عامل سعی شده است تا موازنه ای بین جستجوی محلی و سراسری برقرار گردد. w برای برقراری بهتر این موازنه برای اولین بار در [13] پیشنهاد گردید که ضریب حرکت در جستجوی سراسری را مشخص می کند. دو پارامتر c_1 و c_2 نیز ضریب حرکت در جستجوی محلی را مشخص می کند. با توجه به اینکه w رفته رفته کاهش می یابد ولی c_1 و c_2 همواره ثابت است، بنابراین با ادامه جستجو از میزان جستجوی سراسری کم شده و جستجوی محلی افزایش می یابد تا بهینه ترین جواب بدست آید. هر چه شیب کاهشی w کمتر باشد امکان پیدا کردن جواب بهینه سراسری افزایش پیدا می کند. یکی از مشکلات شناخته شده الگوریتم PSO این است که برای ذره ای که شامل اطلاعات g_best است، قسمت های دوم و سوم معادله (۱)، صفر خواهد شد. بنابراین ذره در راستای بردار حرکت قبلی خود حرکت خواهد کرد. از طرفی چون به طور معمول w کوچکتر از یک می باشد، این قسمت نیز میرا خواهد بود. بنابراین ذره شامل g_best ثابت خواهد شد و سایر ذرات نیز به این ذره همگرا می شوند[2]. در بسیار از این موارد این موضوع باعث همگرایی زودرس الگوریتم به یک بهینه محلی خواهد شد. برای رفع این مشکل ابتدا در سال ۲۰۰۲، الگوریتم جدیدی با نام GCPSO [1] ارائه گردید[8] که در این روش یک پارامتر جدیدی به الگوریتم اضافه شده بود که باعث جستجوهای تصادفی ذره g_best حول جواب بهینه خواهد شد. دومین مشکلی که الگوریتم PSO استاندارد دارد، همگرایی خطی آن است که ممکن است بسیار زمان بر باشد. ما در این مقاله دو روش ترکیبی برای حل این مشکلات ارائه می کنیم.

۳. پیش زمینه ای در مورد سیستم های فازی

منطق فازی تکنولوژی نسبتا جدیدی است که در مقابل روشهای مرسوم برای طراحی و مدلسازی سیستمی که نیازمند ریاضیات و احتمالات پیشرفته و نسبتا پیچیده می باشد، به کار می رود. از مقادیر و قوانین مبتنی بر متغیرهای زبانی و یا به عبارتی از دانش فرد خبره با هدف ساده، دقیق و کارآمد کردن طراحی سیستم استفاده می‌کنند[9,10] . مرحله پردازش که به آن موتور استنباط نیز گفته می شود بر اساس مجموعه ای از قوانین فازی بصورت عبارات IF-THEN عمل می کند. هر سیستم فازی تعدادی قانون دارند که در پایگاه دانش آن ذخیره می گردد. به عنوان مثال: اگر تکرار پایین باشد اولویت جابجایی خیلی بالا است که عبارات اولویت و تکرار متغیرهای زبانی می‌باشند و پایین و خیلی بالا عبارات زبانی می باشند. هر عبارت زبانی مطابق یک تابع عضویت می باشد. دو رویه عمده برای استنباط وجود دارد . نوع اول مدل استنباط فازی ممدانی می باشد که توسط آقای ابراهیم ممدانی در ۱۹۷۵ ارائه شد و نوع دوم استنباط فازی، روش تاکاگی سوگنو می باشد که در ۱۹۸۵ ارائه شده است. این دو روش در بسیاری از جنبه ها یکسان می باشند، مانند فازی سازی ورودیها و

1- Guaranteed Convergence PSO - GCPSO

عملگرهای فازی، اما تفاوت اصلی بین روش سوگنو و ممدانی این است که خروجی روش سوگنو عضو توابعی است که می تواند خطی و یا ثابت باشد، ولی در استنباط ممدانی انتظار داریم که خروجی، توابع عضویت مجموعه های فازی باشند[11]. ما در طراحی سیستم مورد نظر، بر حسب نتایج مورد انتظار از روش آقای ممدانی استفاده کرده ایم.

۴. روش ترکیبی فازی با PSO

با توجه به مطالب ارائه شده در بخش قبل و آشنایی با مشکلات PSO می توان گفت که از جمله مشکلات مهم روش استاندارد یکی افتادن در دام بهینگی محلی و دیگری پایین بودن سرعت همگرایی آن است. برای حل مشکل اول باید راهکاری اندیشید تا بهینگی محلی تشخیص داده شده و از آن اجتناب شود[12]. روش های مختلفی برای این کار پیشنهاد شده است. در [13] برای اجتناب از افتادن در دام بهینگی محلی در صورتی که سرعت ذره از یک حد مشخصی کمتر شود ولی میزان شایستگی جواب بدست آمده قابل قبول نباشد از یک تابعی برای ایجاد یک شک در سرعت ذره استفاده می شود. در [14,15] از یک تابع غیر خطی فازی برای تغییر ضریب اینرسی استفاده می شود، در صورتی که میزان شایستگی بدست آمده قابل قبول نباشد و ضریب اینرسی نیز کاهش یافته باشد، موجب افزایش این ضریب شده و با افزایش سرعت ذره امکان جستجوی سراسری را بیشتر می کند. در [16,17] با شناسایی دو قله محلی نزدیک به هم تا حد امکان از افتادن در دام بهینگی محلی حفظ می کند. همه این روش های ارائه شده به نوعی مانع از افتادن در دام بهینگی محلی خواهد شد. در این مقاله ما یک روش جدید ارائه می کنیم که در واقع یک تابع غیر خطی برای ضرایب شتاب c1 و c2 می باشد. در ادامه دو روش ترکیبی ارائه شده را به صورت مختصر توضیح می دهیم.

۴. ۱. روش ترکیبی FPSO1 [1]

الگوریتم پیشنهادی را FPSO1 می نامیم که ضرایب شتاب c1 و c2 برای این الگوریتم یک تابع فازی در نظر گرفته شده است که سه پارامتر d1 ، d2 و NCBPE به عنوان ورودی این تابع و c1 و c2 خروجی آن می باشد. با توجه با اینکه در اکثر منابع، دو ضریب c1 و c2 به صورت مساوی در نظر گرفته شده است، ما نیز در این الگوریتم این دو مقدار را مساوی هم در نظر می گیریم و تابع فازی پیشنهادی فقط یک خروجی دارد.

$$d1=|p\_best-x| \qquad d2=|g\_best-x| \tag{۳}$$

$$NCBPE = \frac{CBPE - CBPE_{\min}}{CBPE_{\max} - CBPE_{\min}} \tag{۴}$$

در روابط (۳) دو پارامتر d1 و d2 معرف میزان نزدیکی ذره به بهترین تجربه محلی و سراسری را نشان می دهد که می تواند میزان نزدیکی به بهینه محلی و سراسری را بازگو نماید و NCBPE نیز بهترین ارزیابی کارایی فعلی نرمال شده[2] را مشخص می نماید[7,8] که در رابطه (۴) مشخص شده است. $CBPE_{\min}$ بهترین جوابی است که تا کنون بدست آمده و در مقابل $CBPE_{\max}$ بدترین جوابی است که تا کنون بدست آمده است. اما نکته مهمی که در این الگوریتم وجود دارد، انتخاب قوانین فازی می باشد که تاثیر مستقیم بر روی نتایج بدست آمده دارد. جدول (۱) تعدادی از قوانین استفاده شده در این سیستم را نشان می دهد.

---

[1] - Fuzzy PSO

[2] Normal Current Best Performance Evaluation - NCBPE

جدول (۱) تعدادی از قوانین فازی الگوریتم FPSO1

Rules	Input			Output
	d1	d2	NCBPE	c1 & c2
1	High	High	High	Low
2	Low	Low	Low	High
3	Low	Low	High	Low
4	Low	Low	Medium	Medium

با توجه به توضیحاتی که در بخش های قبلی ارائه شد، چون c1 و c2 ضرایب شتاب برای افزایش جستجوی محلی حول بهترین تجربه گروهی و فردی می باشد. بنابراین کاهش c1 و c2 باعث افزایش جستجوی سراسری و افزایش آن باعث افزایش جستجوی محلی می باشد. در نتیجه در این تابع، نمودار c1 و c2 نموداری صعودی خواهد بود تا رفته رفته از میزان جستجوی سراسری کم شده و جستجوی محلی افزایش پیدا کند. فوانین ارائه شده در جدول (۱) نیز بر این اصل استوار می باشد. قوانین ارائه شده به طور واضح مبنای تصمیم گیری را مشخص می کند ولی باید به این نکته توجه کرد که قانون سوم، قانونی است که بهینگی محلی را تشخیص می دهد به این صورت که ذره هم به بهترین جواب محلی و هم به بهترین جواب سراسری نزدیک شده است و از طرف دیگر جواب بدست آمده کارایی لازم را ندارد، در واقع یک بهینه محلی برای مسئله می باشد، بنابراین با کم کردن پارامتر خروجی باعث می شود که ذره از حالت کنونی خارج شده و بتواند به جستجوی سراسری ادامه دهد.

۵. الگوریتم های بهینه سازی گروه ذرات مشارکتی[1]

در این روش ها که تعمیمی بر روش های PSO استاندارد می باشد، به این صورت عمل می شود که به جایی اینکه از یک دسته ذره برای حل مسئله استفاده شود، چندین دسته ذره برای پیدا کردن جواب بهینه به صورت مشارکتی با هم کار می کنند، روش های مختلفی برای این کار پیشنهاد شده است که در ادامه دو نوع پرکاربرد این روش ها را ارائه می نماییم.

۵. ۱. بهینه سازی گروه ذرات مشارکتی ارباب رعیتی

در این روش جمعیت از یک دسته اصلی[2] و چند دسته پیرو[3] تشکیل می شود و رابطه بین دسته های اصلی و پیرو می تواند امکان رسیدن به بهینه ترین جواب را برای دسته اصلی به وجود می آورد. به این روش مدل ارباب رعیتی[4] گفته می شود [18]. هر یک از دسته های پیرو الگوریتم PSO را به صورت مجزا اجرا می کند و موقعیت و سرعت اعضای خود را بروز رسانی می کند. بعد از اینکه تک تک دسته های پیرو موقعیت جدید خود را مشخص کردند، بهترین نتایج بدست آماده محلی را برای دسته اصلی ارسال می کنند. دسته اصلی می تواند به دو صورت متفاوت با این نتایج رفتار نماید. روش اول روش، رقابتی[5] می باشد که دسته اصلی برای بروز رسانی سرعت و موقعیت خود از روابط (۵) و (۶) استفاده می کند.

$$v_i^M(t+1) = w v_i^M(t) + r_1 c_1 (p_i^M - x_i^M(t)) + \varphi r_2 c_2 (p_g^M - x_i^M(t)) + (1-\varphi) r_3 c_3 (p_g^S - x_i^M(t)) \qquad (۵)$$

$$x_i^M(t+1) = x_i^M(t) + v_i^M(t) \qquad (۶)$$

[1] - Cooperative Particle Swarm Optimization
[2] - Master Swarm
[3] - Slave Swarm
[4] - master-slave model
[5] - Competitive Multi-Swarm Cooperative PSO – COM-MCPSO

در روابط (۵) و (۶) M معرف دسته اصلی و S معرف دسته های پیرو می باشد و $p_g{}^M$ بهترین ذره فعلی در دسته اصلی و $p_g{}^S$ بهترین ذره فعلی در دسته های پیرو را نشان می دهد. $g^M{}_{best}$ نیز بهترین مقدار شایستگی بدست آمده از ذرات دسته اصلی و $g^S{}_{best}$ بهترین مقدار شایستگی بدست آمده از ذرات دسته های پیرو می باشند. برای اینکه بتوان تاثیر مقادیر شایستگی را کنترل کرد، می توان مقدار φ را از رابطه (۷) محاسبه کرد.

$$\varphi = \begin{cases} 0 & g^S{}_{best} < g^M{}_{best} \\ 0.5 & g^S{}_{best} = g^M{}_{best} \\ 1 & g^S{}_{best} > g^M{}_{best} \end{cases} \qquad (۷)$$

دسته دوم MCPSO همکاری کننده[1] نامیده می شود که در این روش دسته اصلی با همکاری دسته های پیرو موقعیت و سرعت ذرات خود را بروز رسانی می کند[15]. در زمان اجرا، هر ذره از دسته اصلی مسیر خود را بر اساس تجربیات خود، تجربیات گروهی در دسته اصلی و بهترین تجربیات ذرات در دسته های پیرو محاسبه می کند. این الگوریتم سرعت و موقعیت جدید خود را بر اساس روابط (۸) و (۹) محاسبه می کند. تمامی متغیر های روابط (۸) و (۹) دقیقا مشابه روابط (۵) و (۶) می باشد.

$$v_{id}{}^M = w v_{id}{}^M(t) + r_1 c_1 (p_{id}{}^M - x_{id}{}^M) + r_2 c_2 (p_{gd}{}^M - x_{id}{}^M) + r_3 c_3 (p_{gd}{}^S - x_{id}{}^M) \qquad (۸)$$

$$x_{id}{}^M = x_{id}{}^M + v_{id}{}^M \qquad (۹)$$

از مقایسه دو روش ارائه شده می توان نتیجه گرفت که در روش رقابتی، یک رقابتی برای استفاده از بهترین تجربه دسته اصلی و پیرو وجود دارد و در صورتی که موفقیت در دو نوع دسته یکسان باشد به صورت یکسانی از تجربیات هر دو استفاده می شود.(به کمک رابطه (۷)) اما در روش همکاری کننده در هر صورت از تجربیات هر دو گروه به اندازه مشخصی، بر اساس ضرایب شتاب استفاده می شود[19,20].

۵. ۲. بهینه سازی گروه ذرات مشارکتی چند دسته ای

روش دیگری نیز برای CPSO ارائه گردیده[24]، که $CPSO\text{-}S_k$ نامیده شده است. به طوری که می توان یک مفهوم خیلی ساده برای الگوریتم PSO اضافه کرد. در PSO استاندارد از یک بردار n بعدی استفاده می کنیم که می توان آن را به n بردار یک بعدی تقسیم بندی نماییم و هر بردار را به یک دسته واگذار نماییم. هر دسته باید یک بعد از جواب نهایی را پیدا نمایند. در واقع هر دسته مسئول بهینه سازی یک مسئله یک بعدی می باشد. یکی از پیچیدگی های این مسئله این است که برای کمینه کردن تابع شایستگی نیاز به یک بردار n بعدی داریم[21]. اگر هر یک از دسته ها یک بعد از جواب را برای محاسبه میزان شایستگی بردار جواب ارائه نمایند، واضح است که در این صورت استقلال دسته ها از بین خواهد رفت. بنابراین نیاز به یک بردار مفهومی[2] داریم که مفهوم مناسبی را برای تک تک دسته ها برای محاسبه میزان شایستگی ارائه نماید[23]. ساده ترین روش این است که برای تشکیل این بردار n بعدی، بهترین تجربه هر یک از دسته ها را بکار برد. بنابراین هر دسته n-1 مقدار از سایر دسته ها دریافت کرده و برای n-1 دسته هم بهترین تجربه خود را ارسال نماید. الگوریتم CPSO-S برای اولین بار در [22] ارائه گردید. اما باید به این نقطه توجه کرد که در بعضی موارد ممکن است بعضی جزء های بردار مفهومی به همدیگر بستگی داشته باشد، در این صورت تقسیم بردار n بعدی به n بردار یک بعدی مناسب به نظر نمی رسد. در این صورت می توان

[1] - Collaborative Multi-Swarm Cooperative PSO – COL-MCPSO
[2] - Context Vector

اجزای وابسته را در یک گروه قرار داد. یعنی اجزای غیر همبسته به بردار های یک بعد و اجزای همبسته به بردار های c بعدی (c<n) تقسیم بندی شود، در صورتی که گروه های مرتبط به راحتی قابل تشخیص باشد این روش قابل اعمال خواهد بود.

۶. الگوریتم پیشنهادی

حال می خواهیم تعمیم جدیدی برای الگوریتم PSO و یا به بیان بهتر برای الگوریتم های FPSO ارائه نماییم که اولا مشکلات افتادن در دام بهینگی محلی را به مراتب کاهش می دهد و ثانیا پایین بودن سرعت همگرایی را بهبود ببخشد و همچنین مقدار دهی تصادفی اولیه را از بین می برد و از طرف دیگر به هدف اصلی روش های CPSO که امکان موازی سازی است نائل می شود. شکل (۱) شبه کدی برای الگوریتم CFPSO [1] می باشد.

```
CFPSO Algorithm
define
gbest: array [1..n] k-dimension vector // for all swarms
Split search space to n independent area
initialize n k-dimensions PSOs: P_j   j ∈[1..n]        // n is numbers of swarms and each swarm to one area
for all swarms
   for i=1:t                  // t is 1/10 of all iteration in simulation
      apply FPSO              // each swarm are limited in own area
      update gbest array for each swarm
   end
end
initialize k-dimension PSO with gbest values  // swarm have n particle
select the best area with minimum fitness for next search
for i=t+1:end of simulation
   apply FPSO for search area or selected area
end
```

شکل(۱) : شبه کد الگوریتم CFPSO

الگوریتم ارائه شده شامل دو قسمت اصلی می باشد. قسمت اول شامل n دسته ذرات مستقل می باشد. در مرحله اول فضای جستجو را به n قسمت مستقل تقسیم می کنیم و هر دسته ذرات مسئول جستجو در یک قسمت می باشد. بعد از این تقسیم بندی هر دسته، الگوریتم FPSO را بر روی ناحیه مورد جستجو اعمال می کند و بعد از اینکه ۱۰٪ از کل جستجو ها را انجام دادند، کار دسته های اولیه تمام می شود و بهترین جوابی که برای هر دسته بدست آمده است به عنوان جواب های اولیه دسته اصلی محسوب می شود. بعد از اتمام این مرحله، دسته اصلی را با n ذره مقدار دهی می کنیم که مقدار اولیه آنها، بهترین جواب های بدست آمده از تک تک دسته های مرحله اول می باشد. به بیان دیگر بهترین تجربه گروهی تک تک دسته ها، بهینه های محلی هر ناحیه مستقل می باشد و دسته اصلی با استفاده از بهینه

[1] - Cooperative Fuzzy Particle Swarm Optimization - CFPSO

های محلی فضای جستجو مقدار دهی می شود. بعد از این کار ۹۰٪ تکرار های باقیمانده را با الگوریتم FPSO بر روی کل فضای جستجو و یا فقط بر روی ناحیه ای که بهترین جواب متعلق به آن است، اعمال می شود. نحوه تقسیم فضای جستجوی می تواند متفاوت باشد، ما در این مقاله فضای جستجو را بر حسب فاصله از مبدا مختصات تقسیم بندی نموده ایم.

۶. ۱. مزایا روش پیشنهادی

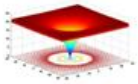
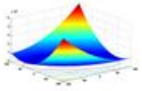
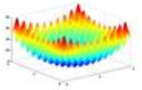
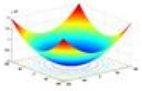
در واقع الگوریتم CFPSO تعمیمی بر الگوریتم FPSO ارائه شده در بخش های قبلی می باشد، اولین هدف این الگوریتم امکان جستجوی موازی می باشد، با تقسیم بندی فضای مسئله هر یک از دسته ها به طور مستقل می تواند عمل جستجو را انجام دهد، از طرف دیگر در دنیای واقعی داده ها به صورت توزیع شده می باشد و هر دسته می تواند در دسته های مختلف عمل جستجو را انجام دهد که این توزیع شدگی نیز ممکن است از نظر جغرافیایی نیز باشد. در این صورت می توان با استفاده از شایستگی جواب هایی که هر یک از دسته ها بدست آورده اند، فضای مسئله را حرص کرد و باعث کوچک شدن فضای مسئله خواهد شود. دومین هدفی که ارائه این الگوریتم داشت ممانعت از افتادن ذرات در دام بهینگی محلی بود. در صورتی که مقدار دهی اولیه ذرات همان بهینه های محلی فضای مسئله باشد، در این صورت مقدار دهی اولیه، تصادفی نخواهد بود و ادامه کار با این مقادیر باعث کاهش افتادن در دام بهینگی محلی می شود.

۷. ارزیابی الگوریتم های پیشنهادی

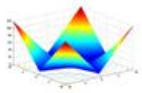
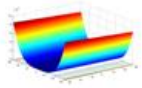
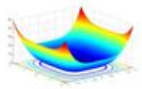
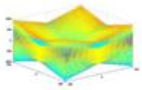
برای مقایسه الگوریتم پیشنهادی با الگوریتم های موجود مشارکتی که در بخش پنجم توضیح داده شد، شبیه سازی های زیادی با نرم افزار MATLAB 7.1 انجام گرفته و با استفاده از توابع محک[1] استاندارد مورد ارزیابی قرار گرفته است این توابع، توابع پرکاربردی می باشد که در ارزیابی های مشابه مورد استفاده قرار گرفته است و در جدول (۲) نشان داده شده است[2,7,9,10,11,15]. همچنین جدول (۲) نمودار های هر یک از توابع محک را برای دو بعد نشان می دهد.

در این ارزیابی، ۲۰ ذره با ۱۰، ۲۰ و ۳۰ بعد و $c_1=c_2=2$ استفاده شده است. w برای توابع بخش ۵ در بازه [0.3 0.9] در نظر گرفته شده است و بعد از ۴۰۰۰ بار تکرار نتایج مورد نظر بدست آمده است. این شبیه سازی ها را ۱۰ بار به طور مستقل انجام داده و نتایج بدست آمده میانگین ده مرحله اجرای مستقل الگوریتم ها می باشد[18]. جداول (۳) نتایج این شبیه سازی ها را برای توابع محک مختلف و ابعاد مختلف نشان می دهد. این حدول بهترین جواب و میانگین جواب های بدست آمده برای الگوریتم های پیشنهادی و الگوریتم های موجود را نشان می دهد. شکل (۲) نمودار شایستگی توابع محک مختلف را برای ۳۰ بعد نشان می دهد.

جدول (۲) توابع محک استاندارد

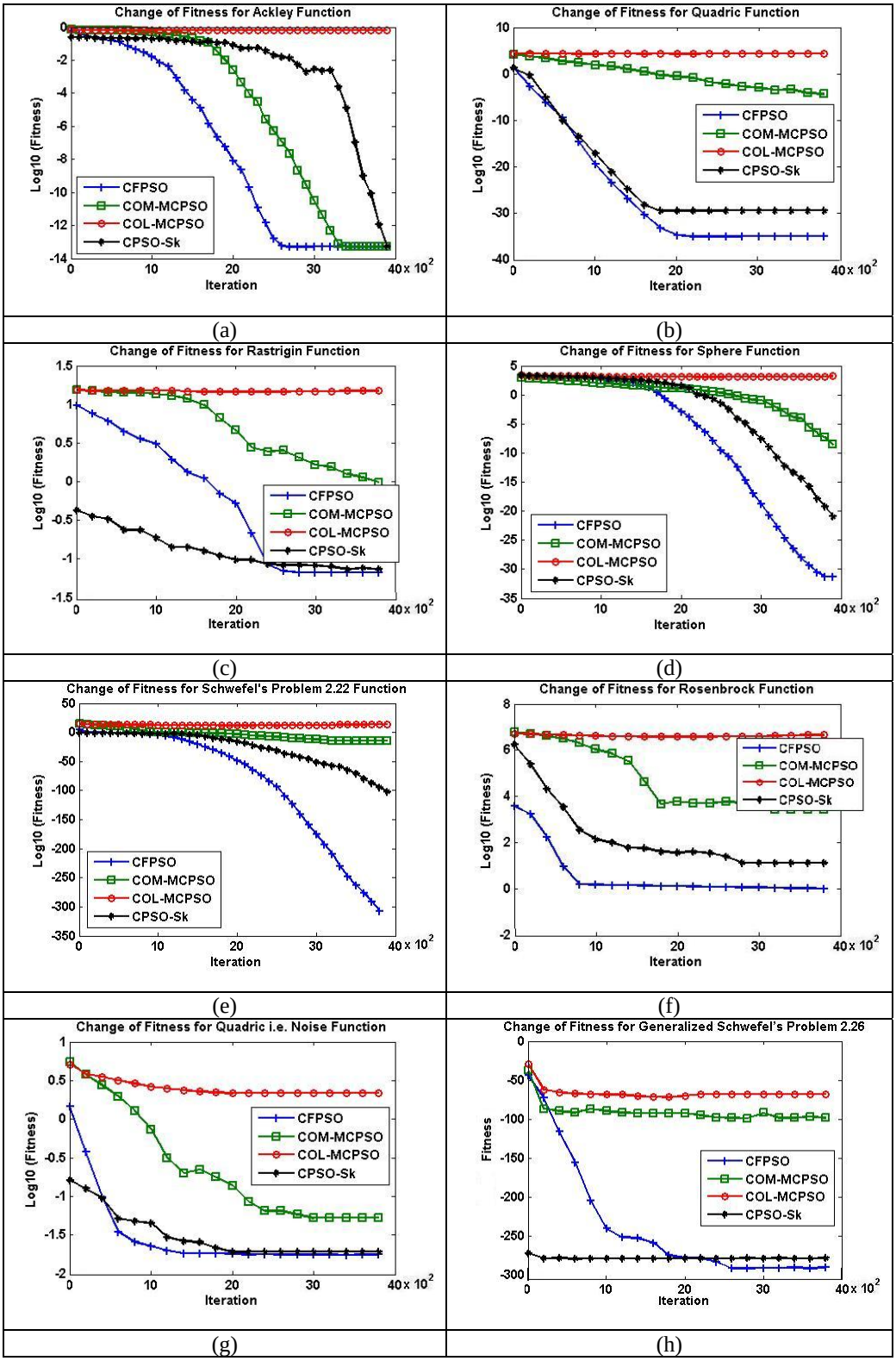
نام تابع	فرمول	دامنه	نمودار (n=2)
Ackley	$f(x)=20+e-20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}}-e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$	$[-32,32]^n$	
Quadric	$f(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_i)^2$	$[-100,100]^n$	
Rasterigin	$f(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	$[-5.12,5.12]^n$	
Sphere	$f(x)=\sum_{i=1}^{n}x_i^2$	$[-100,100]^n$	

1 - Benchmark Function

Schwefel's Problem 2.22	$f(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	$[-10,10]^n$	
Rosenbrock	$f(x)=\sum_{i=1}^{n-1}\left[100(x_{i+1}-x_i^{\,2})^2+(x_i-1)^2\right]$	$[-30,30]^n$	
Quadric i.e. Noise	$f(x)=\sum_{i=1}^{n}ix_i^{\,4}+random[0,1)$	$[-1.28,1.28]^n$	
Generalized Schwefel's Problem 2.26	$f(x)\sum_{i=1}^{n}x_i\sin(\lvert x_i\rvert)$	$[-500,500]^n$	

جدول (۳) نتایج بدست آمده از مقایسه روش های مشارکتی موجود با الگوریتم پیشنهادی با ابعاد مختلف

۳۰ بعد		۲۰ بعد		۱۰ بعد		الگوریتم	توابع محک
مقدار متوسط	بهترین مقدار	مقدار متوسط	بهترین مقدار	مقدار متوسط	بهترین مقدار		
0.0519	1.14 e-14	0.0690	8.88 e-16	0.1036	3.54e-16	CFPSO	Ackley
0.2217	2.64 e-14	0.3128	6.21 e-15	0.4702	2.71 e-15	COM-MCPSO	
18.4521	17.49	16.1984	15.56	10.9454	8.09	COL-MCPSO	
0.1028	4.52 e-13	0.1774	2.66 e-15	0.4463	7.54 e-16	CPSO-S_k	
9.55	7.74 e-36	7.53	1.21 e-71	5.48	9.98 e-162	CFPSO	Quadric
2187.81	3.23 e-5	1224.45	7.27 e-8	431.21	3.42 e-46	COM-MCPSO	
24673.21	20200	10758.34	803.91	7697.54	96.89	COL-MCPSO	
9.49	5.35 e-29	7.00	1.40 e-58	5.11	5.80 e-142	CPSO-S_k	
2.0770	0.0891	1.7901	0.0587	1.5688	0.0315	CFPSO	Rasterigin
7.5615	0.8903	6.6436	0.2333	5.3896	0.2171	COM-MCPSO	
15.0863	11.7829	14.7710	10.8521	14.0205	9.8521	COL-MCPSO	
0.2608	0.0963	0.2235	0.0785	0.2051	0.0514	CPSO-S_k	
241.47	6.10 e-34	167.43	4.99 e-46	104.80	7.31 e-96	CFPSO	Sphere
136.31	9.71 e-12	118.23	8.52 e-37	98.98	2.35 e-45	COM-MCPSO	
1834.42	834.73	1658.54	242.05	1346.71	4.42	COL-MCPSO	
143.48	3.01 e-23	83.49	2.83 e-40	28.43	8.21 e-83	CPSO-S_k	
4.69 e+8	0	1.59 e+5	0	1.5 e+3	0	CFPSO	Schwefel's Problem 2.22
1.36 e+12	1.02 e-13	1.33 e+8	3.18 e-19	3.11 e+4	1.48 e-75	COM-MCPSO	
6.65 e+15	48.15	2.34 e+9	16.53	7.28 e+4	6.04	COL-MCPSO	
5.27 e+13	0	1.80 e+7	0	3.53 e+3	0	CPSO-S_k	
5.63 e+4	19.64	9.03 e+3	5.50	1.34 e+3	2.03 e-6	CFPSO	Rosenbrock
1.02 e+7	80.32	3.00 e+6	30.25	7.10 e+5	0.3697	COM-MCPSO	
2.81 e+10	1.36 e+7	4.58 e+7	2.40 e+6	3.04 e+6	4.21 e+2	COL-MCPSO	
1.32 e+5	24.28	3.05 e+4	9.00	9.84 e+3	6.11 e-3	CPSO-S_k	
0.1262	1.40 e-4	0.1093	1.18 e-4	0.0831	1.04 e-4	CFPSO	Quadric i.e. Noise
0.9002	1.05 e-2	0.4991	5.30 e-3	0.2360	9.86 e-4	COM-MCPSO	
10.6254	7.45	3.5810	1.72	0.6995	0.1319	COL-MCPSO	
0.1464	6.92 e-3	0.9814	2.74 e-3	0.0826	4.70 e-4	CPSO-S_k	
-239.06	-14888	-314.77	-9936	-395.11	-4970	CFPSO	Generalized Schwefel's Problem 2.26
-91.46	-7274	-174.14	-7063	-210.04	-4168	COM-MCPSO	
-66.74	-6002	-96.10	-5617	-111.74	-3382	COL-MCPSO	
-280.16	-12499	-494.55	-9299	-534.37	-4863	CPSO-S_k	



شکل (۲) نمودار میزان شاستگی الگوریتم های مختلف با استفاده از توابع محک استاندارد. (a) تابع Ackley (b) تابع Quadric (c) تابع Rastrigin (d) تابع Sphere (e) تابع Schwefel's Problem 2.22 (f) تابع Rosenbrock (g) تابع Quadric i.e. Noise (h) تابع Generalized Schwefel's Problem 2.26

نتایج ارائه شده در جدول (۳) و شکل (۲) برتری الگوریتم پیشنهادی CFPSO را در برابر روش های موجود نشان می دهد. از مقایسه این نتایج می توان نتیجه گرفت که رفتار تابع پشنهادی خیلی بهتر از دو الگوریتم COM-MCPSO و COL-MCPSO می باشد. این برتری هم در بهترین جواب و هم در متوسط جواب های بدست آمده دیده می شود. اما الگوریتم CPSO-Sk نتایج بهتری نسبت به دو الگوریتم قبلی ارائه کرده است و در بعضی از توابع محک مثل Rastrigin و Quadric with Noise بسیار نزدیک به الگوریتم پیشنهادی می باشد. با توجه به اینکه الگوریتم CPSO-Sk یک مسئله n بعدی را به n مسئله یک بعدی تقسیم می کند، بنابراین همگرایی سریعتر از سایر روش ها می باشد و متوسط جواب های بدست آمده نیز بهتر از الگوریتم پیشهادی می باشد. ولی با توجه به اینکه الگوریتم پیشنهادی فضای مسئله را به n زیر فضا تقسیم می کند و هر دسته، جستجو های ابتدایی را در این زیرفضا ها انجام می دهند، بنابراین بهینه های محلی مسئله به راحتی پیدا می شود. دسته اصلی از این بهینه های محلی برای مقدار دهی اولی ذرات استفاده می کند. در نتیجه جواب های الگوریتم CFPSO بهتر از روش های موجود، خصوصا الگوریتم CPSO-Sk می باشد.

۸. نتیجه گیری

با توجه به نتایج بدست آمده، می توان گفت که استفاده از توابع فازی رفتار الگوریتم PSO استاندارد را به مراتب بهبود می بخشد و این امکان را برای الگوریتم PSO فراهم می کند که در مقابل موقعیت های غیر عادی، عکس العمل مناسبی نشان دهد. همچنین الگوریتم های PSO مشارکتی نیز به دلیل استفاده از جستجوی موازی، رفتار الگوریتم های استاندارد را بهبود می بخشد و ما در این مقاله با ارائه یک روش جدید اولا امکان جستجوی موازی را به الگوریتم استاندارد اضافه کردیم، ثانیا سرعت همگرایی به مراتب افزایش دادیم و از طرف دیگر، به دلیل مقدار دهی اولیه با بهینه های محلی، مشکل اصلی الگوریتم های PSO که افتادن در دام بهینگی محلی می باشد به مراتب مرتفع کردیم.

مراجع

[۱] حسین نظام آبادی پور و مجید رستمی شهر بابکی، "تعمیمی بر الگوریتم GCBPSO"، دوازدهمین کنفرانس مهندسی کامپیوتر ایران، اسفند ۱۳۸۵، صفحات ۲۹-۳۵.

[۲] محمد شیبانی و محمد رضا میبدی، "PSO-LA : یک مدل جدید برای بهینه سازی"، دوازدهمین کنفرانس مهندسی کامپیوتر ایران، اسفند ۱۳۸۵، صفحات ۱۱۶۲- ۱۱۶۹.

[3] J.Kennedy and R. Eberhart, “Particle swarm optimization”, Proceedings of IEEE International Conference on Neural Networks, vol. 4, 1995.

[4] X. Hu and Y. Shi and R. Eberhart, “Recent advances in particle swarm optimization”, IEEE, 2004.

[5] J. Kennedy and R. Mendes, “Population structure and particle swarm performance,” in Proc. IEEE Congr. Evol. Comput., Honolulu, HI, 2002, pp. 1671–1676.

[6] X. Cui, “Document clustering using particle swarm optimization”, IEEE, 2005, pp. 185-191.

[7] L. Hongbo and M. Abraham, “Fuzzy adaptive turbulent particle swarm optimization”, IEEE, 2005, pp. 39-47.

[8] F. Bergh and A. Engelbrecht, “A new locally convergent particle swarm optimizer”, IEEE, 2002.

[9] J. Liebowitz, The Handbook Of Applied Expert Systems. CRC Press LLC, ISBN: 0849331064, 1999.

[10] C. Krishnamoorthy and S. Rajeev, Artificial Intelligence and Expert Systems for Engineers, CRC Press LLC, ISBN: 0849391253, 1996.

[11] T. Kiink and J. S. Vesterstroem and J. Riget, “Particle swarm optimization with spatial particle extension”, Proceedings of the lEEE Congress on Evolutionary Computation, 2002, pp. 1474-1479.

[12] L. Hongbo and M. Abraham, “Fuzzy adaptive turbulent particle swarm optimization”, IEEE, 2005, pp. 39-47.

[13] Y. Shi and R. Eberhart, “A modified particle swarm optimizer ”, IEEE International Conference on Evolutionary Computation, USA, 1998, pp. 255- 262.
[14] R. Eberhart and Y. Shi, “Tracking and optimizing dynamic systems with particle swarms”, Proceedings of IEEE Congress on Evolutionary Computation, Seoul, Korea, 2001, pp. 94–97.
[15] B. Niu and Y. Zhu and X. Xian and H. Shen, "A multi-swarm optimizer based fuzzy modeling approach for dynamic systems processing ", Elsevier, 2007.
[16] M. Clerc and J. Kennedy, “The particle swarm: explosion, stability, and convergence in a multidimensional complex space”, IEEE Trans. Evolut. Comput. 6. 2002, pp. 58–73.
[17] R. Eberhart and S. Yuhui, “Fuzzy adaptive particle swarm optimization”, IEEE, 2001, pp. 101-106.
[18] B. Niu and Y. Zhu, X. He, H. Wu, "MCPSO: A multi-swarm cooperative particle swarm optimizer", Applied Mathematics and Computation 185, Elsevier, 2007, pp. 1050–1062.
[19] Y. Xiong and J. Su, "Parallel cooperative particle swarm optimization based multistage transmission network planning", WSEAS International Conference on Instrumentation, Measurement, Circuits & Systems, Hang Zhou, China, 2007, pp. 15-17.
[20] M. Song and G. Gu and X. Wang and R. Zhang, "Function approximation using the cooperative pso neural network", Harbin Engineering University, 2005.
[21] F. V. D. Bergh and A. Engelbrecht, "A cooperative approach to particle swarm optimization", IEEE Transactions On Evolutionary Computation, Vol. 8, No. 3, 2004.
[22] F. V. D. Bergh and A. Engelbrecht, “Cooperative learning in neural networks using particle swarm optimizers”, South African Comput., vol. 26, 2000, pp. 84–90.
[23] M. Friedman and L. S. Savage, “Planning experiments seeking minima”, in Selected Techniques of Statistical Analysis for Scientific and Industrial Research, and Production and Management Engineering, C. Eisenhart, M. W. Hastay, and W. A. Wallis, Eds. New York: McGraw-Hill, 1947, pp. 363–372.
[24] F. V. D. Bergh and A. P. Engelbrecht. "Effects of swarm size on cooperative particle swarm optimizers”. In Proceedings of the Genetic and Evolutionary Computation Conference. San Francisco. USA, 2001.
[25] M. Clerc and J. Kennedy, “The particle swarm: explosion, stability, and convergence in a multidimensional complex space”, IEEE Trans. Evolut. Comput. 6. 2002, pp. 58–73.