علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۶، شماره ۱ (الف)، ۱۳۸۷
صفحات ۲۱-۳۲
مقاله عادی

بهینه‌سازی اجتماع ذرات باینری: چالش‌ها و راه‌حل‌های جدید

حسین نظام‌آبادی‌پور مجید رستمی شهربابکی ملیحه مغفوری فرسنگی

بخش مهندسی برق، دانشگاه شهید باهنر کرمان، کرمان، ایران

چکیده

بهینه‌سازهای ابتکاری بواسطه توانایی بالایشان در حل مسائل مختلف بهینه‌سازی از توجه خاصی در بین محققان علوم مختلف برخوردار شده‌اند. یکی از جدیدترین این روش‌ها، بهینه‌ساز اجتماع ذرات است. از نقاط قوت این الگوریتم، نسخه باینری آن است که این الگوریتم را در حل مسائل حقیقی و گسسته قادر و توانا ساخته است. در این مقاله الگوریتم جستجوی اجتماع ذرات باینری بطور موشکافانه‌ای تجزیه و تحلیل شده، نقاط ضعف آن بررسی گردیده و راه حل‌های جدیدی برای این الگوریتم ارائه خواهد شد. راه حل‌های پیشنهادی روی توابع محک استاندارد آزموده شده و نتایج آزمایش‌های انجام شده ارائه می‌شود. نتایج آزمایش‌ها ضمن تایید راه حل‌های ارائه شده، افق تازه‌ای در بهینه‌سازی توابع با استفاده از روش‌های پیشنهادی ترسیم می‌کند.

کلمات کلیدی: بهینه‌سازی، الگوریتم‌های ابتکاری، الگوریتم اجتماع ذرات باینری.

۱- مقدمه

بهینه‌سازی نقش مهمی در حل بسیاری از مسائل دنیای امروزی دارد. دامنه این موضوع علمی از مرز رشته‌های مهندسی فراتر رفته و کاربردهای آن در بسیاری از علوم مشاهده می‌شود. در این راستا، در سال‌های اخیر بهینه‌سازهای ابتکاری به خاطر توانائی‌شان در حل مسائل از توجهی خاصی برخوردار شده‌اند. الگوریتم‌های ابتکاری از دسته‌ی الگوریتم‌های اتفاقی هستند که جستجو را با مجموعه‌ای از جواب‌های اتفاقی به نام جمعیت آغاز کرده و با تکیه بر رقابت و همکاری اعضای جمعیت، جواب بهینه را می‌یابند. این روش‌ها، اغلب جواب بهینه را سریعتر از بهینه‌سازهای سنتی می‌یابند. از مشهورترین الگوریتم‌های ابتکاری مبتنی بر جمعیت می‌توان به الگوریتم‌های وراثتی، استراتژی تکاملی، برنامه‌ریزی تکاملی، برنامه‌ریزی وراثتی، جمعیت مورچگان، سیستم دفاع بدن، جستجوی پراکندگی و جستجوی گرانشی اشاره کرد.

روش بهینه‌سازی اجتماع ذرات[1]، عضوی از روش‌های هوش جمعی است که در سال ۱۹۹۵ توسط ابرهارت و کندی با شبیه‌سازی رفتار اجتماعی پرندگان به عنوان یک بهینه‌ساز معرفی شد [۱]. از محسنات این الگوریتم می‌توان به پیاده‌سازی آسان، عدم نیاز به حافظه زیاد، پیچیدگی محاسباتی پایین و عدم نیاز به اطلاعات گرادیان تابع هدف اشاره کرد. کارآیی این الگوریتم در حل بسیاری از مسائل بهینه‌سازی به اثبات رسیده است.

مسائل بسیاری وجود دارد که ماهیت باینری دارند. علاوه بر آن، در بسیاری از کاربردها حل مسائل با فضای پیوسته نیز در فضای رمز شده بصورت باینری انجام می‌شود. بنابراین، از آنجایی که هم مسائل باینری و هم مسائل پیوسته در یک فضای گسسته قابل حل هستند، نیاز نسخه باینری الگوریتم بهینه‌سازی اجتماع ذرات[2] احساس می‌شود. دکتر ابرهارت و دکتر کندی که مبدعان بهینه‌سازی اجتماع ذرات هستند، نسخه باینری این الگوریتم را بصورت انشعابی از الگوریتم اصلی ارائه کردند [۲]. متاسفانه، نسخه باینری به دلایل ضعف‌های ساختاری عمده، نه تنها از همگرایی مطلوبی برخوردار نیست بلکه در پیدا کردن جواب بهینه ناتوان نشان داده است.

در این مقاله، ضمن بررسی مشکلات و ضعف‌های الگوریتم اجتماع ذرات باینری، نسخه باینری جدیدی از آن ارائه می‌شود که مشکلات الگوریتم باینری متداول را برطرف می‌کند. الگوریتم پیشنهادی ضمن برخورداری از همگرایی

مناسب، توانایی بسیار بالایی در یافتن جواب بهینه دارد. سپس مشکلات ناشی از الگوریتم پیوسته که به الگوریتم باینری پیشنهادی سرایت می‌کند، بررسی شده و دو رویکرد متفاوت برای برطرف کردن آن در محیط‌های پیوسته و گسسته ارائه می‌شود. ادامه این مقاله این گونه سازمان‌دهی شده است که در بخش دوم الگوریتم PSO پیوسته و باینری ارائه می‌شود. مشکلات الگوریتم PSO باینری متداول در بخش سوم بررسی و تحلیل می‌شود. الگوریتم باینری پیشنهادی به همراه بررسی کارایی آن در بخش چهارم ارائه می‌شود. پیشنهادهایی برای بهبود الگوریتم پیشنهادی در بخش پنجم معرفی شده، سپس نتایج آزمایش‌ها و تحلیلی و بررسی آنها در این بخش می‌آید. در نهایت، بخش ششم به جمع بندی می‌پردازد.

۲- بهینه‌سازی اجتماع ذرات

این الگوریتم جستجو از زندگی حیواناتی که بصورت گروهی زندگی کرده و بسیاری احتیاجات خود از جمله جستجو و پیدا کردن غذا را با کمک یکدیگر به انجام می‌رسانند، الهام گرفته شده است [۳]. از جمله این حیوانات می‌توان به بعضی از پرندگان و ماهی‌ها اشاره کرد که از چنین فرایندی استفاده می‌کنند.

در این الگوریتم فرض شده است که پرندگان در جستجوی غذا، بصورت غریضی فاصله خود را تا غذا حس می‌کنند در حالیکه از مکان آن اطلاعی ندارند. علاوه بر آن، فرض شده است که تمام پرندگان با به اشتراک گذاشتن اطلاعات خود، موقعیت نزدیکترین پرنده به غذا را می‌دانند. در الگوریتم اجتماع ذرات هر جواب مسئله، یک پرنده در فضای جستجو است که ذره[۳] نام دارد. هر ذره دارای یک مقدار شایستگی است که توسط تابع شایستگی مسئله بدست می‌آید. پرنده‌ای که به غذا نزدیک‌تر است، شایستگی بیشتری دارد. این الگوریتم ماهیت پیوسته دارد و در کاربردهای متعدد کارایی خود را ثابت کرده است.

امروزه این الگوریتم در بسیاری از کاربردها از قبیل انتخاب ویژگی[۴و۵]، تنظیم وزن‌های شبکه‌های عصبی مصنوعی[۶ و ۷]، خوشه‌بندی داده‌ها [۸]، حل مسائل ترکیبیاتی همچون فروشنده دوره گرد [۹]، جایابی قطعات و مسیریابی در ادوات FPGA [۱۰]، طراحی کنترل کننده PID [۱۱]، جایابی ادوات SVC و TCSC در شبکه‌های قدرت [۱۲و۱۳] و بازآرایی شبکه‌های قدرت [۱۴] استفاده می‌شود.

۲-۱- الگوریتم بهینه‌سازی اجتماع ذرات

بهینه ساز اجتماع ذرات با یک جمعیت از جواب‌های تصادفی (ذره‌ها) شروع به کار می‌کند، سپس برای یافتن جواب بهینه در فضای مسئله با به روز کردن مکان ذره‌ها به جستجو می‌پردازد. با فرض D بعدی بودن فضای جستجو، ذره i ام با یک بردار D بعدی بصورت $X_i = (x_{i1}, x_{i2}, ..., x_{iD})^T$ قابل نمایش است. سرعت این ذره (تغییر مکان در فضای جستجو)، با یک بردار D بعدی بصورت $V_i = (v_{i1}, v_{i2}, ..., v_{iD})^T$ نمایش داده می‌شود. بهترین مکان دیده شده در موقعیت‌های قبلی ذره i ام، بصورت $P_i = (p_{i1}, p_{i2}, ..., p_{iD})^T$ بیان می‌شود. از پارامتر g برای نمایش بهترین ذره جمعیت استفاده می‌شود. از این رو بهترین موقعیت دیده شده در کل جمعیت با بردار $P_g = (p_{g1}, p_{g2}, ..., p_{gD})^T$ ارائه می‌شود. در هر مرحله از حرکت جمعیت، سرعت و مکان هر ذره بعد از یافتن دو مقدار بهترین (P_i و P_g) طبق روابط ۱ و ۲ به روز می‌شود [۱].

$$v_{id}(t+1) = w.v_{id}(t) + c_1.rand(p_{id}(t) - x_{id}(t)) + c_2.Rand(p_{gd}(t) - x_{id}(t)) \quad (۱)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad i = 1,2,...,N \quad d = 1,2,...,D \quad (۲)$$

که در رابطه ۱، w وزن اینرسی در بازه [۰،۱]، c_1 و c_2 ضرایب یادگیری یا شتاب در بازه [۲،۱] (که معمولا $c_1 = c_2$ در نظر گرفته می‌شود) و $rand$ و $Rand$ اعدادی تصادفی در بازه [۱،۰] و N بیان کننده تعداد ذرات است. همچنین مقدار نهائی سرعت هر ذره برای جلوگیری از واگرائی الگوریتم به یک بازه محدود مشخص می‌شود، $v_{id} \in [-v_{\max}, v_{\max}]$. شرط خاتمه الگوریتم، همگرایی آن یا توقف بعد از تعداد معینی از تکرار است.

مدلی که طی رابطه‌های ۱ و ۲ ارائه شد، مدل کلی[۴] نام دارد. برای این الگوریتم، مدل دیگری به نام مدل محلی[۵] نیز ارائه شده است. در مدل محلی، برای هر ذره یک همسایگی تعریف شده و بهترین موقعیت دیده شده در تمام همسایگان ذره محاسبه شده و با $L_i = (l_{i1}, l_{i2}, ..., l_{iD})^T$ نمایش داده می‌شود. در این شرایط به جای آنکه تمام ذرات در به روز رسانی از P_g استفاده کنند، هر ذره از L_i مربوط به خود استفاده می‌کند. آزمایش‌ها نشان داده است که مدل محلی در حل مسائل دنیای پیوسته به مراتب بهتر از مدل کلی عمل می‌کند.

۲-۲- الگوریتم بهینه‌سازی اجتماع ذرات باینری

الگوریتم اجتماع ذرات باینری، با باز-تعریف مفهوم سرعت، انشعابی از الگوریتم فضای پیوسته است. در نسخه باینری، موقعیت هر ذره در هر بعد به دو مقدار صفر و یک محدود می‌شود. یعنی هر ذره، در یک فضا محدود به صفر و یک حرکت می‌کند. در این حالت فضای جستجو، یک ابر مکعب است که ذرات از یک راس آن به راس دیگر جابجا می‌شوند. در الگوریتم نسخه باینری، مفهوم سرعت به مفهوم احتمال، تغییر یافته و v_{id} احتمال یک بودن x_{id} را بیان می‌کند. بدین معنی که مقدار v_{id} به یک مقدار بین [۰،۱] نگاشت شده که این مقدار بیانگر احتمال یک بودن X_{id} است. برای پیاده‌سازی رویکرد باینری‌سازی الگوریتم، ابتدا سرعت ذره در هر بعد با استفاده از رابطه ۱ محاسبه می‌شود. سپس، این مقدار با استفاده از تابع محدود کننده سیگموید (رابطه ۳) به مقداری بین صفر و یک نگاشت می‌شود و در نهایت موقعیت ذره i ام در بعد dام با رابطه ۴ به روز می‌شود [۲].

$$S(v_{id}) = Sigmoid(v_{id}) = \frac{1}{1+e^{-v_{id}}} \quad (۳)$$

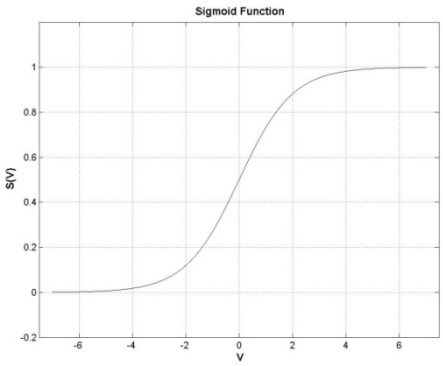
$$if \quad rand < S(v_{id}(t+1)) \quad then \quad x_{id}(t+1) = 1$$
$$else \quad x_{id}(t+1) = 0 \quad (۴)$$

که $rand$ همانند قبل یک عدد تصادفی در بازه [۰،۱] است. همانطور که مقدار v_{id} در الگوریتم پیوسته به یک بازه متقارن محدود می‌شود، در الگوریتم باینری نیز v_{id} به یک مقدار بیشینه $v_{\max}$ محدود می‌شود ($|v_{id}| < v_{\max}$). بطور متداول مقدار $v_{\max}$ برابر ۶ در نظر گرفته می‌شود. اگرچه با این مقدار، خروجی تابع احتمال بکار رفته به بازه [۰/۰۰۲۵، ۰/۹۹۷۵] محدود می‌شود، اما این محدودیت باعث همگرایی بهتر الگوریتم می‌شود. لازم به ذکر است که الگوریتم باینری نیز مشابه نوع حقیقی با استفاده از مدل‌های کلی و محلی قابل پیاده‌سازی است.

۳- معایب الگوریتم اجتماع ذرات باینری متداول

ذره‌ها در الگوریتم اجتماع ذرات پیوسته، با دو کمیت x و v تعریف می‌شوند که x موقعیت ذره‌ها، همان جواب‌های بالقوه مسئله و v معرف سرعت برای هر ذره است که نشان دهنده میزان تغییر در موقعیت آینده ذره، نسبت به موقعیت فعلی آن است. مقادیر بزرگ v نشان می‌دهد که موقعیت جاری ذره مناسب نبوده و ذره تا رسیدن به مکان بهینه، در فضای مسئله فاصله زیادی دارد. بنابراین جابجائی بیشتری برای ذره لازم است تا خود را به مکان بهینه نزدیک کند (رابطه ۲). از طرف دیگر، مقادیر کوچک v حاکی از نزدیک شدن مکان ذره به جواب مسئله است. به گونه‌ای که اگر موقعیت ذره با موقعیت بهینه یکی شود سرعت ذره صفر شده و این نشان دهنده آن است که دیگر جابجائی ذره لازم نیست.

درنسخه باینری الگوریتم، تعاریف متفاوتی از x و v ارائه شده است. بطوریکه سرعت ذره به جای آنکه ذره را به سمت جواب‌های بهینه مساله هدایت کند، بصورت تابع احتمالات، یک یا صفر شدن x را مطرح می‌کند. به عبارت دیگر، مقدار v_{id} احتمال اینکه مقدار x_{id} یک یا صفر باشد را تعیین می‌کند. از آنجاییکه مقدار احتمال باید در بازه صفر و یک بگنجد، v_{id} از تابع محدود کننده سیگموید می‌گذرد. این تابع در شکل ۱ آمده است.



شکل۱- تابع محدود کننده سیگموید $S(v_{id})$

همانطور که مشاهده می‌شود در نسخه باینری، بزرگ بودن مقدار v_{id} نشان دهنده تغییرات زیاد x_{id} نیست، بلکه متناسب با بزرگ بودن v_{id} (در جهت مقادیر مثبت) احتمال یک بودن x_{id}، افزایش می‌یابد. به همین نحو متناسب با بزرگ بودن v_{id} (در جهت مقادیر منفی) احتمال صفر بودن x_{id}، زیاد می‌شود. از سوی دیگر، اگر v_{id} برابر صفر شود مقدار x_{id} ثابت نمانده و با احتمال ۵۰٪ یک و با همین احتمال صفر خواهد بود. از سوی دیگر ملاحظه می‌شود که یک یا صفر شده x_{id} بودن در نظر گرفتن مقدار قبلی آن انجام می‌شود. با توجه به آنچه که بحث شد، می‌توان ایرادات زیر را به نسخه باینری متداول وارد دانست [۱۵].

الف) ایراد نخست از تابع احتمال فعلی (تابع سیگموید، رابطه ۳) است. در حالیکه بزرگ شدن v_{id} در جهت مثبت و منفی از نظر مفهومی با یکدیگر فرقی ندارد و نشان‌دهنده این است که موقعیت ذره برای این بعد خاص باید تغییر کند، در الگوریتم باینری متداول بین این دو حالت فرق قائل شده است به گونه‌ای که بزرگ شدن آن در جهت مثبت باعث افزایش احتمال یک شدن موقعیت ذره و بزرگ شدن آن در جهت منفی باعث افزایش احتمال صفر شدن آن می‌شود. همچنین، وقتی سرعت ذره برای یک بعد مشخص به صفر نزدیک می‌شود، به معنی آن است که ذره در آن بعد موقعیت مناسبی دارد و باید موقعیت ذره تغییر نکند. این در حالی است که با تابع احتمال سیگموید، احتمال صفر یا یک شدن موقعیت ذره در آن بعد برابر خواهد شد.

ب) ایراد دوم از رابطه به روزرسانی موقعیت ذره (رابطه۴) است. در این رابطه، موقعیت قبلی ذره برای محاسبه موقعیت بعدی آن در نظر گرفته نمی‌شود.

نتایج آزمایش‌ها نشان‌دهنده آن است که افزایش میانگین در تکرارهای ابتدایی الگوریتم باینری، مشاهده می‌شود که این نشان دهنده نزدیک شدن ذره‌ها به جواب بهینه است ولی در ادامه، ذره‌ها از جواب بهینه دور می‌شوند. هنگامی که الگوریتم به جواب بهینه رسید باید احتمال تغییر موقعیت ذره‌ها به صفر نزدیک شود در حالیکه درست در همین موقع، احتمال صفر شدن یا یک شدن بدون توجه به حالت قبلی به ۰/۵ می‌رسد و همین موضوع همگرایی الگوریتم را به مخاطره می‌اندازد.

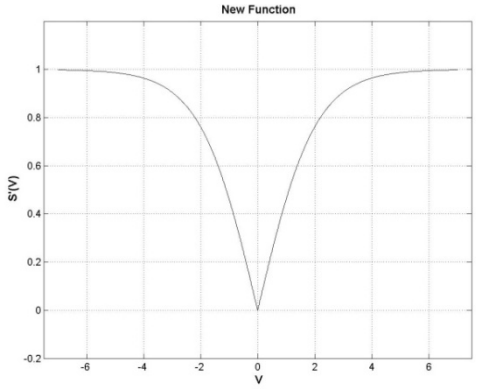
۴- الگوریتم باینری جدید

در این بخش الگوریتم باینری پیشنهادی ارائه سپس، کارایی آن روی توابع محک استاندارد آزموده شده و نتایج تعبیر و تفسیر خواهد شد.

۴-۱- معرفی الگوریتم پیشنهادی

ضعف نخست الگوریتم باینری متداول با ارائه یک تابع احتمال مناسب قابل برطرف کردن است. از طرف دیگر برای برطرف کردن ضعف ساختاری دوم الگوریتم، باید رابطه بروزرسانی موقعیت ذره (رابطه ۴) تصحیح گردد. در الگوریتم پیشنهادی با ایجاد دو تغییر نسبت به الگوریتم متداول، مفاهیم x_{id} و v_{id} با مدل اصلی سازگار شده و نتایج حاصله نشان از همگرائی جمعیت دارد. در الگوریتم پیشنهادی به جای تابع سیگموید از تابع رابطه ۵ (شکل ۲) استفاده شده است. با جایگزینی تابع احتمال رابطه ۵ به جای رابطه ۳، مشکل الف که در بخش ۳ به آن پرداخته شد، برطرف می‌شود. به عبارت دیگر، با تابع جدید میان سرعت‌های بزرگ مثبت و منفی دیگر فرقی نیست. به عبارت دیگر در این تابع وقتی مقدار سرعت بزرگ می‌شود، خروجی تابع نیز افزایش می‌یابد. از سوی دیگر وقتی که سرعت به صفر نزدیک می‌شود، خروجی تابع احتمال پیشنهادی نیز به صفر نزدیک می‌شود. همچنین، برای رفع مشکل ب و در نظر گرفتن وضعیت فعلی موقعیت ذره برای تعیین موقعیت بعدی آن، رابطه ۶ جایگزین رابطه ۴ شده است.

با تصحیحات بعمل آمده، ملاحظه می‌شود که بزرگ بودن v_{id} نشان دهنده موقعیت نامناسب ذره است و باعث تغییر آن از صفر به یک یا از یک به صفر می‌شود و کوچک بودن v_{id} احتمال تغییر x_{id} را کاهش می‌دهد و در نهایت زمانی که v_{id} صفر شود موقعیت ذره بدون تغییر باقی می‌ماند. در رابطه ۵، α یک ضریب ثابت است که مقدار آن حدود یک انتخاب می‌شود.



شکل۲- تابع $S'(v_{id})$

$$S'(v_{id}) = \left|\tanh(\alpha\, x_{id})\right| \quad (۵)$$

$$if \;\; rand < S'(v_{id}(t+1)) \;\; then \;\; x_{id}(t+1) = complement(x_{id}(t)) \quad (۶)$$

$$else \;\; x_{id}(t+1) = x_{id}(t)$$

۴-۲- بررسی کارایی الگوریتم پیشنهادی و نتایج آزمایش‌ها

برای آزمودن میزان کارایی الگوریتم پیشنهادی از ۱۱ تابع محک استاندارد استفاده شده است[۱۶]. جزئیات توابع محک در جدول ۱ آمده است.

جدول۱- توابع محک استفاده شده در آزمایش‌ها، n مشخص کننده بعد مساله و F_{opt} بیانگر مقدار بهینه تابع است. در توابع F_1 تا F_{10} یافتن کمینه و در تابع $Max-Ones$ یافتن بیشینه مد نظر است

Test Function	S	F_{opt}
$F_1(x)=\sum_{i=1}^{n} x_i^2$	$[100,100]^n$	0
$F_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	$[-10,10]^n$	0
$F_3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_i\right)^2$	$[100,100]^n$	0
$F_4(x)=\max_i\{\lvert x_i\rvert, 1\le i\le n\}$	$[-100,100]$	0
$F_5(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	$[-30,30]^n$	0
$F_6(x)=\sum_{i=1}^{n}\left(\lfloor x_i+0.5\rfloor\right)^2$	$[100,100]^n$	0
$F_7(x)=\sum_{i=1}^{n} i x_i^4+random[0,1)$	$[1.28,1.28]^n$	0
$F_8(x)=\sum_{i=1}^{n} x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	$[-500,500]^n$	*
$F_9(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	$[5.12,5.12]^n$	0
$F_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	$[-32,32]^n$	0
$F_{11}(x)=Max-Ones=\sum_{i=1}^{100} x_i$	$\{0,1\}^{100}$	100

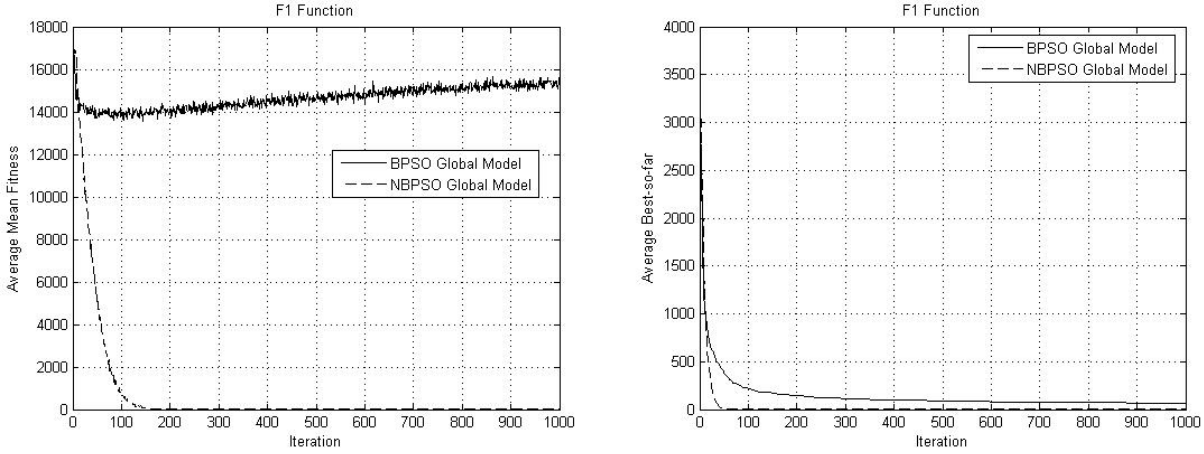
* مقدار مینیمم تابع F_8 بستگی به n دارد و برای $n=5$ این مقدار ۲۰۹۴/۹۱۶- است.

در ده تابع اول، مقدار کمینه تابع و در تابع یازدهم مقدار بیشینه تابع مد نظر است. توابع F_1 تا F_7 از نوع توابعی هستند که صرفاً یک کمینه فرامحلی دارند یا به عبارتی تک بهینه‌ای[۶] هستند. برای این توابع یافتن سریع این کمینه از اهمیت برخوردار است. در این بین، تابع F_6 یک تابع پله خطی- تکه‌ای پیوسته است و تابع F_7 یک تابع کوادراتیک در حضور نویز است. توابع F_8 تا F_{10} از نوع توابع با تعداد کمینه‌های محلی فراوانند یا به عبارتی چند بهینه‌ای[۷] هستند. در این توابع زمان رسیدن به مقدار بهینه به اندازه رسیدن به آن اهمیت ندارد و روش جستجویی برتر است که در مینیمم‌های محلی گرفتار نشده و توان یافتن مقدار بهینه فرامحلی را داشته باشد. تابع ۱۱، تابع $Max-Ones$ است که از نوع توابع پله گسسته است.

الگوریتم پیشنهادی (NBPSO[۸])، به همراه نوع متداولش (BPSO)، در حل مسائل جدول ۱ در شرایطی برابر پیاده‌سازی شده‌اند. هر دو روش به مدل‌های محلی و کلی پیاده‌سازی شده‌اند. در روش محلی از همسایگی ۳ استفاده شده است. تعداد جمعیت برابر ۵۰، در نظر گرفته شده و برای مقایسه، متوسط شایستگی[۹] جمعیت و بهترین جوابی که تا آن لحظه[۱۰] مشاهده شده، محاسبه شده است. این پارامترها برای ۵۰ مرتبه اجرای مستقل برنامه محاسبه شده و متوسط نتایج آن و همچنین مقدار میانه[۱۱] بهترین جواب دیده شده، آمده است. سایر پارامترهای الگوریتم به این صورت تنظیم شده‌اند که $v_{\max}=۶$، $c_1=c_2=۲$ و w نیز بطور خطی از مقدار ۰/۶ تا ۰/۲ بطور کاهشی در نظر گرفته شده است. ده تابع اول، با بعد ۵ فرض شده‌اند (یعنی اینکه $n=۵$) و برای هر بعد مساله ۱۵ بیت لحاظ شده است. بنابراین هر ذره، ۷۵ بعد دارد. تابع یازدهم، یک تابع گسسته است که بعد آن ۱۰۰ بیت در نظر گرفته شده است. از آنجایی که ارائه گرافیکی نتایج تمام توابع حجم زیادی اشغال می‌کند، صرفاً نتایج چند تابع بصورت گرافیکی در خلال شکل‌های ۳ تا ۱۰ ارائه و تفسیر می‌شود.

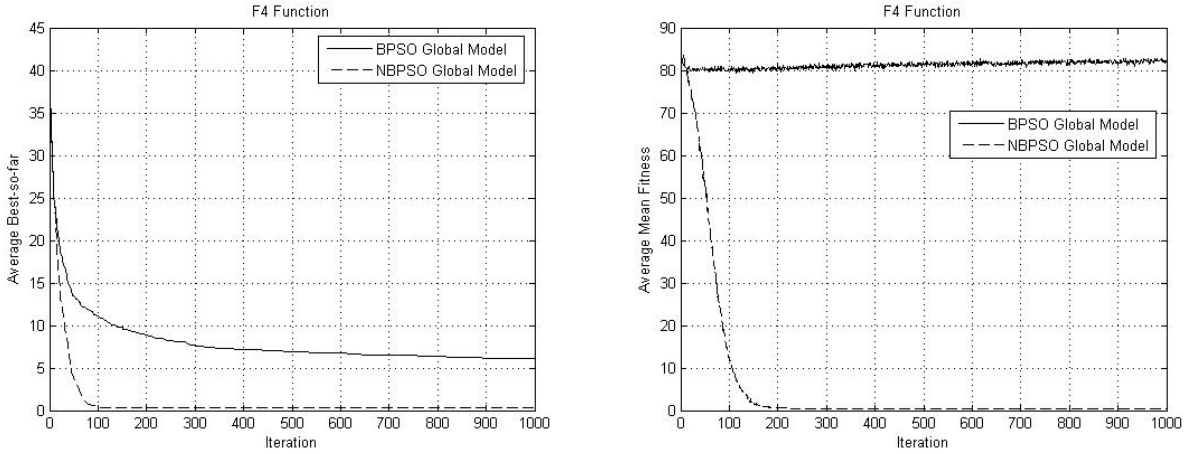
در شکل‌های ۳ تا ۷ به ترتیب نتایج بهینه‌سازی توابع F_1، F_4، F_7، F_9 و F_{11} با مدل کلی ارائه شده است. در شکل‌های الف، گراف مربوط به متوسط شایستگی و در شکل‌های ب، گراف مربوط به بهترین جواب دیده شده برای دو روش متداول و پیشنهادی در کنار یکدیگر آمده است. همانگونه که ملاحظه می‌شود، تحلیل‌های انجام شده در قسمت‌های قبل نمایان شده است. با توجه به مشکلات ساختاری الگوریتم متداول، در هیچیک از توابع ارائه شده متوسط شایستگی آن به سمت بهینه رشد نکرده است. علاوه بر آن، بهترین جواب دیده شده توسط الگوریتم نیز جواب مناسب و قابل قبولی نیست. در این مقایسه مشاهده می‌شود که روش پیشنهادی به خوبی توانسته است معایب روش متداول را بهبود بخشد. چنانچه ملاحظه می‌شود، نه تنها الگوریتم همگرا می‌شود بلکه جواب‌های قابل قبولی نیز ارائه می‌شود. از آنجایی که در شکل‌های ۳ تا ۷ صرفاً به بیان نتایج الگوریتم در مدل کلی پرداخته شد، شکل۸ نتایج بهینه‌سازی تابع F_9 را با مدل محلی به تصویر کشیده است. در مقایسه شکل ۶ و ۸ مشاهده می‌شود که در الگوریتم باینری متداول، متوسط شایستگی جمعیت در مدل محلی نیز مانند مدل کلی، وضعیتی آشفته دارد. این در حالی است که الگوریتم پیشنهادی با هر دو ریکرد محلی و کلی بخوبی عمل کرده است. برای مقایسه بهتر الگوریتم پیشنهادی با مدل‌های محلی و کلی، نتایج آزمایش‌های انجام شده روی دو تابع F_6 و F_9 در شکل‌های ۹ و ۱۰ ارائه شده‌اند. همانگونه که این دو شکل نشان می‌دهند، الگوریتم پیاده‌سازی شده با مدل کلی به سرعت به سمت جواب بهینه همگرا می‌شود در حالیکه در مدل محلی، حرکت به سمت بهینه با آهنگی کندتر صورت می‌گیرد. البته این موضوع طبیعی است و به ماهیت این دو روش برمی‌گردد. در مدل کلی تمام ذرات، ذره‌ی با بهترین شایستگی را تعقیب می‌کنند و در مدل محلی هر چند ذره در یک همسایگی مشخص، یک ذره خاص را تعقیب می‌کند. از اینرو، در روش کلی همگرایی سریعتر از روش محلی اتفاق می‌افتد. مقایسه گرافیکی این دو تابع نشاندهنده آن است که استفاده از مدل کلی برای توابع با یک بهینه (مانند F_6)

باعث سرعت پاسخ بیشتر الگوریتم می‌شود. از سوی دیگر استفاده از مدل محلی برای توابع با چندین بهینه (مانند F_9) بواسطه فرار از رکود و همگرا شدن به بهینه‌های محلی نتایج بهتری خواهد داد. نتایج آزمایش‌های انجام شده در این بخش در جدول ۲ آمده است. میانگین متوسط شایستگی، میانگین و میانه بهترین جواب دیده شده در تکرار ۱۰۰۰ الگوریتم برای ۵۰ اجرای مستقل محاسبه و در این جدول آمده است.



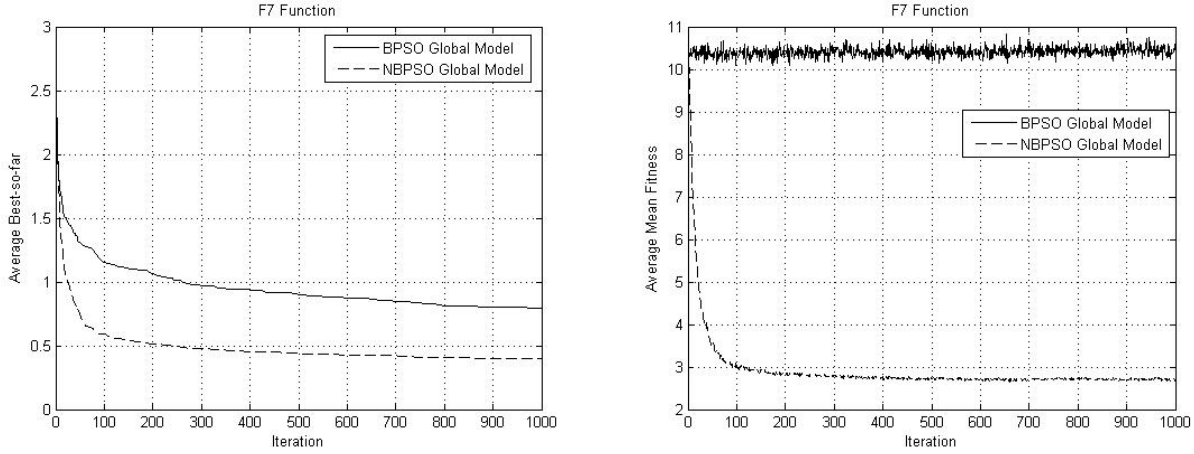
(ب) (الف)

شکل۳- نتایج بهینه‌سازی تابع F_1 در مدل کلی الف) متوسط شایستگی جمعیت ب) متوسط بهترین جواب



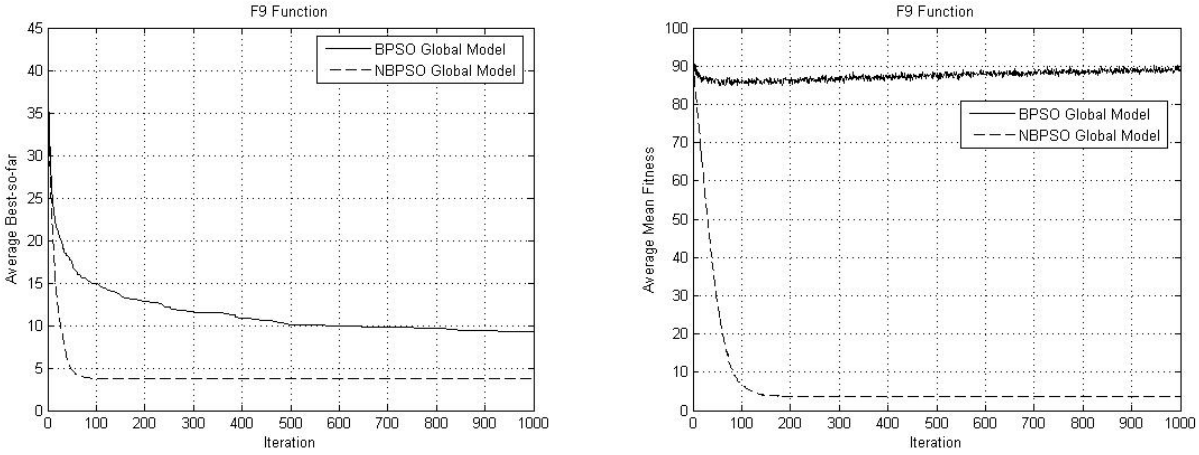
(ب) (الف)

شکل۴- نتایج بهینه‌سازی تابع F_4 در مدل کلی الف) متوسط شایستگی جمعیت ب) متوسط بهترین جواب



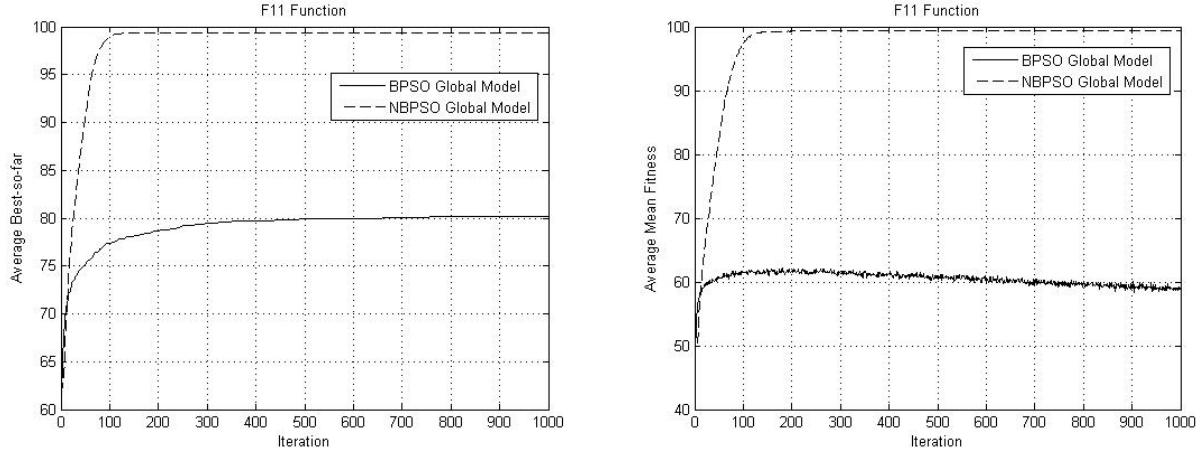
(ب) (الف)

شکل۵- نتایج بهینه‌سازی تابع F_7 در مدل کلی الف) متوسط شایستگی جمعیت ب) متوسط بهترین جواب



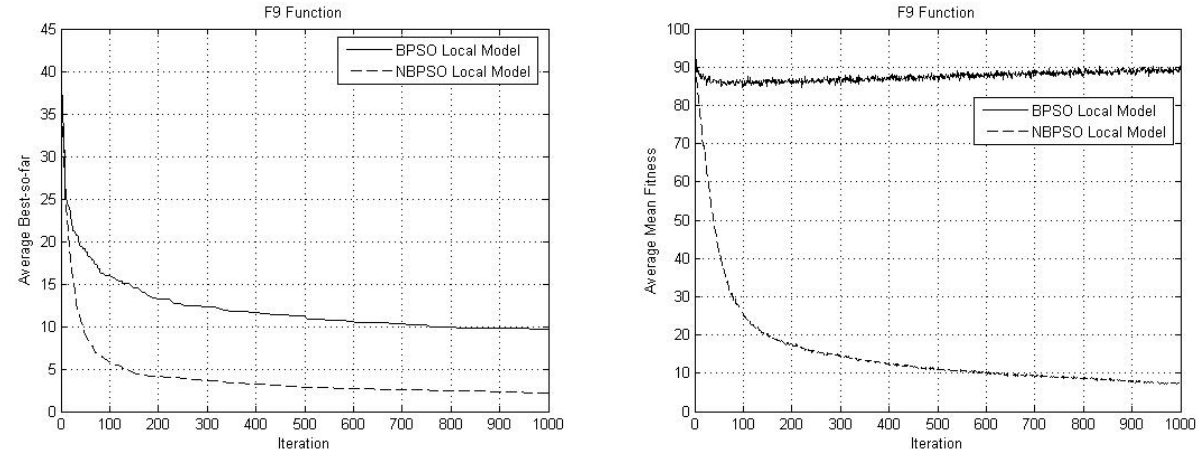
(ب) (الف)

شکل۶- نتایج بهینه‌سازی تابع F_9 در مدل کلی الف) متوسط شایستگی جمعیت ب) متوسط بهترین جواب



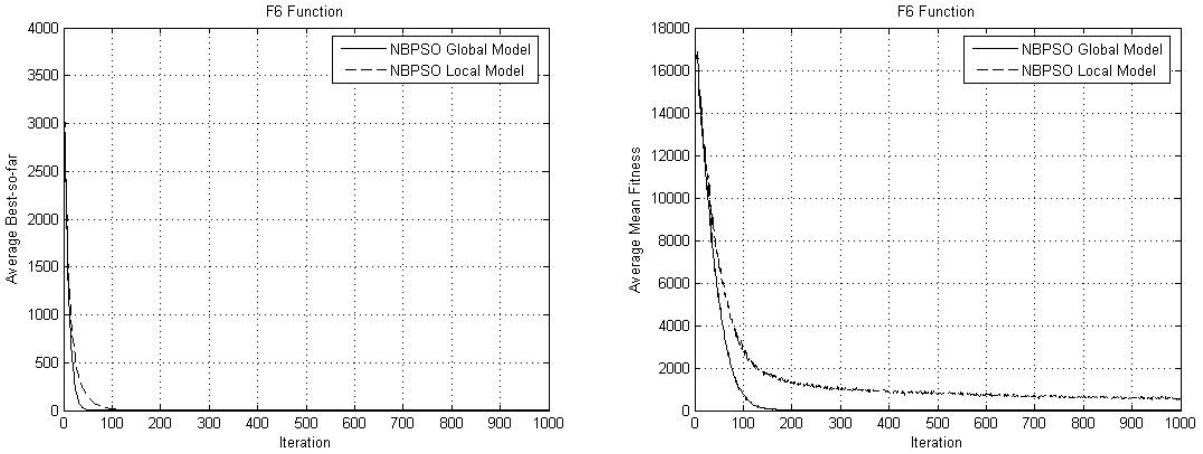
(ب) (الف)

شکل۷- نتایج بهینه‌سازی تابع F_{11} در مدل کلی الف) متوسط شایستگی جمعیت ب) متوسط بهترین جواب



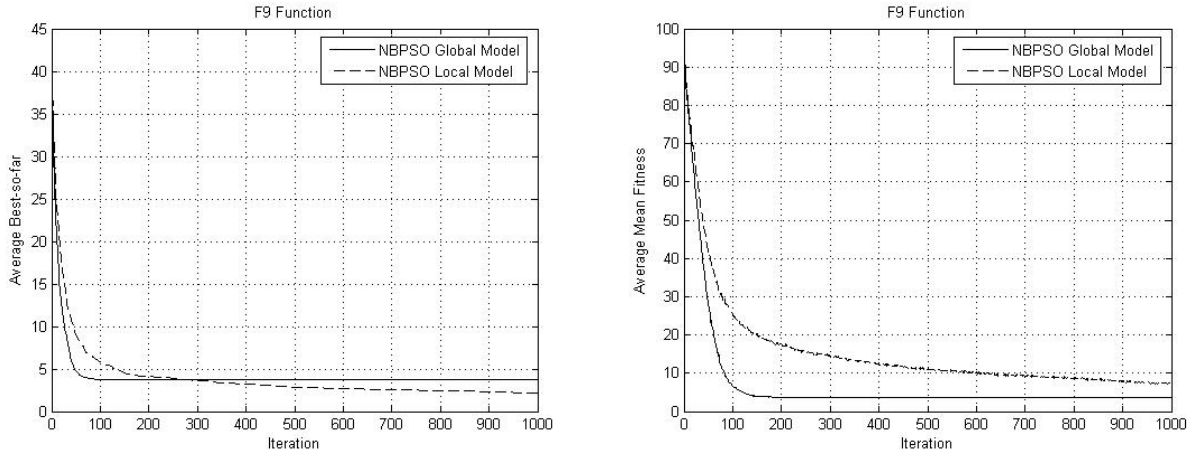
(ب) (الف)

شکل۸- نتایج بهینه سازی تابع F_9 در مدل محلی الف) متوسط شایستگی جمعیت ب) متوسط بهترین جواب



(ب) (الف)

شکل۹- مقایسه مدل‌های کلی و محلی در بهینه‌سازی تابع F_6 الف) متوسط شایستگی جمعیت ب) متوسط بهترین جواب



(ب) (الف)

شکل۱۰- مقایسه مدل‌های کلی و محلی در بهینه‌سازی تابع F_9 الف) متوسط شایستگی جمعیت ب) متوسط بهترین جواب

جدول۲- نتایج پیاده‌سازی الگوریتم پیشنهادی و الگوریتم متداول روی توابع جدول ۱ متوسط شایستگی جمعیت (Ave.fit)، بهترین جواب (Ave.best) برای مرحله ۱۰۰۰ الگوریتم محاسبه شده و برای ۵۰ اجرای مستقل میانگین آنها به همراه میانه بهترین جواب دیده شده (Median) ارائه شده است

تابع	مدل کلی						مدل محلی					
	روش متداول (BPSO)			روش پیشنهادی (NBPSO)			روش متداول (BPSO)			روش پیشنهادی (NBPSO)		
	Ave.fit	Ave.best	Median	Ave.fit	Ave.best	Median	Ave.fit	Ave.best	Median	Ave.fit	Ave.best	Median
F_1	۱/۵۳*۱۰^۴	۶۹/۱۳	۷۱/۹۷	**۰/۰۴۴۰**	**۰/۰۴۴۰**	**۰/۰۰۳۵**	۱/۵۰*۱۰^۴	۸۶/۴۹	۸۷/۱۴	۵۹۹/۱۳	۰/۳۶۱۴	۰/۰۱۱۴
F_2	۲۲۱۰	۱/۲۵	۱/۲۷	**۰/۰۱۷۴**	**۰/۰۱۷۴**	۰/۰۰۸۲	۲۲۱۹/۲	۱/۴۰	۱/۴۵	۸/۹۱۳۰	۰/۰۲۳۱	**۰/۰۰۷۶**
F_3	۴/۶۹*۱۰^۴	۶۶/۴۳	۶۲/۴۸	**۲۴/۹۴۳**	۲۴/۹۴۳۵	**۴/۹۹۹۰**	۴/۶۷*۱۰^۴	۹۷/۱۸	۹۲/۷۹	۲۰۰۴/۴	**۱۰/۶۸۳۰**	۵/۲۰۵۰
F_4	۸۲/۴۴	۶/۰۸	۵/۹۳	**۰/۲۸۸۷**	**۰/۲۸۸۵**	**۰/۱۹۸۳**	۸۱/۸۵	۶/۹۴	۷/۱۶	۵/۴۸۰۰	۰/۶۸۹۳	۰/۴۱۸۱
F_5	۶/۱۱*۱۰^۷	۷۳۵/۱۵	۵۸۴/۹۵	**۱۵۵/۸۰**	۱۵۵/۸۰	**۴/۵۰۸۰**	۵/۹۱*۱۰^۷	۱۰۹۵/۸	۹۲۵/۶۱	۲/۲۷*۱۰^۶	**۱۷/۵۷۷۰**	۷/۹۸۸۰
F_6	۱۵۲۷۴/۲	۶۵/۳۵	۶۵/۵۷	**۰/۶۸۱۹**	**۰/۶۸۱۹**	**۰/۵۹۷۰**	۱۵۹۶۷/۳	۸۷/۵۱	۸۰/۱۶	۵۴۳/۶۸	۰/۷۳۳۵	۰/۶۵۴۱
F_7	۱۰/۴۵	۰/۷۹	۰/۸۳	**۲/۷۲۷۰**	**۰/۳۹۴۷**	**۰/۳۹۴۰**	۱۰/۳۵	۰/۸۱	۰/۷۹	۳/۲۲۳۰	۰/۳۹۹۰	۰/۴۰۳۷
F_8	-۲۲۴/۸۶	-۲۰۰۰/۳	-۲۰۰۲/۳	-۲۰۴۰/۳	-۲۰۴۰/۳	-۲۰۶۰/۲	-۲۴۴/۷۴	-۱۹۷۳/۴	-۱۹۶۳/۳	**-۲۰۹۴/۰**	**-۲۰۹۴/۴**	**-۲۰۹۴/۸**
F_9	۸۸/۷۴	۹/۲۵	۹/۲۵	**۳/۶۹۲۰**	۳/۶۹۲۰	۳/۲۸۸۰	۸۸/۹۴	۹/۶۸	۹/۷۹	۷/۲۳۴۰	**۲/۱۳۵۶**	**۲/۲۶۱۰**
F_{10}	۲۰/۷۳	۵/۵۲	۵/۴۸	**۰/۴۸۲۱**	**۰/۴۸۲۱**	**۰/۰۷۱۶**	۲۰/۷۵	۶/۰۰	۵/۹۰	۲/۴۸۸۰	۰/۵۷۳۴	۰/۱۶۲۵
F_{11}	۵۹/۰۹	۸۰/۲۲	۸۰	۹۹/۳	۹۹/۳	۹۹	۵۷/۷۵	۷۸/۸۲	۷۹	**۱۰۰**	**۱۰۰**	**۱۰۰**

نتایج بهینه‌سازی توابعی که نتایج آنها بصورت گرافیکی ارائه نشد، با نتایج توابعی که نتایج آنها بصورت گرافیکی ارائه شده است، همخوانی دارد. نتایج جدول ۲ و شکلهای ۳ تا ۱۰ علاوه بر آنکه برتری مطلق روش پیشنهادی را بر روش متداول بیان می‌دارد به چند نکته دیگر نیز اشاده دارد که در ادامه فهرست وار می‌آید.

الف) استفاده از مدل کلی برای بهینه‌سازی توابع تک بهینه، سریعتر و بهتر از مدل محلی است.

ب) استفاده از مدل محلی برای بهینه‌سازی توابع چند بهینه، نتایج بهتر و مناسب‌تری بدست می‌دهد.

ج) روش پیشنهادی در حالت‌های کلی و محلی بعد از حدود ۱۰۰ تکرار الگوریتم و قبل از یافتن دقیق پاسخ بهینه دچار رکود می‌شود. نتایج این آزمایش‌ها نشان می‌دهد که توقف الگوریتم در نزدیکی بهینه اتفاق می‌افتد. این موضوع بیانگر آن است که الگوریتم پیشنهادی توان کشف پاسخ را داشته، اما با دلایلی متوقف شده است. اگر چه این رکود برای الگوریتم محلی در تکرارهای بالاتری اتفاق می‌افتد، اما آنچه مهم است این است که اتفاق می‌افتد. در زیر- بخش بعدی به دلایل رکود الگوریتم پرداخته می‌شود.

۴-۳- تحلیلی بر دلایل رکود الگوریتم و همگرایی زودرس

دلیل اصلی رکود در الگوریتم باینری پیشنهادی را باید در الگوریتم اجتماع ذرات اصلی (نوع پیوسته) جستجو کرد. همانگونه که ملاحظه می‌شود، در الگوریتم اجتماع ذرات، تمام ذرات رفته رفته با تعقیب ذره‌ای که بهترین موقعیت (P_g) را مشاهده کرده است، به سمت آن همگرا می‌شوند. بنابراین چنانچه این ذره در یک بهینه محلی گرفتار شود، تمام جمعیت به این بهینه محلی همگرا خواهند شد و همگرایی زودرس اتفاق خواهد افتاد.

یکی از مشکلات شناخته شده الگوریتم اجتماع ذرات این است که برای ذره ای که شامل موقعیت P_g است، ترمهای دوم و سوم رابطه بروز کردن سرعت ذره (رابطه ۱) صفر خواهد شد. بنابراین ذره در راستای بردار حرکت قبلی خود حرکت خواهد کرد. از طرفی چون بطور معمول w کوچکتر از یک است، این ترم نیز میرا خواهد شد. بنابراین ذره شامل P_g ثابت خواهد شد و سایر ذرات نیز به این ذره همگرا می‌شوند. در بسیاری از موارد، این موضوع باعث همگرایی زودرس الگوریتم به یک بهینه محلی خواهد شد.

حال باتوجه به مشکلات موجود در الگوریتم اجتماع ذرات، می‌توان به بررسی الگوریتم اجتماع ذرات باینری پیشنهادی(NBPSO) پرداخت. هنگامی که الگوریتم اجتماع ذرات دچار همگرایی می‌شود، ذرات به سمت یک بهینه محلی حرکت کرده و در آن گرفتار می‌شوند. در این حالت، از آنجایی که ذرات به ذره با موقعیت P_g نزدیک شده‌اند، سرعت آنها به صفر نزدیک می‌شود. در چنین حالتی احتمال تغییر مکان ذره‌ها در الگوریتم باینری(روابط ۵ و ۶، شکل۲) کم شده و به سمت صفر میل می‌کند. این به منزله آن است که الگوریتم باینری نیز به مثابه الگوریتم پیوسته دچار رکود شده و در بهینه محلی گرفتار می‌شود.

۵- بهبود الگوریتم پیشنهادی

دو راه حل برای بهبود الگوریتم NBPSO وجود دارد. یکی اینکه الگوریتم اجتماع ذرات اصلی(نوع پیوسته، رابطه۱) اصلاح شود. در این صورت الگوریتم باینری نیز تصحیح شده و گرفتار بهینه محلی نخواهد شد. راه دیگر این است که بدون توجه به مشکلات و معایب الگوریتم پیوسته به رفع معایب الگوریتم باینری پرداخته شود. چنانچه بخواهیم از روش اول استفاده شود، باید بعضی از اقسام الگوریتم اجتماع ذرات که معایب آنها برطرف شده است، جایگزین نسخه اصلی آن شوند. در روش دوم، باید تدبیری اندیشیده شود تا از رکود و همگرایی الگوریتم به بهینه‌های محلی جلوگیری شود. در این مقاله و در ادامه این بخش به بهبود الگوریتم پیشنهادی بر مبنای این دو راه حل پرداخته می‌شود.

۵-۱- بهبود الگوریتم پیشنهادی با برطرف کردن معایب الگوریتم اصلی

با توجه به مشکلاتی که الگوریتم اجتماع ذرات با آن روبرو است، تعمیمی بر آن که تضمین همگرایی را به همراه دارد در سال ۲۰۰۲ ارائه شد [۱۷و۱۸]. این نسخه تعمیم یافته GCPSO[۱۲] نام گرفته است. اصلاحات انجام شده در مقایسه با الگوریتم استاندارد تنها به عوض کردن معادله بروز رسانی سرعت ذره با موقعیت P_g محدود می‌شود. اگر از نماد τ برای نمایش این ذره استفاده شود، بروز رسانی سرعت این ذره با رابطه ۷ انجام می‌شود [۱۷].

$$v_{\tau d}(t+1) = w.v_{\tau d}(t) - x_{\tau d}(t) + p_{gd}(t) + \rho(t)(1-2rand) \qquad (۷)$$

چنانچه رابطه ۷ با رابطه ۱ در بروز رسانی سرعت ذره τ مقایسه شود، مشاهده می‌شود که ترم اول دو معادله یکسان است. در حالیکه ترم‌های دوم و سوم معادله اول برای ذره τ در معادله ۱ صفر می‌شوند، ترم $-x_{\tau d}(t)+p_{gd}(t)$ در معادله ۷، ذره τ را به موقعیت P_g برمی‌گردانند. در این قیاس، ترم $\rho(t)(1-2rand)$ باقی می‌ماند که به مثابه یک جستجوی اتفاقی پیرامون نقطه P_g عمل خواهد کرد و فضای اطراف آن را جستجو می‌کند. در این رابطه، پارامتر ρ کنترل کننده شعاع جستجوی پیرامون P_g است که بر اساس زمان تعریف می‌شود (رابطه۸).

$$\rho(t+1) = \begin{cases} 2\rho(t) & \text{if } \#\text{successes} \rangle S_C \\ 0.5\rho(t) & \text{if } \#\text{failures} \rangle f_C \\ \rho(t) & \text{otherwise} \end{cases} \qquad (۸)$$

در این رابطه #successes و #failures به ترتیب بیانگر تعداد دفعاتی است که موفقیت و شکست پی در پی اتفاق می‌افتد. یک "شکست" زمانی پیش می‌آید که $P_g(t) = P_g(t-1)$ باشد. در غیر این صورت حالت "موفقیت" روی داده است. در حالت اولیه $\rho(0)=1$ فرض می‌شود. S_C و f_C پارامترهای آستانه هستند که پیشنهاد می‌شود به ترتیب ۱۵ و ۵ در نظر گرفته شوند [۱۷]. برای اینکه رابطه ۸ به درستی تعریف شود، استفاده از قواعد رابطه ۹ الزامی است.

$$\begin{aligned} &if \quad \#successes(t+1) > \#successes(t) \Rightarrow \#failures(t+1) = 0 \\ &if \quad \#failures(t+1) > \#failures(t) \Rightarrow \#successes(t+1) = 0 \end{aligned} \qquad (۹)$$

با استفاده از قاعده‌های رابطه ۹، بعد از روی دادن هر موفقیت شمارنده مربوط به شکست‌ها بازنشانده می‌شود و برعکس. نتایج آزمایش نشان داده است که الگوریتم GCPSO بهتر از الگوریتم PSO عمل می‌کند. همانطور که پیش از این نیز ذکر شد، یکی از راه‌های جلوگیری از رکود الگوریتم باینری پیشنهادی، اصلاح الگوریتم اجتماع ذرات اصلی است. در این مقاله برای این منظور از نسخه GCPSO به جای نسخه PSO استفاده شده است [۱۹]. برای سادگی در نوشتار از این به بعد الگوریتم پیشنهادی که بر مبنای الگوریتم GCPSO پیاده‌سازی شده است را GCNBPSO[۱۳] می‌نامیم.

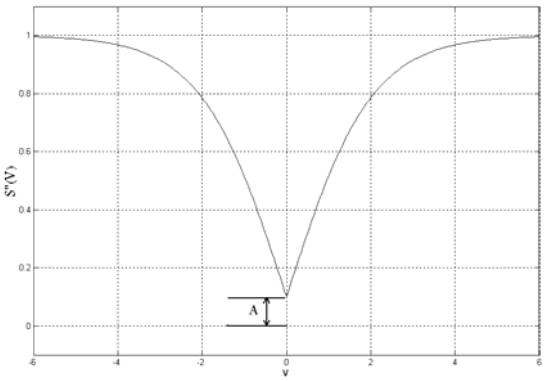
۵-۲- بهبود الگوریتم پیشنهادی در محیط گسسته

همانگونه که در بخش ۴-۳ ملاحظه شد، هنگامی الگوریتم اجتماع ذرات پیوسته در بهینه گرفتار می‌شود، الگوریتم باینری نیز گرفتار خواهد شد. برای گریز از این بحران پیشنهاد می‌شود که تابع رابطه ۵، به تابع رابطه ۱۰ تغییر یابد (شکل۱۱).

$$S''(v_{id}) = A + (1-A).S'(v_{id}) \\ = A + (1-A) \times \left|\tanh(\alpha\, v_{id})\right| \qquad (۱۰)$$

که در این رابطه ضریب A، برای جلوگیری از رکود الگوریتم استفاده شده است. پیش از این بیان شد، چنانچه الگوریتم دچار رکود شود، سرعت ذرات به صفر نزدیک شده و بنابراین با توجه به تابع احتمال رابطه ۵، احتمال تغییر موقعیت ذرات تقریباً غیر ممکن می‌شود. حال چنانچه از رابطه (۱۰) استفاده شود، حتی هنگامی که الگوریتم پیوسته دچار رکود شده و سرعت ذرات صفر شود، احتمال تغییر مکان ذرات برابر ضریب A خواهد بود. این بدین منزله است که الگوریتم قادر خواهد بود از بهینه محلی خارج شده و نقاط جدید را کاوش کند. روش مذکور شبیه عملگر جهش در الگوریتم‌های تکاملی است.

از آنجا که الگوریتم اجتماع ذرات پیوسته در تمام مسائل به بهینه محلی همگرا نمی‌شود، به نظر می‌رسد که استفاده از یک ضریب ثابت A خوشایند نیست. بنابراین پیشنهاد می‌شود که این ضریب، با توجه به وضعیت الگوریتم تغییر کند و مقدار مناسب را بخود بگیرد. یعنی زمانی که الگوریتم در بهینه محلی گرفتار شد، مقدار A زیاد شده و در غیر اینصورت مقدار A کاهش بیابد. با توجه به آنچه گفته شد، رابطه ۱۱ پیشنهاد می‌شود.



شکل۱۱- تابع $S''(v_{id})$

$$A = k(1 - e^{-\frac{F}{T}}) \qquad (۱۱)$$

که در این رابطه k، یک ضریب ثابت، T ثابت زمانی که با توجه به نوع و بعد مساله انتخاب می‌شود. F یک شمارنده است که دفعاتی را که الگوریتم دچار شکست می‌شود، می‌شمارد. شکست اینگونه تعریف می‌شود که الگوریتم نتواند موقعیت بهتری بدست آورد یا به عبارت دیگر $P_g(t) = P_g(t-1)$. بنابراین با هر دفعه شکست، به مقدار F یکی افزوده می‌شود. لازم به ذکر است که F در لحظه شروع برابر صفر است و پس از هر بار موفقیت الگوریتم، این شمارنده صفر خواهد شد. هنگامی که F صفر باشد، مقدار A برابر صفر خواهد بود. یعنی اینکه چنانچه الگوریتم در مسیر درست قرار داشته باشد، هیچگونه جهشی ایجاد نشده و الگوریتم باینری پیشنهادی در شکل متداول خود قرار دارد. برای سادگی در نوشتار از این به بعد الگوریتم باینری بهبود یافته در محیط گسسته را INBPSO[۱۴] می‌نامیم.

۵-۳- نتایج آزمایش

برای تعیین کارایی الگوریتم‌های ارائه شده در بخش‌های ۵-۱ (GCNBPSO) و ۵-۲ (INBPSO)، این دو الگوریتم در حل مسائل جدول۱ پیاده‌سازی شدند. برای پیاده سازی این دو الگوریتم از شرایطی معادل با آنچه الگوریتم‌های باینری (BPSO) متداول و باینری پیشنهادی (NBPSO) در بخش ۴-۲ پیاده‌سازی شده‌اند، استفاده شده است. هر دو مدل کلی و محلی در پیاده‌سازی بکار گرفته شده‌اند. در پیاده‌سازی الگوریتم INBPSO پارامترهای k و T به ترتیب برابر با ۱ و ۱۲۰۰ در نظر گرفته شده‌اند. نتایج پیاده‌سازی این دو الگوریتم در جدول ۳ خلاصه شده‌اند. برای مقایسه راحتر و بهتر نتایج الگوریتم باینری پیشنهادی که در جدول ۲ ارائه شده بود، نیز در این جدول ارائه شده است. همچنین نتایج حاصل از بهینه‌سازی چند تابع بصورت گرافیکی در شکل ۱۲ آمده است.

۵-۴- تحلیل و بررسی نتایج

بررسی نتایج، در دو بخش مدل کلی و محلی انجام می‌شود. مدل کلی روش‌های بهبود یافته (GCNBPSO، INBPSO) توانسته است تا حد قابل قبولی کیفیت پاسخها را افزایش داده و باعث بهبود الگوریتم شود. اما در این بین، روش بهبود یافته INBPSO به مراتب بهتر از روش GCNBPSO عمل کرده است. این مطلب در شکل ۱۲ با وضوح بیشتری مشاهده می‌شود. شکل ۱۲ نشان می‌دهد که الگوریتم GCNBPSO نیز مانند الگوریتم NBPSO دچار رکود و همگرایی زودرس می‌شود. این در حالی است که الگوریتم بهبود یافته پیشنهادی در محیط گسسته (INBPSO)، دچار رکود و همگرایی زودرس نشده و همواره به در راستای یافتن جواب بهینه در حال تکاپو است.

از سوی دیگر، نتایج ارائه شده در جدول ۳ بیان کننده آن است که مدل محلی GCNBPSO کارایی مناسبی ندارد و استفاده از آن نتنها موجبات افزایش کارایی الگوریتم NBPSO نمی‌شود، بلکه به مراتب جواب‌های ضعیف‌تری ارائه می‌دهد. اما مدل محلی INBPSO مانند مدل کلی آن در یافتن جواب بهینه موثر عمل می‌کند. در قیاس مدل‌های محلی و کلی الگوریتم INBPSO، ملاحظه می‌شود که هر دو مدل بسیار قوی و موثر عمل کرده و پاسخ‌هایی بسیار نزدیک بهم ارائه می‌کنند. در این بین بخاطر سادگی پیاده سازی مدل کلی، پیشنهاد می‌شود که از این مدل استفاده شود.

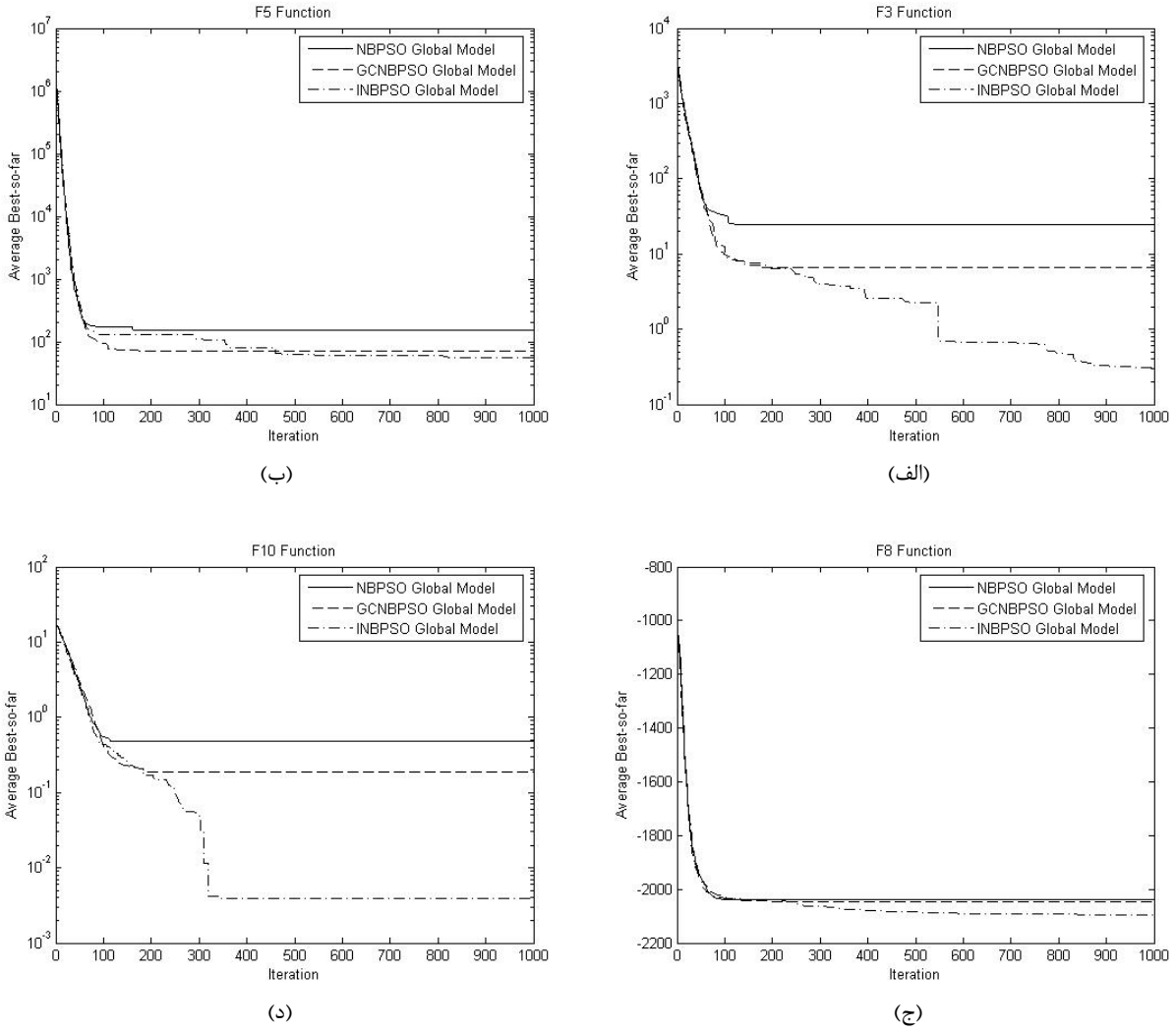
۶- جمع‌بندی

الگوریتم PSO یکی از الگوریتم‌های جستجوی ابتکاری است که کاربردهای وسیعی در زمینه‌های مختلف علوم به خصوص علوم مهندسی پیدا کرده است. این الگوریتم ماهیت پیوسته دارد و برای حل مسائل گسسته، نسخه باینری آن در سال ۱۹۹۷ معرفی شده است. نسخه باینری دو ضعف عمده ساختاری دارد که توانایی حل مساله و همگرایی الگوریتم را به مخاطره می‌اندازد. در این راستا در این مقاله ابتدا با برطرف کردن ضعفهای الگوریتم باینری متداول، الگوریتم باینری جدیدی ارائه شده است.

اگر چه در نسخه جدید، مشکلات اصلی الگوریتم باینری برطرف شده و گام‌هایی بزرگی در راستای بهبود آن برداشته شده است، اما متاسفانه به دلیل آنکه این

جدول۳- نتایج پیاده‌سازی الگوریتم باینری پیشنهادی (NBPSO)، الگوریتم باینری پیشنهادی بهبود یافته در محیط پیوسته (GCNBPSO) و الگوریتم باینری پیشنهادی بهبود یافته در محیط گسسته (INBPSO) روی توابع جدول۱ بهترین جواب دیده شده برای مرحله ۱۰۰۰ الگوریتم محاسبه شده و برای ۵۰ اجرای مستقل، میانگین (Ave.best) و میانه (Median) آن ارائه شده است

تابع	مدل کلی						مدل محلی					
	Ave.best			Median			Ave.best			Median		
	NBPSO	GCNBPSO	INBPSO	NBPSO	GCNBPSO	INBPSO	NBPSO	GCNBPSO	INBPSO	NBPSO	GCNBPSO	INBPSO
F_1	.	۰/۰۱۴۵	$۴/۶۵*۱۰^{-۵}$	۰/۰۰۳۵	$۴/۶۵*۱۰^{-۵}$	$۴/۶۵*۱۰^{-۵}$	۰/۳۶۱۴	۵۸/۹۳	$۴/۶۵*۱۰^{-۵}$	۰/۰۱۱۴	۴۴/۱۴	$۴/۶۵*۱۰^{-۵}$
F_2	۰/۰۱۷۴	۰/۰۰۳۹	۰/۰۰۱۵	۰/۰۰۸۲	۰/۰۰۲۱	۰/۰۰۱۵	۰/۰۲۳۱	۰/۹۹۴۲	۰/۰۰۱۵	۰/۰۰۷۶	۱/۰۱۹۱	۰/۰۰۱۵
F_3	۲۴/۹۴۳۵	۶/۵۸۱۷	۰/۳۱۱۷	۴/۹۹۹۰	۰/۳۰۷۹	۰/۰۰۰۱	۱۰/۶۸۳۰	۱۰۷/۶۴	۱/۶۷۲۳	۵/۲۰۵۰	۱۱۰/۵۶	۰/۱۹۶۶
F_4	۰/۲۸۸۵	۰/۱۵۶۸	۰/۰۰۶۵	۰/۱۹۸۳	۰/۰۳۳۵	۰/۰۰۳۰	۰/۶۸۹۳	۵/۶۵۲۵	۰/۰۰۳۴	۰/۴۱۸۱	۵/۴۹۳۳	۰/۰۰۳۰
F_5	۱۵۵/۸۰	۷۲/۲۸۲	۵۵/۹۲۴	۴/۵۰۸۴	۳/۹۵۲۶	۳/۹۹۵۸	۱۷/۵۷۷۰	۱۰۸۷/۸	۳/۸۱۵۱	۷/۹۸۸۰	۵۰۴/۹۲	۳/۷۷۱۵
F_6	۰/۶۸۱۹	۰/۶۰۸۱	۰/۵۷۲۱	۰/۵۹۷۰	۰/۵۶۶۳	۰/۵۰۶۵	۰/۷۳۳۵	۶۶/۸۵۱	۰/۲۹۴۴	۰/۶۵۴۱	۵۳/۹۷	۰/۲۶۵۹
F_7	۰/۳۹۴۷	۰/۳۸۳۹	۰/۴۱۱۴	۰/۳۹۴۰	۰/۳۷۷۳	۰/۴۲۲۱	۰/۳۹۹۰	۰/۷۵۲۳	۰/۴۰۲۱	۰/۴۰۳۷	۰/۷۵۱۹	۰/۴۱۱۲
F_8	-۲۰۴۰/۳	-۲۰۴۸/۰	-۲۰۹۴/۶	-۲۰۶۰/۲	-۲۰۶۴/۳	-۲۰۹۴/۷	-۲۰۹۴/۴	-۱۹۳۷/۰	-۲۰۹۴/۷	-۲۰۹۴/۸	-۲۰۹۴/۸	-۲۰۹۴/۸
F_9	۳/۶۹۲۰	۲/۷۷۳۵	۱/۰۳۴۳	۳/۲۸۸۰	۲/۲۳۵۸	۱/۲۳۷۵	۲/۱۳۵۶	۷/۵۳۶۲	۰/۸۲۲۱	۲/۲۶۱۰	۷/۳۴۳۶	۰/۹۹۴۹
F_{10}	۰/۴۸۲۱	۰/۱۸۶۴	۰/۰۰۳۹	۰/۰۷۱۶	۰/۰۰۶۴	۰/۰۰۳۹	۰/۵۷۳۴	۴/۳۶۵۲	۰/۰۰۳۹	۰/۱۶۲۵	۴/۹۰۹۷	۰/۰۰۳۹
F_{11}	۹۹/۳	۹۹/۶۶	۱۰۰	۹۹	۱۰۰	۱۰۰	۱۰۰	۸۹/۶۶	۱۰۰	۱۰۰	۹۰/۵	۱۰۰



شکل ۱۲- متوسط بهترین جواب دیده شده در بهینه‌سازی توابع الف) F_3 ب) F_5 ج) F_8 د) F_{10}

نتایج ارائه شده، در پیاده‌سازی مدل کلی بدست آمده‌اند. هر یک از الگوریتم‌ها ۵۰ مرتبه بطور مستقل اجرا شده و متوسط بهترین جواب دیده شده، ارائه شده است.

الگوریتم ریشه در نسخه پیوسته دارد، مشکلات موجود در الگوریتم پیوسته بـه آن سرایت می‌کند. یکی از مشکلات شناخته شده الگوریتم PSO پیوسته کـه در نـوع باینری هم رسوخ می‌کند، همگرایی الگوریتم به بهینه‌های محلی است. بـرای حـل این مشکل در الگوریتم باینری، دو رویکرد وجود دارد. نخست آنکه با برطرف کردن مشکلات الگوریتم پیوسته، معضلات الگوریتم باینری حل شود. دیگر آنکه مشکلات الگوریتم بطور مستقیم در محیط گسسته با ارائه راهکارهای سازنده برطرف شود.

در این مقاله ضمن تحلیل عوامل رکـود الگـوریتم بـاینری پیشـنهادی، از یـک طرف راه حل مناسبی برای برطرف کردن معضل همگرایی زودرس آن در سـاختار پیوسته و از طرف دیگـر راه حـل مناسـبی در سـاختار گسسـته ارائـه شـد. نتـایج آزمایش‌ها و مقایسه با الگـوریتم بـاینری متـداول، مناسـب بـودن سـاز و کارهـای پیشنهادی را تایید می‌کند.

نتایج بیانگر آن است که ساز و کارهای ارائه شده در برطـرف کـردن مشـکلات ساختاری الگوریتم باینری بسیار موثر و مفیـد بـوده اسـت. همچنـین آزمـایش‌هـا نشاندهنده آن است که روشهای پیشنهادی برای برطرف کـردن مشـکل همگرایـی زودرس الگوریتم باینری جدید، در محیط پیوسته و گسسته نیـز مـوثر واقـع شـده است. اگرچه در این بین روش بهبودیافته در محیط گسسته یـا بـه عبـارتی دیگـر الگوریتم INBPSO بسیار موثرتر است.

مراجع

[1] J. Kennedy, and R. C. Eberhart, "Particle swarm optimization," *Proceedings of IEEE International Conference on Neural Networks*, Vol. 4, pp. 1942-1948 1995.

[2] J. Kennedy, and R. C. Eberhart, "A Discrete Binary version of the particle swarm algorithm," *IEEE International Conference on Computational Cybernetics and Simulation*, Vol. 5, pp. 4104-4108, 1997.

[3] K. E. parsopoulos, and M. N. Vrahatis, "Recent approaches to global optimization problems through particle swarm optimization," *Natural Computing*, Vol. 1, pp. 235-306, 2002.

[۴] م. رستمی شهربابکی، و ح. نظام‌آبادی پور، "انتخـاب ویژگـی در طبقـه‌بنـدی معنایی تصاویر با استفاده از الگوریتم PSO،" *یازدهمین کنفـرانس سـالانه انجمـن مهندسی کامپیوتر ایران*، صفحات ۲۶۹-۲۷۵، ۱۳۸۴.

[5] H. A. Firip, and E. Goodman, "Swarmed feature selection," in *IEEE Proceedings of the 33rd Applied Imagery Pattern Recognition Workshop*, pp. 112-118, 2004.

[6] H. B. Liu, Y. Y. Tang, J. Meng, and Y. Jp, "Neural networks learning using vbest model particle swarm optimization," *Proceedings of the 3rd International Conference on Machine Learning and Cybernetics*, pp. 3157-3159, 2004.

[7] B. Al-kazemi, and C. K. Mohan, "Training feed forward neural networks using multi-phase particle swarm optimization," *Proceedings of the 9th International conference on Neural Information*, Vol. 5, pp. 2615-2619, 2002.

[8] D. Merwe, and A. Engelbrecht, "Data clustering using particle swarm optimization," *Proc. of IEEE Congress on Evolutionary Computation*, Vol. 1, pp. 215-220, 2003.

[9] T. R. Machado, and H. S. Lopes, "A hiybrid particle swarm optimization model for the traveling salesman problem," *Adaptive and Natural Computing Algorithms*, pp. 255-258, 2005.

[10] V. G. Gudise, and G. K. Venayagamoorthy, "FPGA placement and routing using particle swarm optimization," *Proc. of the IEEE Computer Society Annual Symposium on VLSI Emerging trends in VLSI Systems Design*, pp. 307-308, 2004.

[11] M. Nasri, H. Nezamabadi-pour, and M. M.Farsangi, "Design of a PID controller using PSO algorithm Incorporating fuzzy objective function," *6th Iranian Conference on Fuzzy Systems and 1st Islamic World Conference on Fuzzy Systems*, pp. 157-168, 2006.

[12] M. M.Farsangi, H. Nezamabadi-pour, and K. Y. Lee, "Multi-objective VAr planning with SVC for a large power system using PSO and GA," *IEEE Power Systems Conference & Exposition*, pp. 274-279, 2006.

[13] S. Mollazei, M. M. Farsangi, and H. Nezamabadi-pour, "Allocation of TCSC to enhance total transfer capability using guaranteed convergence particle swarm optimization," *3rd International Conference on Technical and physical Problems in Power Engineering*, pp. 5-9, 2006.

[14] M. Assadian, M. M.Farsangi, and H. Nezamabadi-pour, "Distribution network reconfiguration for loss reduction using particle swarm optimization," *3rd International Conference on Technical and physical Problems in Power Engineering*, pp. 15-19, , 2006.

[۱۵] م. رستمی شهربابکی، و ح. نظام‌آبادی‌پور، "روش جدیدی برای الگوریتم PSO باینری،" *چهاردهمین کنفرانس مهندسی برق ایران*، ۱۳۸۵.

[16] Yao X, Liu Y, and Lin G, "Evolutionary programming made faster," *IEEE Transaction on Evolutionary Computation*, Vol. 3, No. 2, pp. 82-102, 1999.

[17] F. V. D. Bergh, and A. Engelbrecht, "A new locally convergent particle swarm optimizer," *Proc. of IEEE Conference on System,Man andCybernetics*, Vol. 3, pp. 6, 2002.

[18] F. V. D. Bergh, *An analysis of particle swarm optimizers*, PhD Thesis, Department of Computer Science, University of Pretoria, South Africa, 2002.

[۱۹] ح. نظـام‌آبـادی‌پـور، و م. رسـتمی شـهربابکی، "تعمیمـی بـر الگـوریتم GCBPSO،" *دوازدهمین کنفرانس سالانه انجمن کامپیوتر ایران*، ۱۳۸۵.

حسین نظام‌آبادی‌پور کارشناسی و کارشناسی ارشد خود را در مهندسی برق- الکترونیک به ترتیب از دانشگاه شهید باهنر کرمان و دانشگاه تربیت مدرس در سال‌های ۱۳۷۷ و ۱۳۷۹ دریافت کرد. او سپس مدرک دوره دکترای خود را در مهندسی برق- الکترونیک از دانشگاه تربیت مـدرس در سال ۱۳۸۳ اخذ کرد و در همان سال در سمت استادیاری در بخش مهندسی برق دانشگاه شهید باهنر کرمان مشغول به فعالیت شد. وی در حال حاضر در سمت دانشیاری در این دانشگاه مشغول به خدمت است. زمینه‌های پژوهشی مورد علاقه او الگوریتم‌های ابتکاری، پردازش تصویر، بازشناسی الگو و رایانش نرم است

آدرس پست‌الکترونیکی ایشان عبارت است از:

nezam@mail.uk.ac.ir

مجید رستمی شهربابکی کارشناسی خود را در مهندسی برق- مخابرات در سال ۱۳۸۵ از دانشگاه شهید باهنر کرمان اخذ کرد. او سپس موفق به دریافت کارشناسی ارشد در رشته مهندسی برق- کنترل از همان دانشگاه در سال ۱۳۸۷ نائل شد. زمینه های پژوهشی مورد علاقه او الگوریتم های هوشمند، کنترل بهینه، کنترل فازی و کنترل و پایدارسازی سیستم‌های قدرت است.

آدرس پست‌الکترونیکی ایشان عبارت است از:

md_rostami@yahoo.com

ملیحه مغفوری فرسنگی کارشناسی خود را در مهندسی برق- کنترل از دانشگاه فردوسی در سال ۱۳۷۴ دریافت کرد. او سپس دوره دکترای خود را در مهندسی برق از دانشگاه برونل انگلستان درسال ۱۳۸۲ دریافت کرد و اکنون استادیار بخش مهـندسی بـرق دانشگـاه شهیـد باهنـر کرمـان است. زمینه‌های پژوهشی مورد علاقه وی کنترل و پایداری سیستم‌های قدرت، الگوریتم‌های ابتکاری و رایانش نرم است.

آدرس پست‌الکترونیکی ایشان عبارت است از:

mmaghfoori@mail.uk.ac.ir

---

[1] Particle Swarm Optimization
[2] Binary PSO (BPSO)
[3] Particle
[4] Global Model
[5] Local Model
[6] Unimodal
[7] Multimodal
[8] New BPSO (NBPSO)
[9] Mean Fitness
[10] Best-So-Far
[11] Median
[12] Guaranteed Convergence PSO (GCPSO)
[13] Guaranteed Convergence New Binary PSO (GCNBPSO)
[14] Improved NBPSO (INBPSO)