معرفی الگوریتم SSPCO با استفاده از چندین مدل شبیه‌سازی جهت مکانیزم بهینه‌سازی

روح اله امیدوار[۱]، حمید پروین[۲*]
[۱]عضو باشگاه پژوهشگران جوان و نخبگان، کارشناسی ارشد، واحد یاسوج، دانشگاه آزاد اسلامی، یاسوج، ایران، r.omidvar.uni@gmail.com
[۲]عضو هیات علمی، دکترای تخصصی، واحد نوراباد ممسنی، دانشگاه آزاد اسلامی، نورآباد ممسنی، ایران، parvin@iust.ac.ir

چکیده

بهینه‌سازی امروزه یکی از پرکاربردترین حوزه‌ها در علوم مختلف می‌باشد. الگوریتم‌های مختلفی برای حل این‌گونه مسایل ارائه شده است. شبیه‌سازی ابزاری است که دنیای واقعی را جهت اعمال الگوریتم بهینه‌سازی مهیا سازد. الگوریتم SSPCO یکی از جدیدترین الگوریتم‌های بهینه‌سازی می‌باشد. شبیه‌سازی الگوریتم‌ها نیز دارای چالش‌های بسیاری است که در ارائه الگوریتم به‌صورت موثر اهمیت دارد. متامدل‌ها رویکردی جهت مقابله با مشکل‌های شبیه‌سازی از جمله بار سنگین محاسبات و زمان اجرای بالای آن می‌باشد. متامدل خود دچار مشکلاتی نظیر این است که مدل شبیه‌سازی را از دنیای واقعی خود دور می‌کند. الگوریتم بهینه‌سازی SSPCO یکی از جدیدترین الگوریتم‌های فراابتکاری می‌باشد که از رفتار جوجه‌های پرنده تیهو الگوبرداری شده است. ما در این مقاله با استفاده از الگوریتم SSPCO سعی می‌کنیم ضمن حذف متامدل، زمان رسیدن به جواب بهینه را نیز کاهش دهیم. نتایج را در چهار روش اجرای الگوریتم آورده‌ایم که نتایج نشان می‌دهد که روش‌های جدید شبیه‌سازی توانسته با حذف متامدل زمان اجرا را کاهش دهد.

کلیدواژه

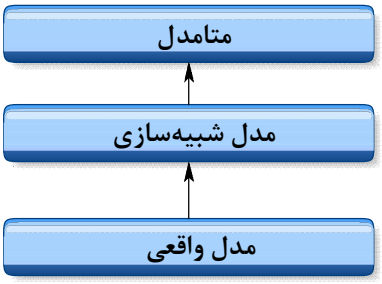
الگوریتم SSPCO، شبیه‌سازی، متامدل، هزینه، زمان اجرا.

مقدمه

بهینه‌سازی یک رویکرد ریاضی می‌باشد که معمولا منظور از بهینه‌سازی یک سامانه کمینه یا بیشینه کردن تابعی است که این تابع معیاری از عملکرد سامانه می‌باشد. این عمل در نهایت به بهبود کارایی سامانه می‌انجامد. در شرایطی که سایر روش‌های بهینه‌سازی قادر به در نظر گرفتن محدودیت‌های حاکم بر مسئله نباشند، بهینه‌سازی از طریق شبیه‌سازی می‌تواند راهگشا باشد. در مسائلی که طبیعت احتمالی دارند و روش‌های ریاضی‌وار همانند نظریه صف، قادر به مدل‌سازی مسئله نباشد، شبیه‌سازی به صورت وسیع استفاده می‌شود. در رویکرد بهینه‌سازی از طریق شبیه‌سازی، از شبیه‌سازی برای محاسبه مقدار تابع هدف مسئله و در عین حال در نظر گرفتن شرایط و محدودیت‌های مسئله استفاده می‌شود. متغیرهای تصمیم مسئله در قالب ورودی به شبیه‌سازی داده می‌شود و با اجرای فرایند شبیه‌سازی، مقدار تابع هدف مسئله به عنوان خروجی حاصل از شبیه‌سازی تعیین می‌شود. با توجه به طبیعت احتمالی مسئله، اجراهای مختلف شبیه‌سازی نتایج متفاوتی به دست می‌دهد و آنچه مدنظر ما است، میانگین این نتایج است. این مقدار میانگین، نتیجه انتظاری است که از ورودی مورد نظر ما حاصل می‌شود و با افزایش تکرار شبیه‌سازی‌ها با توجه به موضوع حد مرکزی به مقدار حدی خود همگرا می‌شود. در پژوهشی از یک سیستم بهینه‌سازی شبیه‌سازی برای سیستم‌های تولیدی، استفاده شده است [۱]. در یک تحقیق دیگر ترکیب شبیه‌سازی و شبکه عصبی برای بهینه‌سازی سیستم‌های صف پشت سر هم پیچیده استفاده شده است [۲]. در تحقیقی از شبیه‌سازی برای بهینه‌سازی مدیریت زنجیره تأمین استفاده شده است [۳]. در یک پژوهش دیگر از شبیه‌سازی برای بهینه‌سازی برنامه‌های بازرسی سیستم‌های تولیدی چندمرحله‌ای استفاده شده است [۴]. در تحقیقی برای مطالعه فرآیندها و تحلیل عملکرد سیستم در یک شرکت تأمین قطعات یدکی خودرو از شبیه‌سازی گسسته استفاده شده است [۵]. در پژوهشی برای تحلیل ریسک زمان پروژه‌ها از ترکیب شبیه‌سازی و مدل‌سازی سیستم‌های دینامیک استفاده شده است [۶]. مشکل اصلی در استفاده از رویکرد بهینه‌سازی از طریق شبیه‌سازی، زمانبر بودن پردازش محاسباتی لازم برای اجرای آن است. این موضوع از دو جهت ناشی می شود: زمانبر بودن اجرای هر فرآیند شبیه‌سازی و لزوم تکرارهای متعدد شبیه‌سازی به ازای هر ورودی مفروض. به

تاریخ دریافت: ۱۳۹۶/۰۵/۲۹ تاریخ پذیرش: ۱۳۹۷/۱۱/۲۰

همین منظور استفاده از متامدل‌ها در کنار رویکرد بهینه‌سازی از طریق شبیه‌سازی مرسوم است. متامدل فضای جواب پیچیده مسئله اصلی را به صورتی ساده‌تر مدل می‌کند. شبیه‌سازی، مسائل دنیای واقعی را در سطحی ساده‌تر و البته با تفاوت‌هایی مدل می‌کند. به عبارتی شبیه‌سازی قادر نیست دنیای واقعی را به همان صورتی که هست، مدل کند. در سطحی بالاتر، متامدل قرار دارد. متامدل، مدل شبیه‌سازی را بار دیگر مدل می‌کند؛ از پیچیدگی‌های آن می‌کاهد و در عین حال مدل را باز هم بیشتر از حالت واقعی دور می‌کند. این ارتباطات در شکل (۱) نمایش داده شده است.



شکل ۱. شماتیک ارتباط مدل‌های واقعی، شبیه‌سازی و متامدل

برای آنکه شناخت بهتری از متامدل‌ها به دست آورد می‌توان به منابع نوار رگرسیونی تطبیقی چند متغیره [۷-۱۱]، کریگینگ [۱۲-۱۴]، تابع پایه شعاعی [۱۵-۱۹]، شبکه عصبی مصنوعی [۲۰-۲۳] و همچنین ماشین بردار پشتیبان [۲۴-۲۹] مراجعه کرد. الگوریتم ارائه شده در این مقاله بر اساس الگوریتم بهینه‌سازی SSPCO است که توانایی جستجو در فضای جواب احتمالی را دارد. ساختار این مقاله به این ترتیب تنظیم گردیده است که بعد از مقدمه، بخش ۲ الگوریتم SSPCO معرفی شده است، سپس در بخش ۳، ۴، ۵ و ۶ الگوریتم را در رویکردهای متفاوت شبیه‌سازی کرده، در بخش ۷ توابع سنجشی را معرفی می‌کنیم. در بخش ۸ نتایج را خواهیم دید و در پایان در بخش ۹ به نتیجه‌گیری خواهیم پرداخت.

الگوریتم بهینه‌سازی SSPCO

ایده پایه ای این الگوریتم بهینه‌سازی از رفتار جوجه‌های یک نوع پرنده وحشی به نام تیهو گرفته شده است [۳۰]. جوجه‌های این نوع پرنده در هنگام احساس خطر برای رسیدن به یک نقطه امن در یک صف منظم قرار گرفته و درست در یک صف منظم پشت سر پرنده مادر شروع به حرکت می‌کنند تا به یک نقطه امن برسند. برای شبیه‌سازی رفتار جوجه‌های این پرنده در قالب یک الگوریتم بهینه‌سازی، هر جوجه به عنوان یک ذره از مسئله تحت بهینه در نظر گرفته می‌شود. حال هر ذره را باید طبق رفتار این نوع جوجه‌ها در یک صف منظم که می‌دانیم این صف ما را به بهترین نقطه بهینه می‌برد قرار دهیم و این به معنای کوچک کردن فضای جستجو نیست بلکه همگرا کردن ذرات پس از مدتی جستجو در قالب یک صف منظم به سمت بهترین نقطه جواب (پرنده مادر) می‌باشد. در زمان تکرارها و حرکت ذرات، ذرات را به سمت صفی رهنمون می‌سازیم که در این صف ذره مادر (پرنده مادر) در جلو صف به سمت جواب بهینه در حال حرکت می‌باشد. برای این کار با تنظیم یک معادله حرکت و سرعت برای ذرات، ذرات آنقدر در صف‌های گوناگون قرار می‌گیرند و از آن جدا می‌شوند تا به بهترین صفی که آن صف حتما به سمت جواب بهینه در حال حرکت است ملحق می‌شوند. در معادله سرعت و حرکت، هر ذره حرکت و سرعت خود را دقیقا و فقط بر اساس ذره‌ای که یک واحد از خود ذره بالاتر است تنظیم می‌کند. برای این کار یک متغیر برای هر ذره به نام متغیر اولویت در نظر می‌گیریم:

$$X_i.priority \quad (۱)$$

در هر بار ارزیابی وقتی ذره‌ای از بهترین تجربه شخصی یا بهینه محلی بهتر بود یک واحد به متغیر اولویت آن ذره اضافه می‌گردد:

$$if \quad X_i.cost > P_{best} \rightarrow X_i.P_{best.cost} = X_i.position \quad and \quad X_i.priority = X_i.priority + 1 \quad (۲)$$

P_{best} هزینه هر ذره در تابع بنچمارک می باشد، $X_i.cost$ بهترین تجربه شخصی هر ذره می باشد و $X_i.position$ مکان هر ذره است. در هر بار ارزیابی اگر بهینه محلی بهتر از بهینه عمومی باشد و بالعکس متغیر اولویت ذره بالاتر می‌رود و یک واحد به آن اضافه می‌گردد:

$$if \quad P_{best.cost} > G_{best.cost} \rightarrow G_{best.position} = P_{best.position} \quad and \quad X_i.priority = X_i.priority + 1 \quad (۳)$$

G_{best} بهینه عمومی است. معادله حرکت هر ذره تقریبا شبیه به الگوریتم توده ذرات به شکل فرمول ۴ تنظیم می‌شود:

$$X_i.position = X_i.position + X_i.velocity \quad (۴)$$

$X_i.velocity$ سرعت هر ذره یا جوجه می‌باشد. حال معادله سرعت ذره طبق فرمول ۵ محاسبه می‌شود:

$$X_i.velocity = w * X_i.velocity + c * rand(0,1) * [position(X_i.priority + 1)] - X_i.position \quad (۵)$$

$X_i.velocity$ سرعت ذره می‌باشد، w ضریب تاثیرگزاری سرعت قبل در معادله سرعت کنونی ذره می‌باشد، c فاکتور

اجتماعی می‌باشد که در الگوریتم توده ذرات نیز آمده است، $rand()$ یک عدد تصادفی در بازه ۰ و یک برای ایجاد حرکت تصادفی ذرات است، در ضمن معادله $[position(X_i.priority+1)]$ موقعیت مکانی ذره با اولویت یکی بالاتر از ذره کنونی می‌باشد که ذره کنونی سعی می‌کند سرعت خود را بر اساس این ذره تنظیم نماید، $X_i.position$ نیز مکان فعلی ذره می‌باشد. مشاهده می‌شود طبق معادله ۵ هر ذره دقیقا حرکت خود را بر اساس ذره‌ای با اولویت یکی بالاتر از خود تنظیم می‌کند بدین صورت که دیگر کاری به بهینه‌های محلی و عمومی ندارد و در هر لحظه از زمان ذرات فقط و فقط به دنبال ذره‌ای حرکت می‌کنند که در فضای جستجو از آن ذره یک واحد جلوتر باشد و به‌همین خاطر تعداد و زمان محاسبات در این الگوریتم نسبت به الگوریتم‌های گذشته بهینه‌سازی دارای سودمندی بالایی می‌باشد. ذرات طبق این معادله این قدر حرکت می‌کنند تا ذره‌ای را که به‌عنوان ذره مادر است به جواب بهینه هدایت کنند و بقیه ذرات پشت سر این ذره به سمت جواب بهینه حرکت کنند.

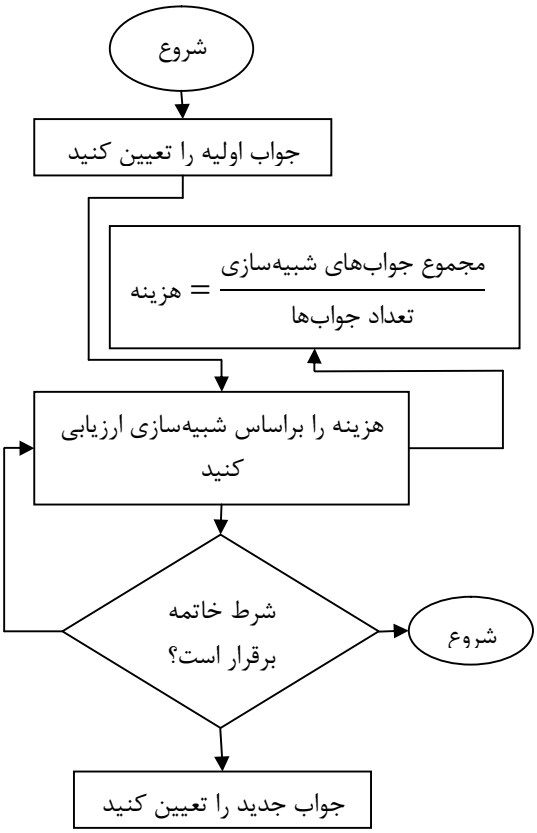
```
1.//initialize all chicken
2.Initialize
3.Repeat
4.  For each chicken i
5.    //update the chicken's best position and priority
6.    If f(x_i) > f(pb_i) then
7.    pb_i = x_i
8.    prioirity_i = prioirity_i + 1
9.    End if
10.   //update the global best position and priority
11.   If f(pb_i) > f(gb) then
12.       gb = pb_i
13.       prioirity_i = prioirity_i + 1
14.   End if
15. End for
16. //update chicken's velocity and position
17. For each chicken i
18.     For each dimension d
19.
v_{i,d} = v_{i,d} + C * Rand(0,1) * [position(priority_{i+1})] - x_{i,d}
20.   x_{i,d} = x_{i,d} + v_{i,d}
21.     End for
22. End for
23. //advance iteration
24. it = it + 1
25.Until it > MaxIterations
```

شکل ۲. شبه‌کد الگوریتم بهینه‌سازی SSPCO [۳۰]

در شکل ۲ که شبه‌کد الگوریتم SSPCO ارایه شده است، پس از جمعیت‌دهی اولیه در خطوط ۶ و ۷ تعیین بهینه محلی آمده است. در خط ۸ تاکید شده در صورت این که یک ذره در مکان جدید بهینه محلی خود شود، باید یک واحد به متغیر اولویت آن اضافه گردد، در خطوط ۱۱، ۱۲ و ۱۳ همین مورد برای یافتن بهینه عمومی تکرار شده است. در خط ۱۹ معادله سرعت و در خط ۲۰ معادله حرکت ذره برای رفتن به مکان جدید اعمال شده است. در خط ۲۴ با انجام یک سری از اعمال قبل یک واحد به تعداد تکرار الگوریتم اضافه شده و در خط ۲۵ شرط پایان تکرار تعیین‌شده برای الگوریتم آمده است. در الگوریتم SSPCO به دلیل حرکت سلسله‌مراتبی ذرات به دنبال یکدیگر با مشکل پوشش بیشتر فضای جستجو برای مسائلی که احتیاج به پوشش وسیع فضای مسئله دارد، روبرو خواهد بود. لذا ما با تقسیم جمعیت به چند زیرجمعیت سعی خواهیم کرد که همان مکانیزم الگوریتم SSPCO را در فضاهای مختلف از فضای جستجو انجام دهیم و سپس بهترین ذرات در هر زیرجمعیت را با هم در یک ارزیابی قرار دهیم.

رویکرد مستقیم

در رویکرد مستقیم، در مرحله تعیین مقادیر تابع هدف برای ذرات با توجه به فضای احتمالی مسئله، 20 تا 100بار شبیه‌سازی در مکان هر ذره انجام می‌گیرد و از نتایج آنها میانگین گرفته و به عنوان مقدار تابع هدف آن ذره ثبت می‌کنیم. با افزایش تعداد شبیه‌سازی‌های انجام گرفته در مکان هر ذره، با توجه به قانون حد مرکزی به مقدار واقعی تابع هدف در آن نقطه بیشتر نزدیک می‌شویم. شکل(۳) دیاگرام رویکرد بهینه‌سازی از طریق شبیه‌سازی را نمایش میدهد. در شکل (۴) دیاگرام روش میانگین‌گیری آمده است که در آن مشاهده می‌شود بعد از حرکت ذرات در فضای مسئله عمل میانگین‌گیری صورت پذیرفته است.

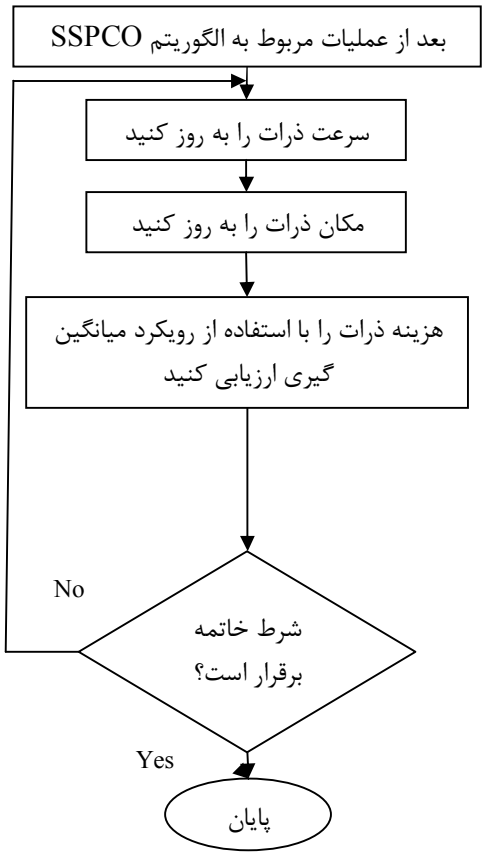


شکل ۳. دیاگرام روش رویکرد بهینه‌سازی از طریق شبیه‌سازی در حالت مستقیم

یکی از مشکلات استفاده از رویکرد مستقیم این است که اجرای شبیه‌سازی زمان‌بر است و یکی دیگر از مشکلات رویکرد مستقیم این است که نیاز به شبیه‌سازی‌های مکرری وجود دارد تا به خروجی واقعی نزدیک شویم، ما در این مقاله سعی بر حذف این مورد داریم.

رویکرد میانگین‌گیری

در این روش به جای شبیه‌سازیهای متوالی در یک نقطه برای رسیدن به مقدار تابع هدف، از میانگین‌گیری موضعی در آن منطقه استفاده میکنیم .به عبارتی هدف آن است که ذرات مقدار تابع خود را در فضای احتمالی با کمک از ذرات اطراف خود به دست آورند. برای همه ذرات یک بار شبیه‌سازی انجام می‌گیرد و نتایج ذخیره میشود. از آنجا که فقط یک مرتبه شبیه‌سازی انجام گرفته است، خطای جواب‌های به دست آمده زیاد خواهد بود. در مرحله بعد برای تصحیح این خطاها از رویکرد میانگین‌گیری استفاده میشود.



شکل ۴. دیاگرام رویکرد میانگین‌گیری

همسایه هر ذره را تعیین می‌کنیم و مقادیر حاصل از شبیه-سازی مربوط به ذرات همسایه میانگین گرفته و عدد حاصله به عنوان مقدار تابع هدف آن ذره در نظر گرفته می‌شود. از آنجا که مقدار تابع هدف در نقاط بسیار دور از نقطه مورد نظر در مقدار تابع هدف آن تأثیری ندارد و تغییرات وسیع تابع در این بازه بلند مانع از آن می‌شود که بتوان از آن استفاده کرد، باید محدوده‌های برای ذره تعریف شود که فقط ذرات در آن محدوده در میانگین‌گیری شرکت خواهند داشت. جواب جدید هزینه مکان جدیدیست که ذره با استفاده از معادله حرکت ذرات به آنجا منتقل شده است، می‌باشد. این رابطه را به شکل معادله ۶ بیان می‌شود.

$$\tilde{f_i} = \frac{1}{m}\sum \tilde{f_j}_{j=(|x_i - x_j| \le d_{max})} \qquad (۶)$$

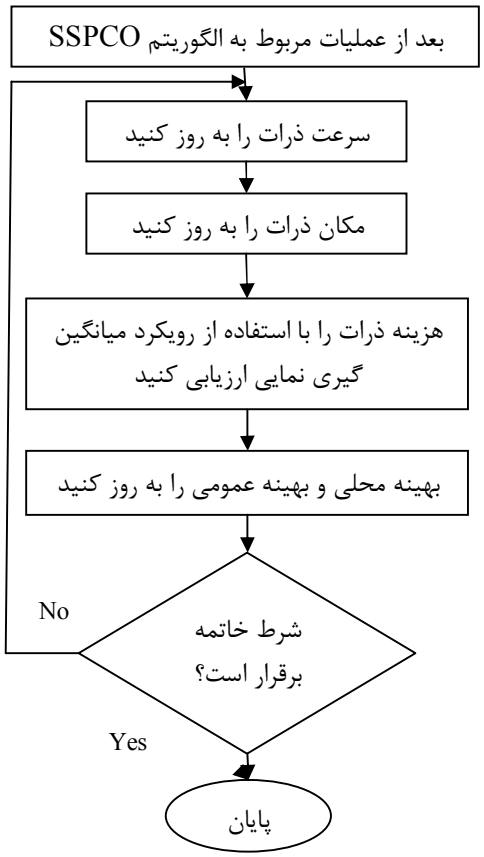
در این معادله d_{max} بیشترین فاصله‌ای است که ذرات در آن فاصله از ذره موردنظر در میانگین‌گیری برای حصول مقدار تابع هدف شرکت می‌کنند و m تعداد کل ذراتی است که شرط فاصله در آنها صدق می‌کند.

رویکرد میانگین‌گیری نمایی

در این روش هم مانند روش قبیل میانگین‌گیری جای شبیه-سازی مکرر را گرفته است. در موقعیت مکانی هر ذره یک بار شبیه‌سازی انجام گرفته و سپس معادله ۷ محاسبه می‌گردد.

$$f_i^- = \hat{f_{max}} - \frac{1}{n}\sum_{j=1}^{n}(\hat{f_{max}} - \hat{f_j})e^{-k|x_i - x_j|} \quad (۷)$$

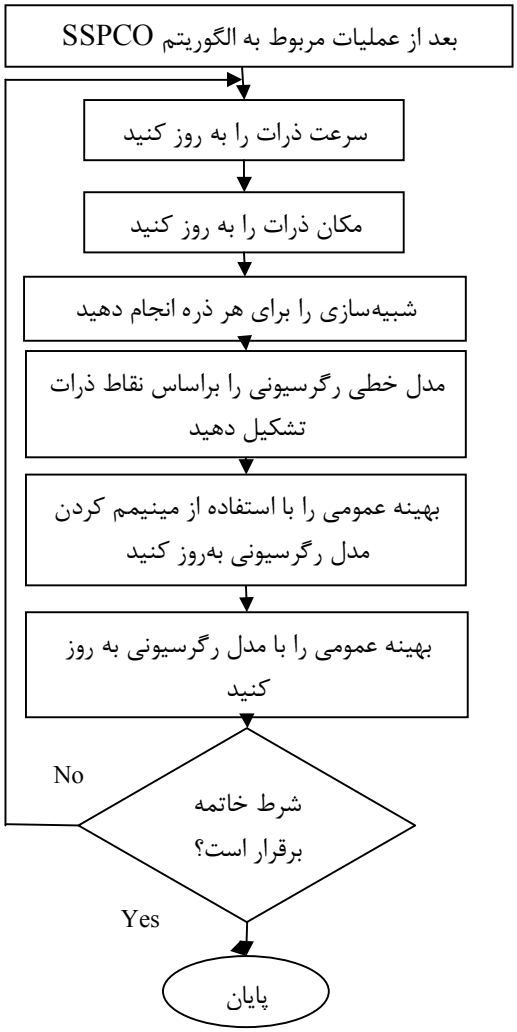
در این معادله $\hat{f_{max}}$ حداکثر مقدار تابع هدفی است که در شبیه‌سازی بدست آمده است و n تعداد ذرات می‌باشد، همچنین k ضریبی است که با توجه به مسئله مورد نظر تعیین می‌گردد. در اینجا به تعداد n نقاط جرمی روی یک صفحه قرار دارند. نقاطی که تابع هدف آنها کمتر باشد، جرم بیشتری نسبت به بقیه دارند. روشهای میانگین‌گیری در توابع هموار بهتر عمل می‌کنند. دیاگرام مربوط به شکل (۵) روش میانگین-گیری نمایی را نشان می‌دهد که در آن هزینه ذرات بعد از جابجایی ذرات در فضا ارزیابی شده و سپس بهینه‌های محلی و عمومی به روز می‌شوند.



شکل ۵. دیاگرام رویکرد میانگین‌گیری نمایی

شبیه‌سازی با متامدل دینامیک

روش کار در استفاده از متامدل رگرسیونی، تعیین پایه‌های فضا(تعیین نقاطی از فضا که از روی آنها مدل رگرسیونی ساخته می‌شود) و سپس به دست آوردن نقطه بهینه در فضای متامدل ایجاد شده است. در این رویکرد، ما پایه‌های فضا را روی ذرات الگوریتم جستجو پیاده می‌کنیم. در این رویکرد از نقطه بهینه متامدل به عنوان بهینه عمومی الگوریتم SSPCO استفاده می‌شود. در هر گام ذرات در نقاط جدید قرار گرفته، سپس متامدل به‌روز می‌شود. در شکل (۶) دیاگرام رویکرد متامدل دینامیک روی الگوریتم SSPCO را خواهیم دید.



شکل ۶. دیاگرام رویکرد متامدل دینامیک

توابع سنجشی

نتایج شبیه‌سازی را براساس توابع زیر انجام داده‌ایم.

F1: تابع سینوسی

$$min\, f(x) = x_1 \sin(x_2) + x_2 \sin(x_1) \quad (۸)$$
$$-2\pi \le x_1 \le 2\pi$$
$$-2\pi \le x_2 \le 2\pi$$

F2: تابع نامتقارن

$$min\, f(x) = \sum_{i=1}^{8} [2^{x_i - 4} + (6 - x_i)] \quad (۹)$$

برای هر یک از مسائل تعریف شده در بالا، چندین تابع توزیع نرمال تعریف شده است که نقش ماهیت احتمالی مسائل مطرح شده در رویکرد بهینه‌سازی از طریق شبیه‌سازی را بازی میکنند. این توابع درجدول (۱) آمده‌اند.

جدول ۱. توابع توزیع نرمال

		Error Functions	
		1.Small Error	2.Large Error
Problems	**F1**	η~N(0,1)	η~N(0,5)
	F2	η~N(0,7)	η~N(0,35)

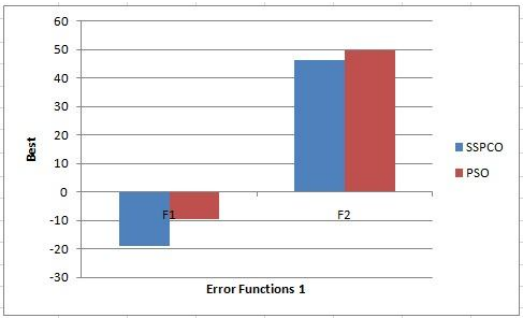
رویکرد مستقیم روی هر مسئله، با یکی از تابع خطای 20 بار پیاده میانگین جواب‌ها در ادامه ارائه شده است. تعداد تکرار شبیه‌سازی در هر نقطه مفروض 25 بار است که از میانگین آن به عنوان مقدار تابع هدف در آن نقطه استفاده شده است.

نتایج

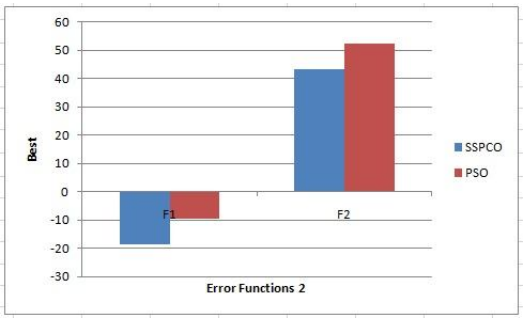
هر یک از رویکردهای ارائه شده را روی مسائل تعریف شده با خطاهای مختلف پیاده کرده‌ایم .این کار برای هر یک از حالات مورد بررسی 25 بار تکرار شده و نتایج در قالب میانگین و بهترین جواب حاصله ارائه شده است. توجه شود که در این رویکرد، برای محاسبه مقدار تابع هدف هر ذره، فقط یک بار شبیه‌سازی انجام میگیرد. در جدول ۲ نتایج حاصل از اجرای روش مستقیم روی مسائل آزمونی را داریم. نتایج را در سه بخش میانگین هزینه (Mean)، بهترین هزینه (Best) و زمان اجرای پردازنده (Cpu Time) و روی ۲ تابع سینوسی (F1) و تابع نامتقارن (F2) در ۲ بخش تابع خطای کوچک (Error Functions 1) و تابع خطای بزرگ (Error Functions 2) آورده‌ایم. همه موارد گفته‌شده با پیاده‌سازی مدل‌های مدکور با الگوریتم بهینه‌سازی توده ذرات نیز مقایسه شده است تا میزان اتکا به الگوریتم SSPCOنیز مشخص گردد.

جدول ۲. نتایج استفاده از روش مستقیم با ۲۵ بار تکرار

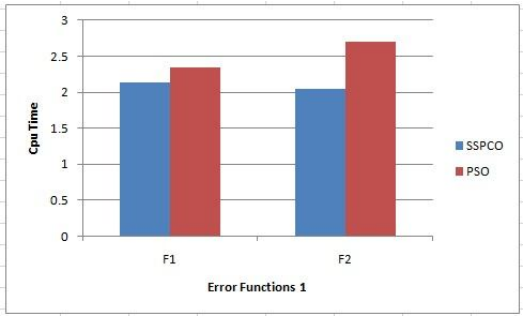
		Error Functions 1		
		Mean	Best	Cpu Time
F1	SSPCO	-18.29	-18.93	2.13
F2		42.32	46.18	2.05
F1	PSO	-9.59	-9.62	2.34
F2		46.74	49.55	2.70
		Error Functions 2		
		Mean	**Best**	**Cpu Time**
F1	SSPCO	-19.35	-18.78	1.98
F2		49.81	43.31	2.19
F1	PSO	-9.41	-9.62	2.36
F2		51.48	52.40	2.82



شکل ۶. بهترین هزینه برای روش مستقیم و تابع خطای کوچک



شکل ۷. بهترین هزینه برای روش مستقیم و تابع خطای بزرگ



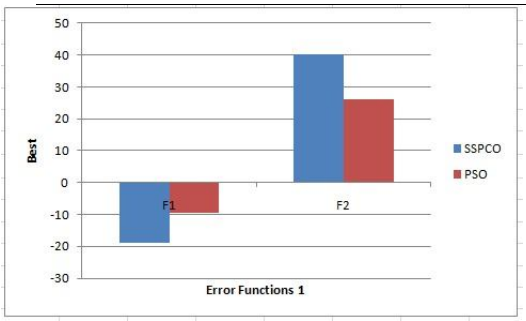
شکل ۸. زمان اجرا برای روش مستقیم و تابع خطای کوچک

شکل ۹. زمان اجرا برای روش مستقیم و تابع خطای کوچک

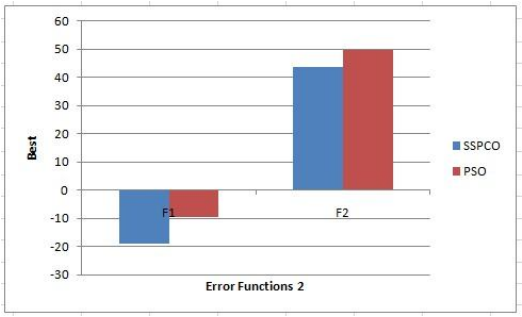
نتایج روش مستقیم در الگوریتم SSPCO نشان می‌دهد کیفیت نتایج در مسایل Small Error(**F1**) بهتر بوده ولی در مسائل Large Error (**F2**) زمان اجرای کمتری داشته‌ایم.

جدول ۳. نتایج استفاده از رویکرد میانگین‌گیری

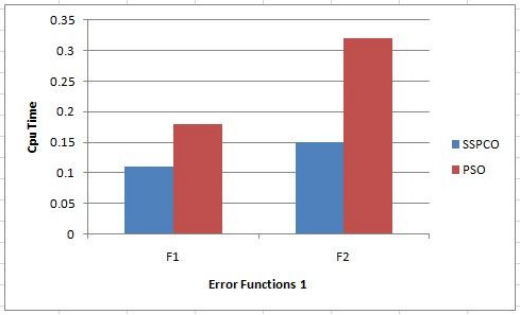
		Error Functions 1		
		Mean	Best	Cpu Time
F1	SSPCO	-16.82	-18.74	0.11
F2		44.51	40.21	0.15
F1	PSO	-8.41	-9.57	0.18
F2		32.07	26.23	0.32
		Error Functions 2		
		Mean	Best	Cpu Time
F1	SSPCO	-14.29	-18.94	0.10
F2		40.21	43.65	0.24
F1	PSO	-6.86	-9.61	0.14
F2		49.66	49.72	0.32



شکل ۱۰. بهترین هزینه روش میانگین‌گیری و تابع خطای کوچک



شکل ۱۱. بهترین هزینه روش میانگین‌گیری و تابع خطای بزرگ



شکل ۱۲. زمان اجرا روش میانگین‌گیری و تابع خطای کوچک

شکل ۱۳. زمان اجرا روش میانگین‌گیری و تابع خطای کوچک

نتایج در رویکرد میانگین‌گیری نشان می‌دهد کیفیت نتایج نسبت به رویکرد مستقیم پایین‌تر ولی زمان اجرای بهتری را داشته است. در جدول ۴ نتایج در روش میانگین‌گیری نمایی آمده است که نشان می‌دهد در تابع ۲ نسبت به روش میانگین‌گیری نتایج کیفیت بهتری دارد ولی نسبت به روش مستقیم همچنان کیفیت نتایج پایین‌تر است.

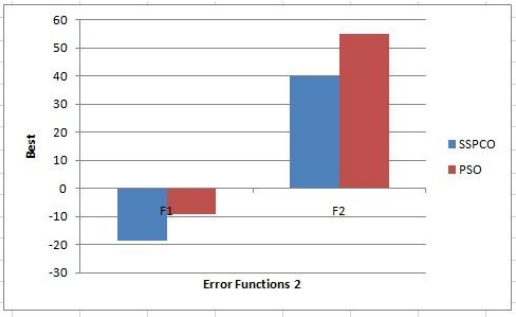
جدول ۴. نتایج در روش میانگین‌گیری نمایی

		Error Functions 1		
		Mean	Best	Cpu Time
F1	SSPCO	-16.28	-18.41	0.09
F2		43.28	46.15	0.10
F1	PSO	-8.41	-9.57	0.18
F2		32.07	26.23	0.32
		Error Functions 2		
		Mean	Best	Cpu Time
F1	SSPCO	-14.05	-18.48	0.11
F2		40.27	40.09	0.21
F1	PSO	-8.65	-9.06	0.17
F2		41.48	55.17	0.26

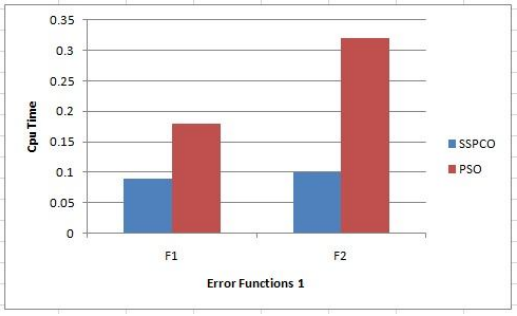
جدول ۵. نتایج در روش متامدل دینامیک

		Error Functions 1		
		Mean	Best	Cpu Time
F1	SSPCO	-18.65	-18.49	0.11
F2		40.35	35.64	0.22
F1	PSO	-9.45	-9.60	0.21
F2		35.23	27.68	0.31
		Error Functions 2		
		Mean	**Best**	**Cpu Time**
F1	SSPCO	-18.55	-18.49	0.15
F2		40.47	40.56	0.23
F1	PSO	-9.12	-9.45	0.23
F2		46.13	56.01	0.33

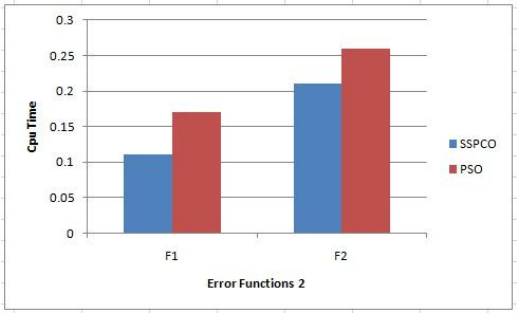
شکل ۱۴. بهترین هزینه روش میانگین‌گیری نمایی و تابع خطای کوچک



شکل ۱۵. بهترین هزینه روش میانگین‌گیری نمایی و تابع خطای بزرگ

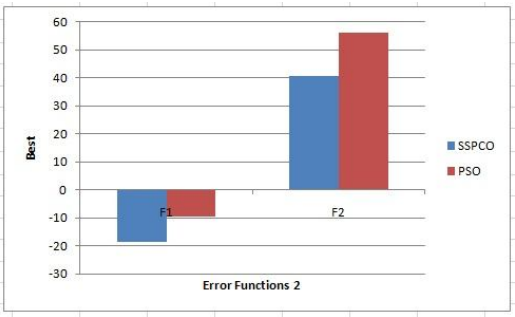


شکل ۱۶. زمان اجرا روش میانگین‌گیری نمایی و تابع خطای کوچک



شکل ۱۷. زمان اجرا روش میانگین‌گیری نمایی و تابع خطای کوچک

شکل ۱۸. بهترین هزینه روش میانگین‌گیری و تابع خطای کوچک



شکل ۱۹. بهترین هزینه روش دینامیک و تابع خطای بزرگ

شکل ۲۰. زمان اجرا روش دینامیک و تابع خطای کوچک

شکل ۲۱. زمان اجرا روش دینامیک و تابع خطای کوچک

نتایج در روش متامدل نسبت به روش مستقیم توانسته در زمان اجرای کمتری انجام گردد، گرچه نسبت به روش مستقیم کیفیت نتایج پایین‌تر است و این کیفیت نتایج در روش متامدل نسبت به دو روش میانگین‌گیری و میانگین‌گیری میانی پایین‌تر است. زمان اجرای بهتری نسبت به روش میانگین‌گیری داشته است. کیفیت نتایج الگوریتم SSPCO در همه موارد به جز در تابع F2 روش متامدل دینامیک نسبت به الگوریتم سنتی PSO بهتر بوده و از مقدار زمان کمتری نیز برای شبیه‌سازی برخوردار بوده است.

نتیجه‌گیری

در این مقاله الگوریتم SSPCO که از رفتار جوجه‌های پرنده تیهو در هنگام خطر الگوبرداری شده است را شبیه‌سازی گردید. این الگوریتم در این مقاله طوری پیاده‌سازی گردید که با حدف متامدل‌های کلاسیک زمان دستیابی به جواب‌های بهینه را کاهش دهد. در شبیه‌سازی این الگوریتم سه روش میانگین-گیری، روش میانگین‌گیری نمایی و متامدل دینامیک ارائه شد و برای بررسی کارآیی آنها، دو مسئله بهینه‌سازی مطرح و روشها روی آنها پیاده شدند. اگرچه کیفیت نتایج به دست آمده از روشهای ارائه شده از کیفیت جواب‌های حاصل از روش مستقیم کمتر بود، ولی زمان لازم برای رسیدن به جواب‌ها به مراتب کمتر بود که این موضوع، استفاده از این روش‌ها را توجیه می‌کند. در حقیقت هدف اصلی که شبیه‌سازی الگوریتم SSPCO با روش‌های فوق برای حذف متامدل دنبال می‌شد، محقق گردید. برای کارهای آینده می‌توان نسبت به اعمال مکانیزم‌های یادگیری ذرات در الگوریتم SSPCO متمرکز شد.

مراجع

1. Y. Guo, W. Liao, X. Cheng, L. Liu, "SimOpt: A new simulation optimization system based virtual simulation for manufacturing system." *Simulation Modeling Practice and Theory*, Vol. 14, No. 5, PP. 577- 85, 2006.
2. A. Azadeh, Z. Faiz, S.M. Asadzadehand, R. Tavakkoli, "Simulation optimization using particle swarm optimization algorithm with application to assembly line design." *Applied Soft Computing*, Vol 11, No. 1, PP 605-613, 2011.
3. X. Wan, J F. Pekny, G V. Reklaitis, "Simulation-based optimization with surrogate models application to supply chin management" *Computers and Chemical Engineering*, Vol. 29, No. 6, PP. 1317- 1328, 2005.
4. A. Vaghefi, V. Sarhangian, "Contribution of simulation to the optimization of inspection plans for multi-stage manufacturing systems." *Computers & Industrial Engineerring*, Vol. 57, No. 4, PP. 1226-1234, 2009.
5. M R. Memar Jafari, Z S. Gatmiry, M. Khakzar Bafrouei, " A Discrete Simulation Analysis of Customer Order Supply System: A Case Study (Technical note)." *Journal of Industrial Engineering*, Vol. 45, No.1, PP. 95-102, 2001.
6. E. Hassan Nayebi, B. Kiyani, " Combination of Monte Carlo Simulation and System Dynamics Modeling for Project Time Risk Analysis." *Journal of Industrial Engineering*, Vol. 44, No.2, PP. 169-180, 2010.
7. J.H. Friedman, "Multivariate adaptive regression splines." *Annals Statistics*, PP. 1-67, 1991.
8. V. Pilla, J M. Rosenberger, V. Chen, N. Engsuwan,S. Siddappa, "A multivariate adaptive regression splines cutting plane approach for solving a two-stage stochastic programming fleet assignment model." *European Journalof Operational Research*, Vol. 216, No. 1, PP. 162-171, 2002.
9. F. Vidoli, "Evaluating the water sector in Italy through a two stage method using the conditional robust nonparametric frontier and multivariate adaptive regression splines." *European Journalof Operational Research*, Vol 212, No.3, PP. 583-595, 2011.
10. M. Durmaz, M O. Karslioglu, "Non-parametric regionl VTEC modeling with Multivariate Adaptive Regression B-Splines." *Advance in*

space research, Vol. 48, No. 9, PP. 1523-1530, 2011.

11. J. De Andrés, P. Lorca, F J.de Cos Juez, F. Sánchez-Lasheras, " Bankruptcy forecasting: A hybrid approach Fuzzy c-means clustring and Multivariate Adaptive Regression Spline (MARS)" *Expert Systems with Applications*, Vol. 38, No. 3, PP. 1866-1875, 2011.

12. W.C. Van Beers, J P. Kleijnen, "Kriging for interpolation in random simulation." *Journal of the Operational Research Society*, Vol. 54, No. 3, PP. 255-262, 2003.

13. J P. Kleijnen, W C. Van Beers, "Application-driven sequential designs for simulation experiments: Kriging metamodeling." *Journal of the Operationa Research Society*, Vol. 55, No. 8, PP. 876- 883, 2004.

14. J P. Kleijnen, W C. Van Beers, "Robustness of Kriging when interpolating in random simulation with heterogeneous variances: Some experiments." *European Journal of Operational Research*, Vol. 165, No. 3, PP. 826-834, 2005.

15. G. Meghabghab, A. Kandel, "Stochastic simulations of Web search engines: RBF versus second-order regression models." *Information Sciences*, Vol. 159, No. 1, PP. 1-28, 2004.

16. R G. Regis, C A. Shoemaker, "Parallel radial basis function methods for the global optimization of expensive functions" *European Journal of Operational Research*., Vol. 182, No. 2, PP. 514-535, 2007..

17. R G. Regis, "Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objeective and cconstraint functions." *Computers & Operation Research*. Vol. 38, No. 5, PP. 837-853, 2011.

18. Z. Luo, L. Tong, Z. Kang, "A level set method for structural shape and topology optimization uses radial basis functions " *Computers and structures*.,Vol. 87, No. 7, PP. 425-434, 2009.

19. M. Korürek, B. Doğan, "ECG beat classification using particle swarm optimization and radial basis function neural network." *Expert system with Application*.,Vol. 37, No. 12, PP. 7563-7569, 2010.

20. D J. Fonseca, D. Navaresse, G P. Moynihan, "Simulation metamodeling through artificial neural networks." *Engineering Applications of Artificial Intelligence*, Vol. 16, No. 3, PP. 177-183, 2003.

21. D J. Fonseca, D. Navaresse, "Artificial neural networks for job shop simulation." *Advanced Engineering Informatic.*, Vol. 16, No. 4, PP. 241-246, 2007.

22. M S. Mirtalaie, M A. Azadeh, M. Saberi, A. Ashjari, " A Trust-based Credit Scoring Model Using Neural Network." *Journal of Endustrial Engineering*, Vol. 46, No. 1, PP. 91-104, 2012.

23. M. Khashei, M. Bijari, " Gold price forecasting using hybrid artificial neural networks with fuzzy regression model." *Journal of Endustrial Engineering*, Vol. 44, No. 1, 2012.

24. S M. Clarke, J H. Griebsch, T W. Simpson, "Analysis of support vector regression for approximation of complex engineering analyses." *Journal of Mechanical Design*, Vol. 127, No. 6,PP. 1077- 1078, 2005.

25. T. Eitrich, B. Lang, "Efficient optimization of support vector machine learning parameters for unbalanced datadets." *Journal of Computational and Applied Mathematics*., Vol. 196, No. 2, PP. 425-436, 2005.

26. A. Haldun, S. Serpil, "Using support vector machine to learn the efficient set in multiple objective discrete optimization." *European journal of operation research*, Vol. 193, No.2 ,PP. 510-519, 2009.

27. U. Norinder, "Support vector machine in drug design: application to drug transport processes and QSAR using simplex optimisations and variables selection." *Neurocomputing*, Vol. 55, No. 1, PP. 337-346, 2003.

28. I. Aydin, M. karakose, E. Akin, "A multi-objective artificial immune algorithm for parameter optimization in support vector machine." *Applied soft computing*, Vol. 11, No. 1, PP. 120-129, 2011.

29. R M. Balabin, E I. Lomakina, "Support vector machine regression (SVR/LS-SVM)—an alternative to neural networks (ANN) for analytical chemistry? Comparison of nonlinear methods on near infrared (NIR) spectroscopy data." Analyst, vol. 136, No. 8, 1703-1712, 2011.

30. R. Omidvar, H. Parvin, F. Rad, SSPCO Optimization Algorithm (See-See Partridge Chicks Optimization). In: Fourteenth Mexican International Conference on Artificial Intelligence (MICAI); 25-31 October: Cuernavaca, Mexico: IEEE. pp. 101-106, 2015.