تمرین سوم (پروژه ی دوم) درس رایانش تکاملی

نام : حسین سیم چی

۹۸۴۴۳۱۱۹

نام استاد : آقای دکتر حامد ملک

۱۳۹۹/۱۰/۳۰

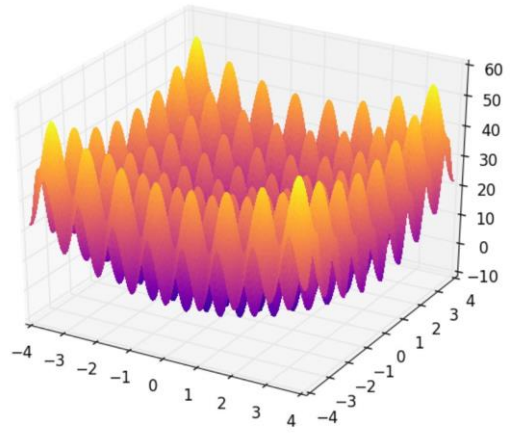
مقدمه

امروزه استفاده های بسیار زیادی از الگوریتم های تکاملی برای بهینه سازی توابع استفاده می شود. انواع الگوریتم های استفاده شده برای این منظور عبارت است از الگوریتم Evolution strategies، PSO و Differential evolution. در این تمرین هدف بررسی سه الگوریتم نام برده شده جهت پیدا کردن مقدار بهنیه ی سراسری توابع فوق است. همانطور که از شکل های این توابع مشخص است، این توابع دارای تعداد بسیار زیادی نقاط بهنیه محلی هستند. نقاط بهنیه ی محلی نقاطی هستند که در قسمتی از فضای جستجو، تابع دارای کمترین مقدار است ولی در کل فضای جستجو لزوما کمترین مقدار ممکن نیستند. یکی از مشکلات موجود در الگوریتم های ارائه شده در سالیان اخیر گرفتار شدن الگوریتم ها در نقاط بهینه محلی است که با استفاده از تکنیک های مختلف سعی می شود که از این نقاط فرار کرد و به بهینه ی سراسری همگرا شد.

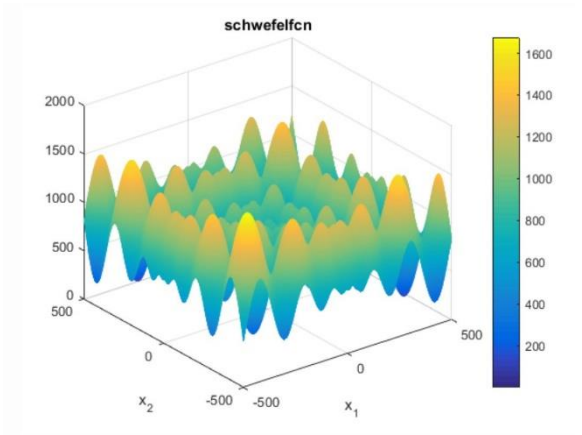
عملگرها و مراحل مختلفی در الگوریتم های تکاملی وجود دارند که باعث فرار از نقاط بهینه ی محلی و جستجو در تمام فضای مسئله می شوند. در بهینه سازی توابع، فضای جستجو شامل مقادیر تابع به ازای مقادیر مختلف x برای تابع است که برای بعدهای بالاتر این x می تواند به صورت برداری بیان شود. در این تمرین ما x را به ازای ابعاد ۲،۴ و ۲۰ بررسی کرده ایم و خواهیم دید که اثر افزایش ابعاد در هر الگوریتم به ازای پارامترهای مختلف چه خواهد بود. پس هدف از تمرین بیان شده، مقایسه ی سه الگوریتم بیان شده در پیدا کردن نقطه ی بهنیه ی سراسری است که خواهیم دید سرعت و دقت هر کدام از آن ها چه تفاوت هایی برای ابعاد کم و زیاد خواهد داشت.

توابع مورد استفاده

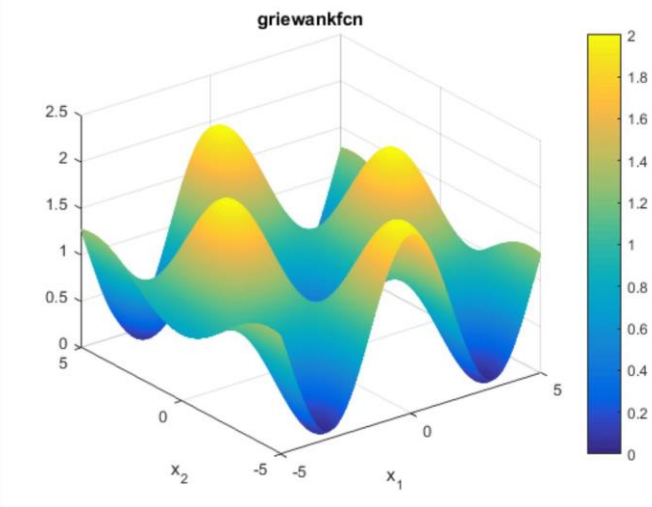
همانطور که در تمرین ذکر شده است در این تمرین از سه تابع بسیار معروف استفاده شده است که هر کدام از آن ها دارای نقاط بهینه ی محلی بسیار زیادی هستند و هدف ما بررسی کل فضای جستجو به ازای x های مختلف خواهد بود به طوری که در انتها بعد از انجام Iteration های مختلف دوست داریم تا مجموعه ی جواب خوبی داشته باشیم و خوب بودن بدین معنا است که جواب های ارائه شده به مقدار مینیمم سراسری نزدیک باشند. پس به طور کلی دوست داریم مختصات نقاطی را بیابیم که در این مختصات تابع مورد نظر بعد از تعداد تکرار معین دارای کمترین مقدار ممکن (نزدیک به مینیمم سراسری) باشد.



شکل ۱) تابع **Rastrigin** (در دو بعد)



شکل ۲) تابع **Schwefel** (در دو بعد)



شکل ۳) تابع **Griewank** (در دو بعد)

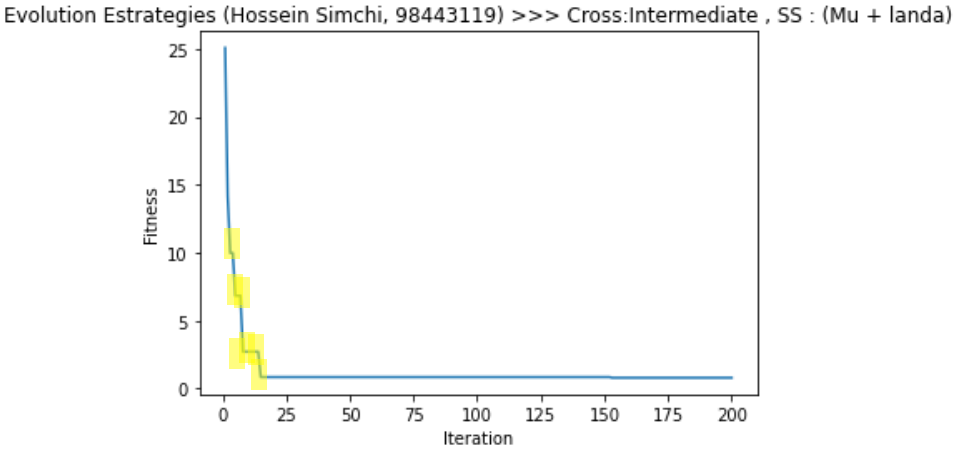
در ابتدا هر سه الگوریتم خواسته شده را بر روی تابع Rastrigin بررسی می کنیم و در انتها الگوریتم منتخب (به لحاظ دقت و سرعت) را بر روی دو الگوریتم دیگر امتحان می کنیم.

Evolution Esterategies (ES)

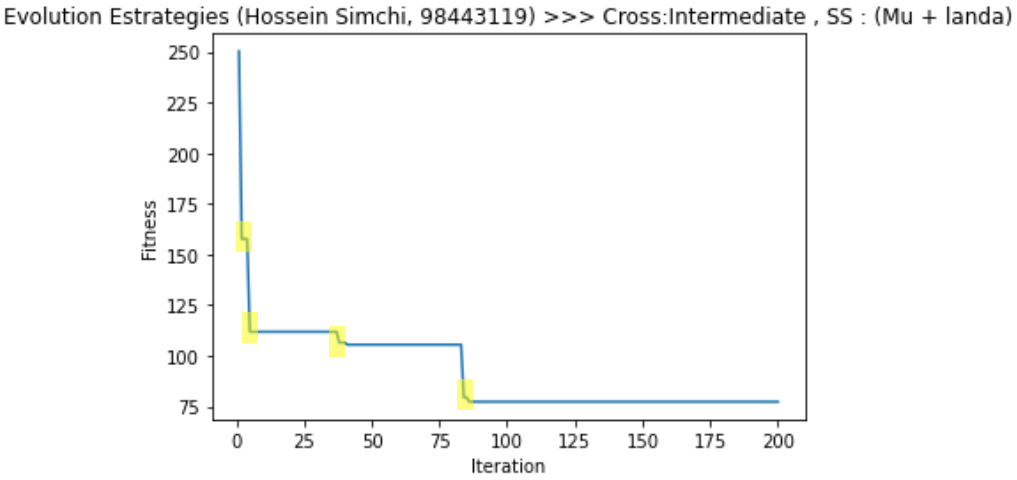
از الگوریتم ES برای بهینه سازی توابع عددی استفاده می شود و از مزایای آن همانطور که در ادامه خواهیم دید همگرایی سریع به نقطه ی بهینه سراسری است و به خاطر استفاده از جهش Gaussian خود را با شرایط جدید را تطبیق می دهد. ۴ نوع مختلف از الگوریتم ES را در این قسمت بررسی می کنیم که در ادامه به جزئیات آن ها می پردازیم.

1. ES : Recombination : Intermediatory, Survivor selection : (mu + landa)

برای این روش کافی است درابتدا جمعیت اولیه را تولید کنیم، سپس با استفاده از نوع Recombination بر روی جمعیت اولیه شروع به تولید جمعیت اولیه کنیم. همانطور که از نوع آن مشخص است باید دوتا والد انتخاب کنیم که برای انتخاب والد بر اساس Unifrom عمل می کنیم که محبوبیت بیشتری در این الگوریتم دارد و بدین صورت عمل می کند که همه ی والدین دارای شانس یکسان برای انتخاب دارند. پس از انتخاب دو والد برای تولید فرزند به این صورت عمل میکنیم که برای ژن i ام فرزند، میانگین ژن i ام دو والد را محاسبه می کنیم و جایگذاری می کنیم (Intermediatory).
در مرحله ی بعدی عملگر جهش را بر روی نسل موجود انجام می دهیم و بدین صورت که بر اساس قانون یک پنجم موفقیت عمل میکنیم یعنی در ابتدا سیگمایی را درنظر می گیریم و سپس در نسل بعد خواهیم دید که آیا این سیگما در بیشتر از یک پنجم از نسل ما باعث موفقیت بوده است یا خیر؟!، موفقیت بدین معنا که آیا از این نسل به نسل بعد فیتنس افزایش یافته است یا خیر. در این صورت مقدار سیگما را زیاد می کنیم تا فضای بیشتری را جستجو کنیم و درغیر این صورت این مقدار را کاهش می دهیم. در نهایت جمعیت اولیه و فرزندان انتخاب شده را ادغام می کنیم و از بین آن ها به اندازه ی جمعیت اولیه سلکشن را انجام می دهیم. که این روش بیانگر نوع سلکشن بیان شده در انتخاب بازماندگان است. **در هرنسل مقدار فیتنس که در این تمرین برابر با مینیمم تابع استفاده شده است را برمی گردانیم**. درنتیجه توقع داریم در تعداد مراحل معین در اجرای این الگوریتم به مینیمم سراسری دست پیدا کنیم. در زیر مقدار تابع فیتنس و دقت را پس از گذشت ۲۰۰ مرحله مشاهده می کنیم.



همانطور که در شکل فوق قابل برداشت است در ابتدا به ازای ۴ بعد، مقدار تغییرات تابع فیتنس رسم شده است. در جاهایی از شکل فوق که خط افقی داریم (هایلایت شده) و عملا تغییری در تابع فیتنس نداشته ایم در بهینه ی محلی گرفتار شده ایم. *ولی نکته ی مثبت این الگوریتم همگرایی بسیار سریع به مینیمم سراسری است که تقریبا در همان حدود ۲۰ iteration اول به مقدار مینیمم نزدیک صفر رسیده است.* البته ذکر این نکته نیز ضروری است که **مکان جمعیت اولیه بسیار در همگرایی موثر است** ولی چیزی که استنباط می شود و به صورت تئوری نیز ثابت شده است، همگرایی به نقطه ی مینیمم سراسری در الگوریتم ES بسیار سریع اتفاق می افتد. حال اگر به ازای ۲۰ بعد نیز مقدار تابع فیتنس را بدست آوریم خواهیم داشت :



تفاوت حالت ۲۰ بعد با ۴ در این است که به دلیل افزایش تعداد ابعاد ممکن است در تعداد مراحل بیشتری به بهینه ی سراسری همگرا شویم (تعداد مراحل بیشتری نیاز داشته باشیم). ولی همچنان سرعت قابل قبولی نسبت به الگوریتم های دیگر دارد.

توجه: نمودارهای رسم شده به ازای کمترین مقدار تابع در هر نسل رسم شده اند و هدف این بوده است که بتوانیم سرعت و دقت نهایی الکوریتم ها را باهم مقایسه کنیم.

نتیجه گیری

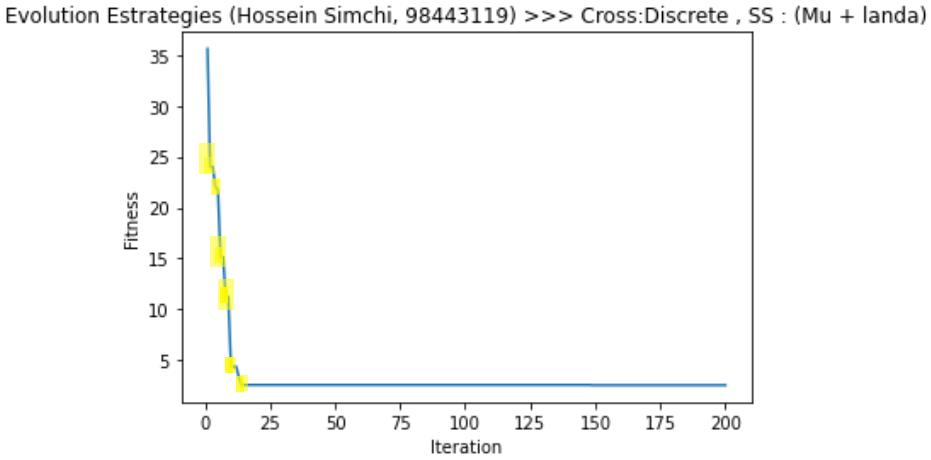
- ✓ سرعت الگوریتم ES در این حالت و انتخاب این نوع پارامترها و عملگرها بسیار خوب بوده و در ابعاد بالا نیز سرعت بسیار خوبی در همگرایی به بهینه سراسری دارند.
- ✓ عملگر انتخاب شده برای انتخاب بازماندگان به خاطر اینکه محیط م سئله پویا نی ست و تابع فیتنس در هر مرحله تغییر نمی کند بسیار خوب عمل می کند.

2. ES : Recombination : Intermediatory, Survivor selection : (landa)

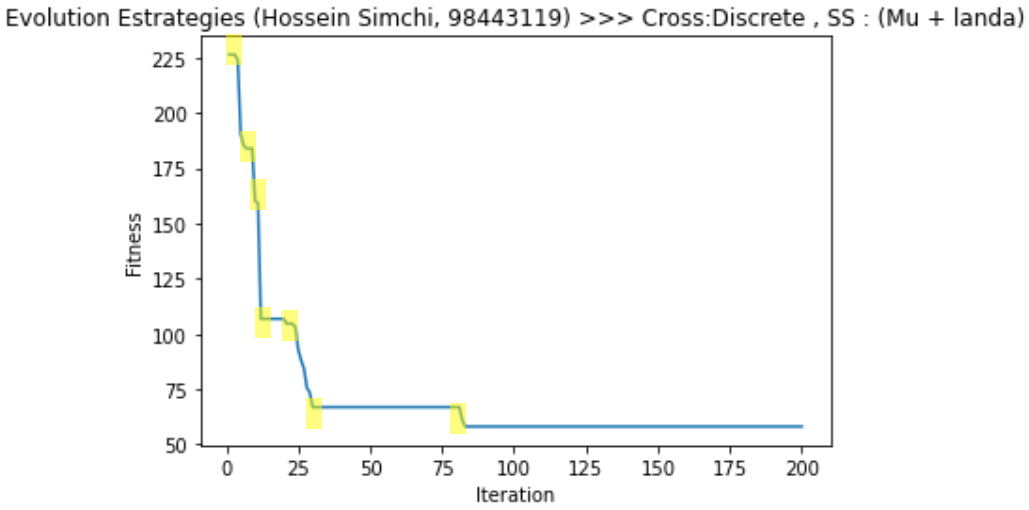
تفاوت این روش با روش قبل در انتخاب نوع بازماندکان است بدین صورت که انتخاب فقط بر اساس نسل فعلی می باشد. یعنی به جای اینکه از بین نسل قبل و نسل فعلی ۲۰ تا از بهترین ها را انتخاب کنیم، فقط از نسل فعلی استفاده می کنیم و نسل قبلی را کنار می گذاریم. **این نوع انتخاب بازماندگان برای مسائلی که تابع هدف دائما تغییر می کند و یا به عبارتی مسائل پویا بکار می رود و همانطور که در ابتدا بیان شد، مسئله ی ما مسئله ای از نوع بهینه سازی تابع ایستا است که در طول زمان و مراحل مختلف، مقادیر بهینه ی آن تغییر نخواهد کرد درنتیجه استفاده از این نوع عملگر تنها باعث همگرایی و یا نوسان اطراف بهینه سراسری خواهد شد.** درنتیجه برای این مسئله کاربردی ندارد و تنها باعث گمراهی خواهد شد. با استفاده از این عملگر در هر مرحله در یک نقطه از فضای جستجو قرار خواهیم داشت که این نقطه ضریبی از مکان قبلی است. و اگر جهش یا ریکام خیلی بالایی اتفاق بیفتد ممکن است از نقطه ی هدف دورشویم و همگرایی یا کلا صورت نگیرد یا در مراحل بسیار بیشتری اتفاق خواهد افتاد. درنتیجه با توضیحات بیان شده و تحقیقات صورت گرفته این عملگر برای این مسئله کاربردی ندارد.

3. ES : Recombination : Discrete, Survivor selection : (Mu + landa)

در این حالت نیز مانند حالت اول عمل میکنیم با این تفاوت که از نوع ریکامبینیشن Discrete استفاده کرده ایم یعنی برای ژن I ام فرزند تولید شده، به انتخاب و رندوم از یکی از ژن های i ام دو والد استفاده می کنیم.
که در زیر به ازای تعداد ابعاد ۴ و ۲۰ خواهیم داشت:



و به ازای تعداد ابعاد ۲۰ خواهیم داشت:



***** نقاطی که با هایلایت زرد مشخص شده است، نقاطی هستند که در دو مرحله یا بیشتر مقدار فیتنس یکسانی داشته اند یا به اصطلاح نقاط بهینه ی محلی میباشند.**

تقریبا خروجی این قسمت با قسمت اول یکسان است و تحلیل های مشابهی هم دارند

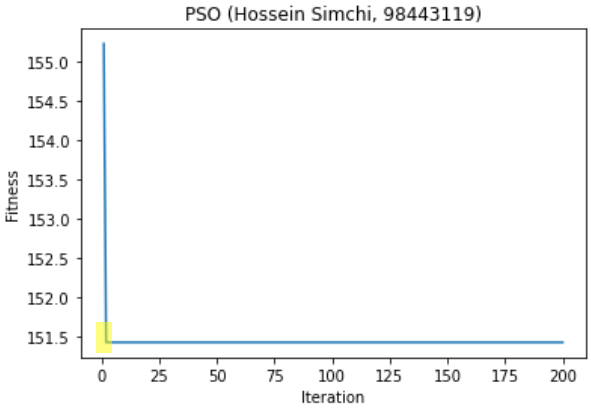
PSO

از این الگوریتم نیز برای بهینه سازی و یافتن توابع غیرخطی استفاده می شود. در این الگوریتم عملگر Recombination و Mutation به آن معنای خاص مانند چیزی که در الگوریتم قبلی دیدیم نداریم. **پس برای اجرای این الگوریتم recombination درنظر نمیگیریم و Mutation را مانند چیزی که در کلاس بحث شد درغالب یک پارامتر سرعت درنظر می گیریم**. درنتیجه به ازای هر عضو از جمعیت یک فرزند خواهیم داشت. و انتخاب بازماندگان را به صورت انتخاب ۲۰ تا (طول جمعیت) از بین فرزندان و والدین درنظر می گیریم یعنی هربار بیست تا از بهترین جواب ها را درنظر می گیریم تا به نقطه ی بهنیه سراسری همگرا شویم. در زیر پارامتر سرعت قابل مشاهده است که هربار برای مقادیر تعیین شده ای به مقدار هربردار اضافه می گردد.

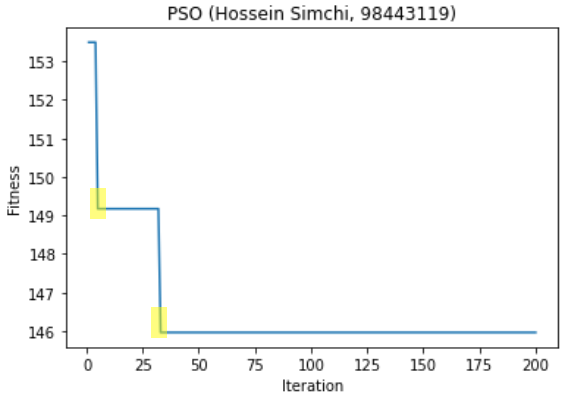
$$V = w.v \ + \ \varphi_1 v_1 \ (y - x) + \ \varphi_2 v_2 \ (z - x)$$

که در فرمول فوق، y برابر بهترین جواب ممکن برای x و z بهترین جواب ممکن تاکنون بوده است. *انتخاب پارامترهای موجود در عبارت فوق تاثیر بسیار زیادی در روند همگرایی به نقطه ی منیمم سراسری خواهد داشت. به طوری که با تغییر کوچک در پارامترهای مسئله می توان سرعت و دقت بسیار متفاوتی نسبت به حالت های دیگر بدست آورد*. برای این مسئله مقادیر مختلفی برای پارامترها درنظر گرفته شده است که در تمامی این مقادیر و تحلیل نتایج حاصل کاملا مشهود و یکسان است و آن هم همگرا نشدن به مینیمم سراسری خواهد بود. **از هر نقطه ای (جمعیت اولیه) با هر ابعاد که شروع به اجرای الگوریتم کنیم، در تعداد محدوی تکرار الگوریتم، الگوریتم‌به نزدیکترین مینیمم محلی در اون فضا همگرا خواهد شد و دلیل آن بخاطر نداشتن عملگر Recombination است**. می دانیم که عملگر Recombination باعث می شود که فضای های مختلف و تقریبا تمام فضای مسئله جستجو شود و در این الگوریتم به دلیل درنظر نگرفتن Recombination مناسب، الگوریتم در حدود جمعیت اولیه باقی می ماند و فضاهای جدید را بررسی نمی کند

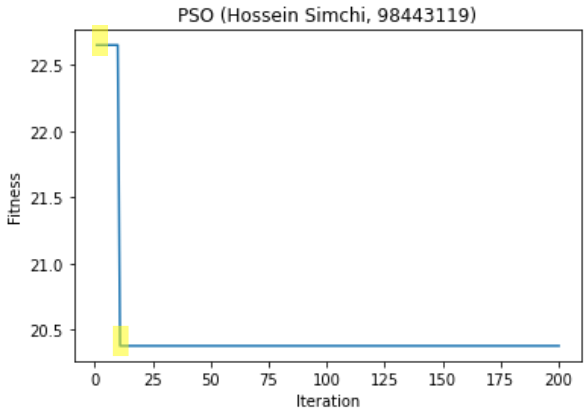
درنتیجه در تعداد محدودی تکرار، الگوریتم به بهنیه ترین مقدار محلی در آن فضـا همگرا خواهد شـد. در زیر به ازای ابعاد ۲ و ۴ می توانیم مشـاهده کنیم که الگوریتم پس از بررسـی نواحی اطراف جمعیت اولیه به بهنیه ترین مقدار در آن موقعیت همگرا می شود که کمترین مقدار بهینه محلی در آن محدوده است. سرعت همگرا شدن به بهنیه ی محلی اطراف جمعیت اولیه کاملا وابســـته به مقدار w اســـت *یعنی هرچه مقدار w را افزایش دهیم با سرعت بیشتری به نقطه ی بهینه محلی همگرا خواهیم شد.* البته مقادیر پارامترهای دیگر نیز اهمیت زیادی دارد ولی درکل اصل کار الگوریتم و نحوه ی همگرا شدن به یک نقطه مانند چیزی است که در خطوط قبلی بیان شد.



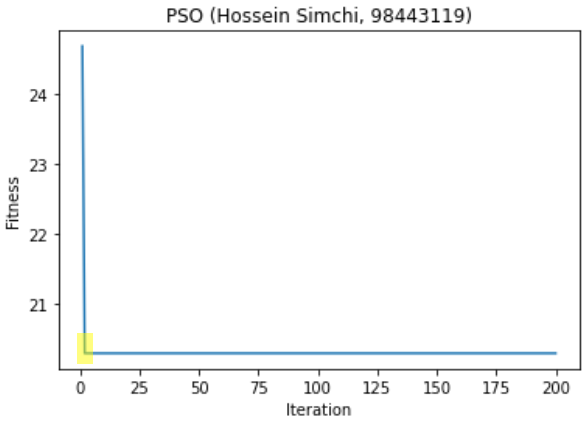
ابعاد برابر ۴ و مقدار **w=0.8**



ابعاد برابر ۴ و مقدار **w=0.4**



ابعاد برابر ۲ و مقدار **w=0.4**



ابعاد برابر ۲ و مقدار **w=0.8**

همانطور که از مقایسه ی شکل های فوق مشاهده می شود، زمانی که مقدار w را از ۰.۴ به ۰.۸ افزایش می دهیم سرعت همگرایی به بهینه ترین مینیمم محلی افزایش یافته و در یک مرحله یا تعداد مراحل کمتری به نقطه ی بهینه ی محلی همگرا شده ایم.

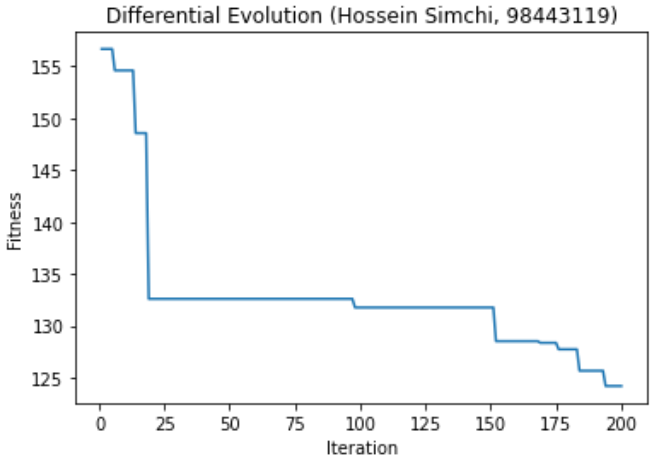
Differential Evolution (DE)

الگوریتم آخری که در این تمرین بررسی می کنیم الگوریتم معروف DE است. که بسیار شبیه ES می باشد و برای بهینه سازی توابع غیرخطی و مشتق ناپذیر استفاده می شود. در این الگوریتم از عملگر Unifrom برای recombination استفاده می کنیم یعنی به یک احتمالی یکی از بردارهای جمعیت اولیه را انتخاب می کنیم با این تفاوت که یکی از مولفه ها را به صورت رندوم انتخاب می کنیم و مقدار آن را دقیقا برابر با والد اول می گذاریم.

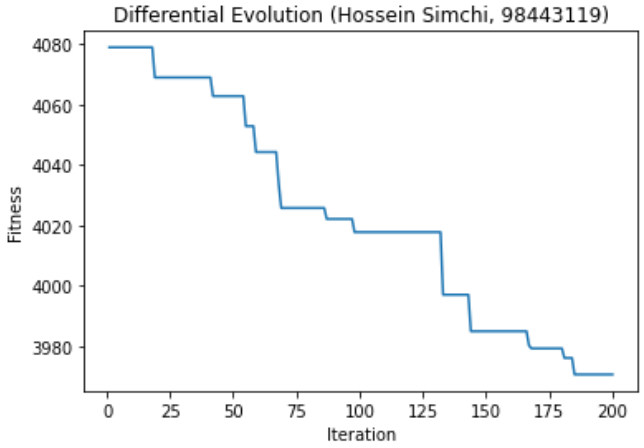
روال کار الگوریتم به صورت زیر میباشد

۱. انتخاب جمعیت اولیه به صورتی که ابعاد هر عضو از جمعیت برابر با ابعاد تابعی است که قصد داریم مینیمم آن را بدست بیاوریم
۲. بر روی اعضای جمعیت mutation را لحاظ می کنیم بدین صورت که دو بردار رندوم در جمعیت را انتخاب کرده، تفاضل دو بردار را محاسبه می کنیم و با ضریبی این مقدار را به بردار مورد نظر اضافه می کنیم. انتخاب ضریب تفاضل دو بردار نیز چالشی است که باید مقدار بهینه آن را نیز بدست آورد.
۳. Recombination بر روی اعضای جمعیت حاصل مانند چیزی که در خطوط اول توضیحات این الگوریتم بیان شد
۴. پس از گام سوم فرزندانی تولید خواهند شد که با مقایسه ی فرزندان با والدین، هرکدام فیتنس بهتری داشته باشند را انتخاب می کنیم (انتخاب بازماندگان)

برای این الگوریتم نیز مانند سه الگوریتم بیان شده در ابعاد مختلف عملکرد آن را بررسی کرده ایم که درشکل های زیر قابل مشاهده است.



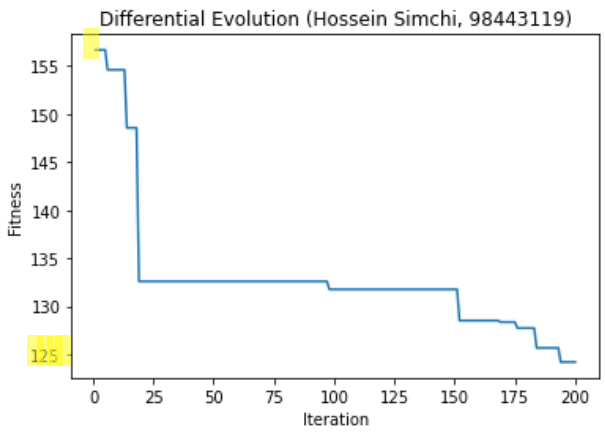
ابعاد ۴

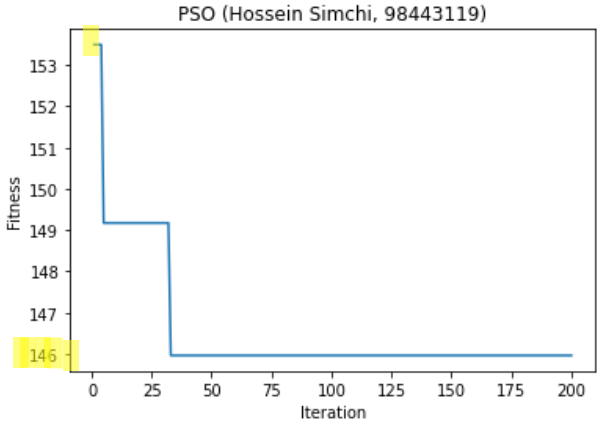


ابعاد ۲۰

تفاوت الگوریتم DE و PSO

✓ اگر شـکل های مربوط به دو الگوریتم فوق را با یکدیگر مقایسـه کنیم خواهیم دید که الگوریتم DE به مینیمم سراسری با **دقت بالاتری همگرا** خواهد شد (شکل های زیر)





با مقایسـه ی دو شـکل فوق مشـاهده می شـود که الگوریتم DE در تعداد مراحل یکسـان با الگوریتم PSO به دقت بهتری رسیده است یعنی توانسته مقدار کمتری برای تابع بدست بیاورد.

✓ تفاوت دیگر به خاطر استفاده از عملگر Recombination است؛ در الگوریتم PSO به خاطر اینکه از این عملگر استفاده نمی کنیم به بهینه محلی اطراف جمعیت همگرا می شویم و به همین دلیل هرچقدر تعداد مراحل تکرار الگوریتم را (Iteration) زیاد کنیم باز هم تاثیری در جواب حاصل نخواهیم داشت چون عملا شانس جستجو در فضای جدید را نخواهیم داشت. ولی در الگوریتم DE چون از عملگر Recombination استفاده می کنیم اگر تعداد مراحل تکرار الگوریتم را افزایش دهیم ممکن است مانند شکل های مرتبط با DE که دائما در حال فرار از نقاط بهینه ی محلی است بتوانیم نهایتا به بهینه ی سراسری همگرا شویم.

تفاوت سه الگوریتم بیان شده و انتخاب بهینه ترین الگوریتم ممکن

نام الگوریتم	معایب	مزایا
ES	اگر از انتخاب بازماندگان را از نوع Generational درنظر بگیریم ممکن است واگرایی اتفاق بیفتد	همگرایی سریع به بهینه سراسری در تعداد مراحل تکرار کم، سادگی و زمان اجرای پایین
PSO	انتخاب پارامترهای مناسب، نداشتن عملگر Recom و همگرا شدن به بهینه ی محلی و از دست دادن شانس همگراشدن به بهینه ی سراسری	بهینه ی محلی در اطراف جمعیت اولیه را با سرعت بالایی پیدا میکند.
DE	زمان اجرای بالا و تعداد مراحل بسیار زیاد جهت همگرایی به مینیمم سراسری	برخلاف PSO نهایتا می تولند در تعداد مراحل تکرار بیشتری نسبت به ES به بهینه ی سراسری همگرا شود

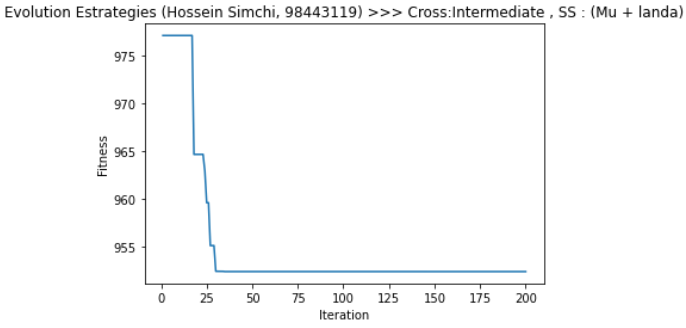
مقایسه ی دقت و سرعت باتوجه به نکات بیان شده

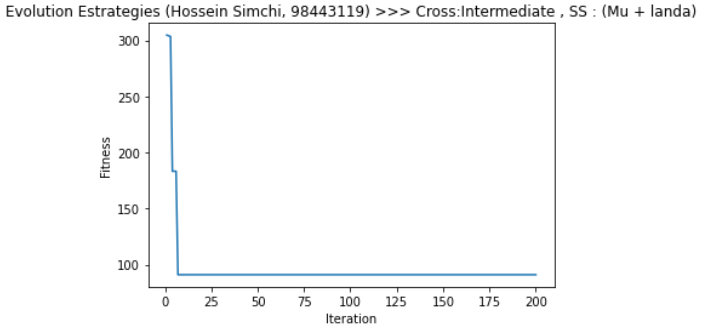
$$ES > DE > PSO$$

با توجه به جدول فوق و نکات بیان شده برای الگوریتم های فوق، از الگوریتم ES برای بهینه سازی دو تابع دیگر استفاده خواهیم کرد

استفاده از الگوریتم ES برای یافتن مقادیر بهینه برای دو تابع دیگر

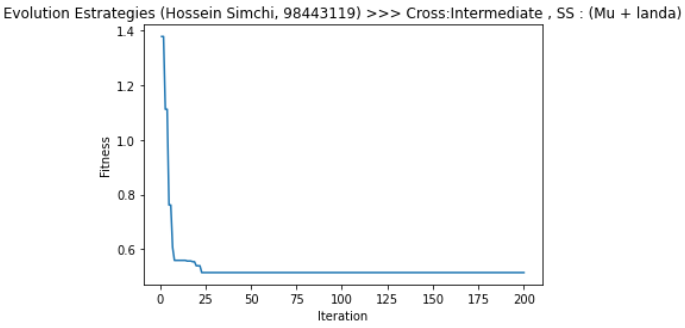
✓ بهینه سازی تابع Schwefel به ازای ابعاد ۲ و ۴

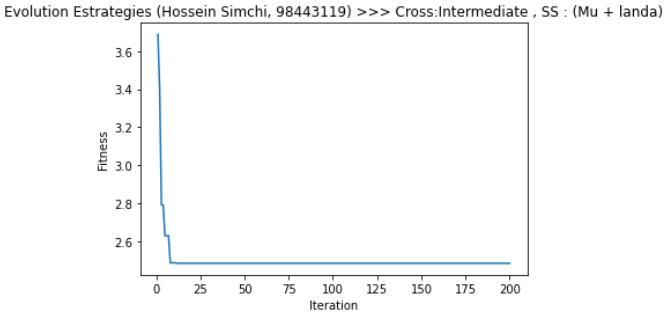




همانطور که از شـکل های فوق نیز مشـخص اسـت زمانی که تعداد ابعاد را زیاد می کنیم با تعداد مراحل ۲۰۰ خیلی به نقطه ی بهینه ی سـراسـری نزدیک نمی شـویم ولی همانطور که در توضـیحات ES بیان شـد سـرعت همگرایی بسیار سریع خواهد بود که در شکل های فوق نیز قابل برداشت است.

✓ بهینه سازی تابع Grienwank به ازای ابعاد ۲ و ۴





تقریبا می توان گفت برای تابع Grienwank نقطه ی بهینه سراسری را در تعداد اپوک ۲۰۰ تا پیدا می کند. و دلیل آن هم از شکل تابع که در بخش ابتدایی رسم کردیم مشخص است و آن هم داشتن تعداد بهینه ی محلی کمتر نسـبت به دو تابع دیگر اسـت پس طبق دانسـته های قبلی می دانیم که هرچه تعداد بهینه ی محلی کمتر باشد شانس رسیدن به بهینه ی سراسری افزایش پیدا می کند.

تمامی فایل های کدنویسی نوشته شده حاصل تلاش بنده بوده است و از هیچ کدام از کدهای آماده استفاده نشده است که در پیوست ضمیمه و ارسال گردیده است. همچنین تمامی تصاویر استفاده شده نیز ضمیمه شده است.

باتشکر از زحمات شما

پایان، حسین سیم چی