بهینه سازی فضای منحنی(Curved Space Optimization - CSO): یک روش جدید برای بهینه سازی اکتشافی

فریدون فرهی مقدم*

چکیده

در این مقاله یک روش جدید برای بهینه سازی اکتشافی معرفی و پیاده سازی می شود. این روش، بر پایه انحنای فضای جستجو استوار است؛ بدین معنی که فضای متغیرها در محل قرار گرفتن نسل قبل دچار کشیدگی می شود. بر همین اساس در این الگوریتم به هر عضو جمعیت، "جرم" (mass) گفته می شود. روش جدید، الگوریتم جستجوی تصادفی که نقاط جدید در آن کاملا بر پایه انتخاب تصادفی مشخص می گردند را به گونه ای تعمیم می دهد که عملکرد ضعیف آن در توابع تست شناخته شده، به طور موثری بهبود می یابد؛ به گونه ای که با الگوریتم های معروفی همچون الگوریتم ژنتیک قابل مقایسه می گردد. در این روش برای تولید هر عضو جدید از نسل بعدی از کلیه اطلاعات نسل قبلی در قالب یک تابع احتمال روی کل فضای جستجو استفاده می شود. این تابع، احتمال حضور نقاط بهینه را در کل فضای متغیرها مشخص می سازد و اجرام جدید با توجه به این احتمال حضور به صورت تصادفی از کل فضای متغیرها انتخاب می گردند. روش مذکور از نظر عملکرد با تعدادی از الگوریتم های معروف مقایسه شده و نتایج اعمال آن روی توابع تست در مقایسه با الگوریتم ژنتیک ارائه شده است. نتایج حاصله نشان می دهد که الگوریتم جدید در اکثر توابعی که مورد تست قرار گرفته است دارای عملکرد بهتری از الگوریتم ژنتیک بوده است.

کلمات کلیدی

بهینه سازی اکتشافی، بهینه سازی فضای منحنی، الگوریتم ژنتیک، جستجوی تصادفی، ذوب شبیه سازی شده، جرم، انحنای فضا

Curved Space Optimization (CSO): A novel approach to heuristic optimization

F. Farrahi-Moghaddam
Electrical Eng. Dept., S.B. Univ. of Kerman, ffarrahi@mail.uk.ac.ir

Abstract

In this paper a novel heuristic optimization method is introduced and implemented. The method is based on the concept of introducing curvature in the search space; the space is bent near the points of the previous generation. We call any member of population of this algorithm a "mass". The method modifies the "random search" algorithm, which its new generations are produced in a fully random way, in such a way that its weak performance for well-known test functions is improved drastically; so that, its performance is comparable with famous algorithms such as Genetic algorithm. In this method, the next generation is produced using all of previous generation data through a probability function on all of search space. This probability function determines the probability of presence of optimal points on the whole of variables space. The new points (masses) are obtained randomly from all parts of the variables space using this probability function. A comparison is made between the proposed method and well-known algorithms from performance point of view, and also results of application of the method are compared with genetic algorithm results for some test functions. The results show that the new algorithm has better performance in the majority of test functions in comparison to genetic algorithm.

Keywords

heuristic optimization, curved space optimization, genetic algorithm, random search, simulating annealing, mass

* عضو هیات علمی بخش مهندسی برق، دانشگاه شهید باهنر کرمان، ffarrahi@mail.uk.ac.ir

۱- مقدمه

همانطور که در مقاله " No Free Lunch Theorem for Optimization" آمده است، میانگین عملکرد تمام روشهای جستجو بر روی تمام توابع با هم برابر است [1]. این بدین معنی است که نمی توان هیچ الگوریتم جستجویی را ارائه کرد که نسبت به سایر الگوریتم ها بر روی تمام توابع، عملکرد بهتری داشته باشد. بلکه هر یک از الگوریتم های موجود دارای حوزه عملکردی خاص خود در مجموعه توابع می باشد. آنچه که باعث شهرت بعضی از الگوریتم های جستجو به توانمندی بیشتر شده، تطابق حوزه عملکرد آنها در دامنه توابع با توابع شناخته شده کاربردی در علوم مختلف می باشد.

همانگونه که مشخص است الگوریتم ساده جستجوگر تصادفی دارای کاربرد محدودی در مسائل بهینه سازی می باشد. اما این الگوریتم دارای دو شاخصه مهم و خوب است که مورد توجه این مقاله قرار دارد. عدم گیر افتادن در نقاط بهینه محلی و عدم برتری دادن نقاط هم رتبه بر یکدیگر. به عنوان مثال در الگوریتم ژنتیک احتمال افتادن زود هنگام در بهینه محلی وجود دارد و از جهش برای بیرون آمدن از این بهینه های محلی استفاده می گردد. همچنین مجموعه نقاطی که به عنوان کاندیدای نسل بعدی مورد توجه قرار می گیرند نیز محدود است (کرموزومهای نسل بعد الگوریتم ژنتیک مستقیما از محدوده اطلاعات نسل قبل به وجود می آیند). در واقع ما در اینجا با استفاده از یک مفهوم فیزیکی عملکرد جستجوگر تصادفی را به گونه ای دگرگون ساخته ایم تا ضمن حفظ نسبی این دو خاصیت خوب ذکر شده، حوزه توابع با عملکرد قابل قبول این الگوریتم، تا حد امکان به حوزه توابع شناخته شده نزدیک گردد. مفهوم فیزیکی مورد استفاده انحنای فضا در نظریه نسبیت عام می باشد که در حضور جرم، فضای اطراف جرم، انحنا پیدا می کند. در اینجا جرم را با شایستگی[1] نقاط و میزان انحنای فضا را با تابع احتمالی که در ادامه توضیح خواهیم داد متناظر ساخته ایم. در واقع ایده اصلی روش جدید این است که هرچه فضا در اطراف نقاطی با شایستگی بیشتر، انحنای بیشتری پیدا کند، احتمال جستجوی جستجوگر تصادفی در اطراف آن نقطه نیز بیشتر می گردد.

ما ابتدا در بخش ۲ به بررسی الگوریتم های شناخته شده فعلی می پردازیم و سعی می کنیم خصوصیات بارز آنها را برشماریم. در بخش ۳ ایده الگوریتم جدید و خصوصیات بارز آن را به تفصیل بیان می کنیم. در بخش ۴ بین خصوصیات الگوریتم های شناخته شده و الگوریتم جدید مقایسه ای انجام می دهیم. بخش ۵ را به نحوه پیاده سازی الگوریتم جدید اختصاص داده ایم و در بخش ۶ نتایج مقایسه الگوریتم جدید و الگوریتم ژنتیک روی توابع تست شناخته شده را آورده ایم.

۲- الگوریتم های شناخته شده

الگوریتم های زیادی در مبحث جستجوی اکتشافی مطرح هستند که ما در اینجا فقط به عنوان نمونه به آنهایی اشاره خواهیم کرد که در مسیر تبیین الگوریتم جدید به ما کمک می کنند. از این جمله می توان به الگوریتمهای جستجوی تصادفی، ذوب شبیه سازی شده و الگوریتم ژنتیک اشاره کرد؛ که در ادامه همین بخش به آنها اشاره خواهیم کرد. در اینجا ذکر این نکته ضروری است که در کل الگوریتم های جستجو بر اساس دو منطق کلی عمل خود را انجام می دهند که کسر زیادی از موفقیت این الگوریتم ها در به دست آوردن جواب بهینه، چگونگی به انجام رسانیدن این دو عمل می باشد؛ کاوش[2] وبهره وری[3]. کاوش مربوط به آن قسمت از الگوریتم می شود که الگوریتم می بایست نواحی جدیدی را که احتمال وجود بهینه در آنها وجود دارد پیدا کند و بهره وری مربوط به آن قسمت از الگوریتم می گردد که نقاط پیدا شده در قسمت قبل به دقت مورد توجه و استفاده قرار گیرند. الگوریتمی که یکی از این دو پارامتر را در نظر نگیرد هر چند که در پارامتر دوم هم توفیق داشته باشد در به دست آوردن مقدار بهینه تابع به جایی نخواهد رسید. در واقع توجه به هر کدام از این پارامترها، پارامتر دوم را در نقطه ضعف قرار می دهد و شاید یکی از عوامل توفیق الگوریتم های جستجو برقراری تعادل بین توجه کردن به این دو پارامتر مهم باشد.

۲-۱- جستجو گر تصادفی

از نظر اجرا شاید ساده ترین الگوریتم جستجو اکتشافی، الگوریتم جستجوی تصادفی باشد. عملکرد این الگوریتم بدین گونه است که از کل فضای متغیر ها نقاطی به تصادف انتخاب می گردند و انتخاب نقاط بعدی هیچ وابستگی ای به نقاط قبلی و مقدار تابع شایستگی آنها ندارد. الگوریتم کار خود را ادامه می دهد تا شرط پایان الگوریتم فرا رسد. از نظر اجرا الگوریتم فوق الذکر دارای هیچ پیچیدگی و محاسباتی نمی باشد و اگر به الگوریتم زمان کافی داده شود قادر است جواب مناسب را به دست آورد. مزیت این الگوریتم یکنواختی نگاه به نقاط مختلف فضای متغیر ها می باشد که این امر باعث می شود تا این الگوریتم در بهینه های محلی گیر نیافتد و همچنین هیچ نقطه ای از فضای متغیرها از دید این الگوریتم دور نماند. در کل می توان گفت که از نظر کاوش الگوریتم جستجوی تصادفی دارای عملکرد قابل قبولی می باشد؛ ولی ضعف عمده آن، عدم وجود همگرایی و بهره وری در این الگوریتم است. در واقع هیچ رابطه ای بین نقاط نسل قبل و نسل بعد در این الگوریتم وجود ندارد.

۲-۲- ذوب شبیه سازی شده

بر عکس الگوریتم جستجوی تصادفی که هیچ رابطه منطقی ای بین نقاط نسل قبل و بعد آن وجود ندارد؛ در الگوریتم SA پایه انتخاب نقاط جدید دقیقا نقاط قبل محسوب می گردند. در این الگوریتم یک نقطه به صورت تصادفی در فضای متغیرها به عنوان نقطه اول انتخاب می گردد. برای به وجود آوردن نقطه بعدی اول نقطه ای (p_{new}) از رابطه $p_{new} = d\, p_{old}$ محاسبه می گردد. که در واقع پارامتر d

نقطه جدید را به صورت تصادفی یک گام حرکت می دهد. در ابتدای اجرای الگوریتم گامها بایستی بزرگ باشند تا اصل کاوش به خوبی انجام پذیرد و به تدریج از اندازه گامها در طول اجرای الگوریتم کاسته شود. بعد از اینکه نقطه جدید به دست آمد اگر میزان شایستگی آن از میزان شایستگی نقطه قبل کمتر بود (در الگوریتم های کمینه یاب) نقطه جدید جانشین نقطه قبل می گردد و اگر کمتر نبود پذیرش نقطه جدید منوط به برآورده شدن شرط $r \le e^{(f(p_{old})-f(p_{new}))/T}$ می باشد. r یک عدد تصادفی بین ۰ و ۱ می باشد و T پارامتری است که متناظر دما انتخاب شده است. هرچه میزان دما بیشتر باشد احتمال انتخاب نقاط با شایستگی ای بیش از نقطه قبلی بالا تر خواهد بود و بدین صورت عمل کاوش ادامه پیدا خواهد کرد و هر چه دما کمتر شود تعداد این نقاط پذیرفته شده کاهش می یابد تا در دمای ۰ که احتمال انتخاب این گونه نقاط به صفر می رسد و فقط عمل بهره وری تا همگرایی کامل انجام می پذیرد. این فرآیند تا برآورده شدن شرایط ختم الگوریتم ادامه پیدا می کند. در هر مرحله اجرای الگوریتم پارامترهای d و T باید به گونه ای مناسب به تدریج کاهش یابند تا همگرایی به وجود آید [2,3].

۲-۳- الگوریتم ژنتیک

یکی از قویترین الگوریتمهای موجود در مبحث جستجوی اکتشافی؛ الگوریتم ژنتیک می باشد. این الگوریتم که الهام گرفته از یک پدیده کاملا طبیعی است در پیدا کردن جوابهای موثر روی توابع تست شناخته شده عملکرد بالایی از خود نشان داده است. در این الگوریتم از تعدادی کروموزوم به عنوان جوابهای تابع مورد تست استفاده می گردد. معمولا نسل اول همانند اغلب روشها به صورت تصادفی انتخاب می گردد؛ اما برای تشکیل نسل دوم در ابتدا بایستی یکسری از کروموزومهای موجود به عنوان نسل والد انتخاب گردند که این انتخاب به روشهای مختلفی انجام می پذیرد مثلا کروموزمهایی با بیشترین مقدار شایستگی انتخاب گردند. البته تعداد کروموزمهایی که به عنوان نسل والد انتخاب می شوند نیز اهمیت به سزایی دارد. در مرحله بعدی بایستی از نسل والد کروموزمهایی برای تولید افراد جدید انتخاب گردند که این مرحله نیز به روشهای مختلفی از جمله انتخاب تصادفی والدها انجام می پذیرد. مرحله بعدی آمیزش والدهای انتخاب شده است که به عمل همبری در الگوریتم ژنتیک معروف می باشد. عمل همبری را نیز به شیوه های مختلفی می توان به انجام رسانید. چون که در هر مرحله کروموزمهای ضعیف با کروموزمهای قوی تر جانشین می گردند، الگوریتم به سمت نقاط بهینه همگرا می گردد. اما این همگرایی ممکن است حول یک نقطه کمینه محلی صورت پذیردکه برای خروج از این حالت از جهش در مراحل مختلف روی کروموزمها استفاده می شود. تا مرکز همگرایی در صورت لزوم به نواحی دیگر فضای متغیر ها نیز منتقل گردد. عمل فوق در مراحل مختلفی تا رسیدن به شرایط ختم الگوریتم ادامه پیدا می کند. شیوه عملکرد الگوریتم ژنتیک بدین گونه است که نقاط پراکنده شده به صورت تصادفی در مرحله اول حلقه محاصره ای حول نقاط مرکزی تشکیل می دهند و در هر مرحله اجرای الگوریتم این حلقه محاصره از مسیرهایی که دارای شایستگی کمتر می باشند تنگ تر و تنگ تر می گردد تا به نقطه بهینه رسانیده شود [2,5,6,7].

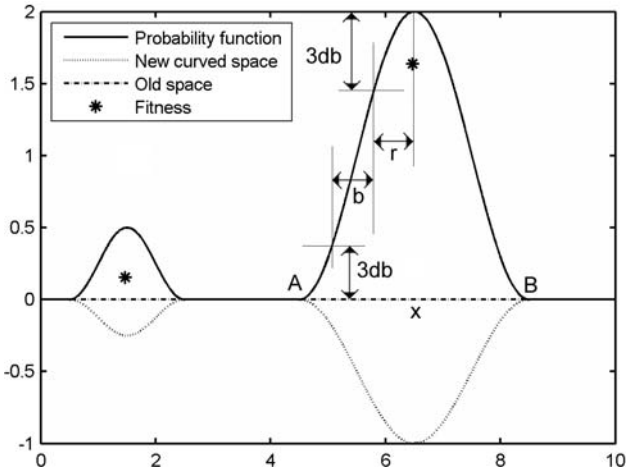
۳- الگوریتم CSO

در این بخش ایده اصلی الگوریتم بهینه سازی فضای منحنی مطرح خواهد شد و خصوصیات این الگوریتم به تفصیل بیان خواهند شد. در ابتدای بخش ایده محوری الگوریتم بیان خواهد شد، بعد نحوه عملکرد الگوریتم توضیح داده خواهد شد و پس از آن تابع احتمال معرفی خواهد شد که برای عملی کردن الگوریتم مورد استفاده قرار می گیرد. سپس شعاع نفوذ تعریف خواهد شد و شیوه همگرایی و جهش الگوریتم نیز در آخر بیان خواهد شد. در انتها خصوصیات عملکردی الگوریتم با توجه به تمامی پارامترهای آن مورد بحث قرار خواهد گرفت.

۳-۱- انحنای فضا

هنگامی که در اکثر الگوریتم های جستجو، نسل اول به صورت تصادفی در تمام دامنه ورودی پراکنده می گردد، قدرت یک الگوریتم خوب در این است که با توجه به اطلاعات جستجو شده یعنی محل قرار گرفتن آنها و مقدار شایستگی آنها نقاط جدیدی را پیشنهاد کند که احتمال منجر شدنشان به جواب بهینه بیشتر باشد. به صورت آماری می توان انتظار داشت که نقاط بهینه در نزدیکی نقاطی با شایستگی بیشتر (نقاط خوب) قرار گرفته باشند تا در نزدیکی سایر نقاط (نقاط متوسط و بد). در الگوریتم ارائه شده شیوه برتری دادن به اجرام خوب نسبت به اجرام متوسط و بد بدین صورت است که ما فضای متغیر ها را در محلی که جرم قرار گرفته است به میزان شایستگی آن انحنا می دهیم. دقیقا همان وضعیتی که در نظریه نسبیت عام بین جرم و فضا اتفاق می افتد. حال برای انتخاب نسل بعدی ما با یک فضا روبرو هستیم که در n (تعداد اعضای جمعیت) جرم نسل قبلی دچار انحنا شده است. بعضی جاها انحنا زیاد و بعضی جاها انحنا کم می باشد (شکل ۱). بدیهی است که طول محور x' در شکل (۱) از آنچه که بوده ، x، بزرگ تر شده است. در اینجا x محور قبل و x' محور جدید را مشخص می سازند. یک تبدیل $x' = A(x)$ با توجه به انحنای به وجود آمده مقادیر محور x را به x' و بلعکس تبدیل می کند. تنها کاری که ما باید برای انتخاب نسل بعد انجام دهیم این است که نقاط تصادفی جدید را روی محور x' انتخاب نماییم (بر خلاف جستجو گر تصادفی که از محور x استفاده می کند). در واقع ما در هر مرحله همان جستجوی تصادفی را انجام می دهیم با این تفاوت که محور و انحنای آن را عوض می کنیم. بعد که اجرام تصادفی را روی محور x' ریختیم آنها را با استفاده از عکس تبدیل $A(x)$ به x تبدیل می کنیم و محل اجرام جدید را به دست می آوریم. کاملا واضح است که

اجرام جدید، حول جرمهایی از نسل قبل که دارای شایستگی بیشتری بوده اند بیشتر جمع شده اند تا اجرامی که دارای مقدار شایستگی کمتری بوده اند. همانطور که روی شکل (۱) نیز دیده می شود فاصله بین نقاط A و B در روی محور x بسیار کمتر از محور x' است و چون ما اجرام جدید را به صورت تصادفی و یکنواخت روی x' انتخاب نموده ایم پس روی x دارای فشردگی جرمها خواهیم بود. از اجرام نسل قبل و جدید یکسری جرم را که دارای بهترین شایستگی می باشند را به عنوان نسل بعد انتخاب کرده و این مراحل را آنقدر تکرار می کنیم تا به شرایط ختم الگوریتم برسیم.



شکل (۱): انحنای فضای، r: شعاع نفوذ، b: ناحیه مرزی

۳-۲- تابع احتمال

همانگونه که در بخش قبل توضیح داده شد در الگوریتم CSO در هر مرحله محورها و فضای متغیر ها تغییر می کنند و می بایستی که توسط تبدیل های $A(x)$ جرمهای جدید را محاسبه و به دست آورد. برای اینکه از انجام این تبدیل ها در هر مرحله از اجرای الگوریتم فارغ شویم و محاسبات هر مرحله را کاهش دهیم، از تکنیک دیگری به جای منحنی کردن محورها استفاده می کنیم که تاثیر آن دقیقا برابر با انحنای محورها و فضا باشد. این تکنیک عبارت است از تعریف یک تابع احتمال روی کل فضای متغیر ها. این تابع احتمال که سطح زیر آن روی کل فضای متغیرها برابر واحد می باشد مشخص می سازد که شانس انتخاب هر نقطه از فضای متغیرها چه اندازه می باشد. به عنوان مثال اگر تابع احتمال در نقطه ای صفر باشد. در انتخاب تصادفی، آن نقطه هرگز انتخاب نخواهد شد و اگر تابع احتمال در یک نقطه از فضا تابع دلتای دیراک باشد آنگاه شانس انتخاب آن نقطه قطعی می باشد. در صورتی که تابع احتمال ثابت باشد یعنی مقدار آن در تمام نقاط فضا با هم برابر باشد، الگوریتم جدید دقیقا همانند جستجوی تصادفی عمل خواهد کرد. در واقع در جستجو گر تصادفی تابع احتمال برابر با $p(x) = 1/S$ می باشد که S نشان دهنده مساحت فضای s (تعداد متغیرهای مسئله) بعدی متغیر ها می باشد. در الگوریتم جدید همانگونه که در شکل (۱) دیده می شود تابع p(x) در نقاطی با شایستگی بیشتر دارای قله های بزرگتری نسبت به نقاطی با شایستگی کمتر می باشد.

۳-۳- تابع انحنای فضا و شعاع نفوذ

برای مشخص کردن میزان و چگونگی انحنای فضا یا انحنای تابع احتمال در یک نقطه (تبدیل $x' = A(x)$)، منحنی های زیادی می توان در نظر گرفت؛ منحنی هایی با شیب زیاد یا شیب کم؛ منحنی هایی با دامنه بزرگ یا دامنه کوچک؛ که هر کدام از این منحنی ها می تواند تاثیر خاص خودشان را روی کارکرد الگوریتم جدید بگذارد. هر چه دامنه بزرگ تر باشد نقاطی از اطراف جرم مورد نظر که تحت تاثیر شایستگی آن قرار می گیرند بیشترند ؛ که این مساله نقش به سزایی در همگرایی الگوریتم بازی می کند. ما در اینجا از جرم مورد نظر تا ۳ دی بی افت دامنه تابع احتمال را شعاع نفوذ شایستگی آن نقطه تعریف می کنیم. این شعاع نفوذ تقریبا همان نقشی را بازی می کند که پارامتر d در الگوریتم SA دارد و نحوه کاهش آن میتواند نقش کلیدی در به ثمر رساندن الگوریتم داشته باشد. کمی یا زیادی شیب کاهش انحنای تابع احتمال به باریک بودن یا ضخیم بودن مرز بین نقاط تحت تاثیر نقطه مورد نظر و نقاط تاثیر نگرفته منجر شود. در شکل (۱) نواحی شعاع نفوذ و مرز بر روی یک منحنی مثال مشخص شده است.

۳-۴- همگرایی و جهش

طبق تعریف تابع احتمال، احتمال انتخاب نقطه تصادفی بعدی در ناحیه تحت تاثیر شعاع نفوذ بیشتر از نقاط دیگر فضای متغیر ها می باشد هرچند انتخاب نقطه جدید در سایر نواحی نیز منتفی نمی باشد. هرچه شایستگی یک جرم بیشتر باشد به دلیل انحنای بیشتر تابع احتمال حول آن، احتمال انتخاب نقاط جدید حول آن نقطه نیز به مراتب افزایش می یابد. این به منزله همگرایی حول این جرم می باشد. هر چه که شعاع نفوذ کمتر انتخاب شده و ناحیه مرزی نازکتر انتخاب گردد، این همگرایی شدید تر خواهد بود. به همین منظور در ابتدای کار الگوریتم شعاع نفوذ مقدار بزرگی انتخاب می گردد تا تمام نقاط فضای متغیر ها تحت پوشش مناسب قرار گیرند و در هر مرحله اجرای الگوریتم شعاع نفوذ به تدریج کاهش پیدا می کند. این کاهش تدریجی را می توان توسط تابع تبخیر به وجود آورد. با تعریف پارامتر α مطابق فرمول (۱) در هر مرحله شعاع نفوذ به اندازه α کاهش پیدا می کند (α عددی بین صفر و یک و نزدیک به یک انتخاب می گردد). انتخاب α حول ۰/۹ نتایج بهتری به همراه دارد.

$$r_{new} = \alpha\, r_{old} \qquad (۱)$$

برای توابع با بهینه های محلی کم، α کوچکتر منجر به همگرایی سریعتر خواهد شد و برای توابعی شامل بهینه های محلی زیاد ، α بزرگتر می تواند در قبال زمان بیشتر جستجو، منجر به جوابهای بهتری شود. به هر به ازای هر مقدار α ممکن است الگوریتم حول یک نقطه کمینه محلی گیر بیافتد که در اینجا با استفاده از تکنیک جهش مطابق با جهش تعریف شده در الگوریتم ژنتیک آن را بیرون خواهیم

آورد. عملکرد جهش در این الگوریتم دقیقا مشابه عملکرد جهش در الگوریتم ژنتیک خواهد بود.

۳-۵- خصوصیات

چونکه الگوریتم جدید از انتخاب تصادفی بر پایه یک تابع احتمال برای تولید نسل های بعدی استفاده می کند لذا در این الگوریتم میزان محاسبات کم خواهد بود و با شیوه ای که در بخش ۵ ارائه شده است خواهیم دید که این محاسبات به حداقل ممکن خود خواهد رسید. با توجه به شیوه عملکرد الگوریتم ما هیچ نقطه ای از فضا را در هیچ مرحله ای به صورت ۱۰۰ درصد از دست نمی دهیم بلکه فقط احتمال انتخاب در نواحی غیر محتمل را کاهش می دهیم بدین معنی که اگر از جهش نیز استفاده نکنیم احتمال پرش الگوریتم از کمینه های محلی به نواحی جدید وجود دارد اما ممکن است طبق تعریف تابع احتمال بسیار کم شده باشد که استفاده از جهش لازم به نظر برسد. چونکه ما از تابع تصادفی برای انتخاب استفاده می کنیم لذا تمامی نقاط نواحی مختلف شانس انتخاب شدن خواهند داشت و مجموعه انتخاب همانند الگوریتم ژنتیک محدود به اطلاعات نسل قبل نخواهد بود. در این الگوریتم به محض تشخیص اینکه در یک ناحیه ممکن است شایستگی نامناسب باشد جستجو در آن ناحیه با احتمال کمتری ادامه پیدا می کند و بیشتر توان الگوریتم روی نواحی محتمل تر صرف خواهد شد. با تعریف مناسب α نیز می توان هر میزان که لازم باشد سرعت همگرایی را با توجه به نوع تابع کند یا تند نمود ضمن اینکه یک مقدار پیش فرض ۰/۹ نیز برای α موجود می باشد که برای تمام توابع جوابهای قابل قبولی ارائه می دهد. در صورت گیر افتادن الگوریتم در کمینه محلی الگوریتم توسط جهشهای مقطعی که در نسل های خود ایجاد می کند به اکتشاف نواحی جدید در کل فضای متغیر ها خواهد پرداخت.

۴- مقایسه عملکردی الگوریتم جدید و الگوریتم های شناخته شده

در بخشهای قبل به موارد تفاوت و تشابه جستجوگر تصادفی و الگوریتم CSO اشاره کردیم. اگر بخواهیم الگوریتم جدید را با الگوریتم SA مقایسه نماییم خواهیم دید بر خلاف اینکه الگوریتم SA یک الگوریتم گروهی نیست اما نحوه انتخاب نقاط جدید در این الگوریتم با توجه به پارامتر d آن شباهتهای زیادی به شعاع نفوذ در الگوریتم جدید دارد. در مقام مقایسه با الگوریتم ژنتیک باید گفت که در الگوریتم ژنتیک زمانی که والدها انتخاب می گردند طبق یک الگوی خاص بچه ها نیز تولید می گردند که این بچه ها حاصل اطلاعات تنها دو والد خود خواهند بود. حال آنکه در الگوریتم CSO اطلاعات تمامی والد ها در قالب تابع احتمال گنجانده شده اند و بچه های جدید حاصل این اطلاعات جمعی می باشند. همچنین محدودیت تعداد نقاط ممکن در الگوریتم ژنتیک در CSO به هیچ وجه وجود ندارد.

۵- پیاده سازی الگوریتم

نحوه پیاده سازی الگوریتمی که در این مقاله معرفی گردید مستقیما مربوط می گردد به نحوه انتخاب و چگونگی تابع احتمال. ما در اینجا بخش ساده ترین روش پیاده سازی الگوریتم جدید را بیان می کنیم. مطمئنا با بهینه کردن این روش پیاده سازی می توان نتایج بهتری را از الگوریتم جدید به دست آورد؛ اما نتایج به دست آمده از همین روش پیاده سازی برای معرفی الگوریتم جدید و مقایسه آن با الگوریتم ژنتیک کافی به نظر می رسد. در ابتدا شبه کد الگوریتم آورده شده و در ادامه بندهایی از این شبه کد که احتیاج به توضیح داشته مورد بررسی قرار گرفته است.

۵-۱- شبه کد الگوریتم

- پیکره بندی
- تولید و محاسبه شایستگی نسل اول
- تکرار مراحل زیر تا رسیدن به شرایط ختم
  - تولید نسل جدید با توجه به تابع احتمال و محاسبه شایستگی آنها
  - ایجاد جهش در موقعیت بعضی از اجرام و محاسبه شایستگی آنها
  - انتخاب نسل بعد از میان بهترین های نسل قبل و نسل جدید
  - محاسبه عمق نفوذ جدید برای تابع احتمال با توجه به پارامتر α

۵-۲- پیکره بندی

در قسمت پیکره بندی الگوریتم پارامترهای مختلف آن تنظیم می گردند. مهمترین پارامترهای آن که می بایست به درستی در این قسمت مشخص شوند میزان تبخیر عمق نفوذ تابع احتمال می باشد که توسط پارامتر α مشخص می گردد. همچنین نرخ جهش در هر مرحله نیز می بایستی در این قسمت مشخص گردد.

۵-۳- تولید و محاسبه شایستگی نسل اول

نسل اول کاملا به صورت تصادفی در فضای متغیر ها استقرار پیدا می کنند و مقدار شایستگی آنها توسط تابع شایستگی تعیین می گردد.

۵-۴- شرایط ختم

شرایط ختم الگوریتم را می توان همانند شرایط ختم سایر الگوریتم های موجود، رسیدن الگوریتم به مقدار بهینه ای خاص، یا تکرار الگوریتم به اندازه ای مشخص شده یا رسیدن به حد خاصی از همگرایی در نظر گرفت.

۵-۵- تولید نسل جدید با توجه به تابع احتمال

در ساده ترین حالت تابع احتمال را به صورت تابع پله در نظر می گیریم که در فضای s بعدی به شکل یک مکعب حول هر یک از اجرام نسل قبل خواهد بود. در ابتدا مکعبها را با اضلاعی برابر با کل فضای جستجو در نظر می گیریم. بدین صورت که اگر متغیر x_1 تابع بین دو مقدار $x_{1_{high}}$ و $x_{1_{low}}$ متغیر باشد در مرحله اول طول ضلع مکعب فضای s بعدی در بعد x_1 را برابر با $x_{1_{high}} - x_{1_{low}}$ در نظر می گیریم. مسلم است که در مرحله اول ممکن است قسمتی از مکعب ها خارج از فضای متغیرها قرار گیرد. بدیهی است اندازه عمق نفوذ تابع احتمال در این حالت برابر با $\left(x_{1_{high}} - x_{1_{low}}\right)/2$ می باشد. اندازه مکعب های فضای s بعدی ای که به اجرام نسل قبل اختصاص می دهیم همه با هم برابر است اما احتمالی که به هر یک نسب می دهیم دقیقا تابعی از مقدار شایستگی آن جرم در مرحله قبل می باشد. لازم به ذکر است که در این شیوه پیاده سازی احتمال انتخاب خارج از فضای مکعب ها برابر با صفر می باشد. احتمال انتخاب مکعب مربوط به نقطه iام را با p(i) مشخص می سازیم.

$$p(i) = \frac{2^{-r(i)}}{\sum_{j=1}^{n} 2^{-j}}, 1 < i < n \qquad (۲)$$

r(i) رتبه نقطه iام از نسل قبل می باشد. مثلا اگر هدف الگوریتم پیدا کردن کمینه تابع باشد جرمی از نسل قبل که دارای کمترین مقدار باشد دارای r(i)=1 و جرمی که دارای بیشترین مقدار باشد دارای r(i)=n خواهد بود. البته دامنه انتخاب نقاط جدید را می توان محدود به m نقطه برتر نسل قبل نمود.

$$p(i) = \frac{2^{-r(i)}}{\sum_{j=1}^{m} 2^{-j}}, 1 < i < m \le n \qquad (۳)$$

همانطور که از تعریف p(i) مشخص است مجموع تمامی آنها برابر با یک خواهد بود. در واقع کاری که اینجا انجام داده ایم این است که برای راحتی اجرای الگوریتم به جای اینکه تابع p(x) را به گونه ای انتخاب کنیم که انتگرال آن روی کل فضای متغیر ها برابر واحد باشد، p(i) را به گونه ای تعریف کردیم که $\sum_{i=1}^{m} p(i) = 1$. بدین صورت برای انتخاب نقاط جدید، اول ناحیه مورد جستجو را با توجه به p(i) انتخاب می کنیم بعد که ناحیه انتخاب شد در آن ناحیه (مکعب s بعدی) به صورت تصادفی نقطه ای بر میگزینیم. مراحل زیر برای انتخاب n نقطه جدید با توجه به p(i) انجام می شوند.

- مراحل زیر را n با تکرار کن
  - یک عدد تصادفی a بین صفر و یک انتخاب کن
  - k را به گونه ای انتخاب کن که:

$$\sum_{j=1}^{k-1} p(j) \le a < \sum_{j=1}^{k} p(j)$$

  - درون مکعب فضای s بعدی kام یک نقطه به تصادف به عنوان نقطه جدید بعدی انتخاب کن

۶- نتایج پیاده سازی

در این بخش نتایج مقایسه عملی الگوریتم ژنتیک و CSO روی توابع تست شناخته شده جدول (۱) آورده شده است.

جدول (۱):توابع تست

#	تابع	حدود
۱	$f_1 = \sum_{i=1}^{n} x_i^2$	$[-۱۰۰, ۱۰۰]^n$
۲	$f_2 = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-۱۰, ۱۰]^n$
۳	$f_3 = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	$[-۱۰۰, ۱۰۰]^n$
۴	$f_4 = \sum_{i=1}^{n-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right)$	$[-۳۰, ۳۰]^n$
۵	$f_5 = \sum_{i=1}^{n} \left(\lfloor x_i + 0.5 \rfloor\right)^2$	$[-۱۰۰, ۱۰۰]^n$
۶	$f_6 = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	$[-۱/۲۸, ۱/۲۸]^n$
۷	$f_7 = \sum_{i=1}^{n} x_i \sin\left(\lvert x_i \rvert\right)$	$[-۵۰۰, ۵۰۰]^n$
۸	$f_8 = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	$[-۵/۱۲, ۵/۱۲]^n$
۹	$f_9 = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$[-۳۲, ۳۲]^n$
۱۰	$f_{10} = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-۶۰۰, ۶۰۰]^n$

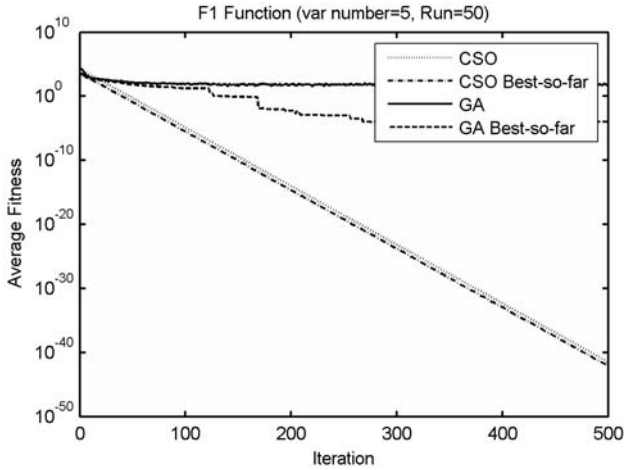
تمامی توابع با تعداد متغیرهای ۵ و ۳۰ مورد آزمایش قرار گرفتند. در آزمایشهای با ۵ متغیر هر الگوریتم تا ۵۰۰ نسل و در آزمایشهای با ۳۰ متغیر هر الگوریتم تا ۱۵۰۰ نسل مورد آزمون قرار گرفتند و نتایج به دست آمده مربوط به میانگین ۵۰ مرتبه اجرای الگوریتم برای ۵ متغیر و ۳۰ مرتبه اجرای الگوریتم برای ۳۰ متغیر می باشد. در هر آزمایش میانگین تابع شایستگی هر نسل و بهترین مقدار شایستگی به عنوان خروجی های الگوریتم ها مورد مقایسه قرار گرفته اند.

در این مقایسه از الگوریتم CSO با مقدار تبخیر ۰/۹ استفاده شده است. همچنین در الگوریتم ژنتیک مورد استفاده برای انتخاب نسل والد از تکنیک چرخ رولت، برای آمیزش از همبری تک نقطه ای با نرخ آمیزش ۰/۹ و نرخ جهش ٪۱/۵ برای متغیر ها استفاده شده است. نتایج مربوط به مقایسه دو الگوریتم در جدول (۲) آورده شده اند.

جدول (۲): نتایج

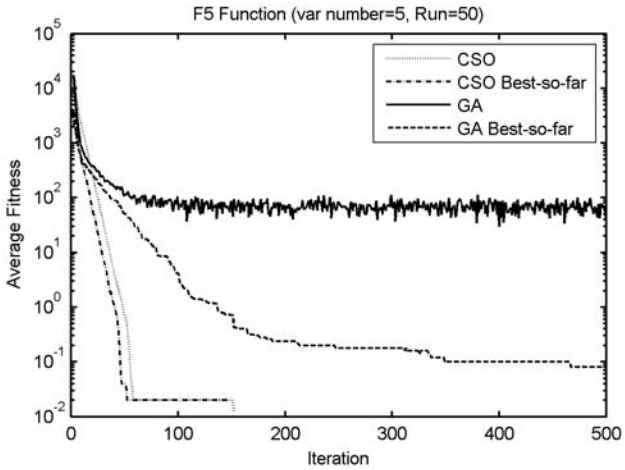
$f_{\#}$	n	GA	CSO	GA Best	CSO Best
۱	۵	۶/۵۸e+۱	۳/۱۷e-۴۲	۹/۶۱e-۵	۶/۵۸e-۴۳
۱	۳۰	۳/۹e+۲	۴/۰۸	۵/۴۵e-۴	۳/۹۹
۲	۵	۹/۵۴e-۲	۳/۲۶e-۲۲	۱/۳۹e-۳	۱/۵۶e-۲۲
۲	۳۰	۷/۵۴e-۱	۷/۴۳e-۱	۸/۵۹e-۳	۷/۴۲e-۱
۳	۵	۱/۱۸	۱/۲۲e-۲۱	۳/۴۲e-۱	۵/۵۴e-۲۲
۳	۳۰	۱/۴۸e+۱	۶/۴۵	۱/۰۴e+۱	۶/۴۴
۴	۵	۲/۲۲e+۵	۲/۹۲	۱/۸۵e+۳	۲/۹۲
۴	۳۰	۲/۰۳e+۶	۳/۸۶e+۲	۸/۴۳e+۲	۳/۸۳e+۲
۵	۵	۳/۷۸e+۱	۰	۸e-۲	۰
۵	۳۰	۴/۲۸e+۲	۵/۴۹	۰	۵/۴۳
۶	۵	۵/۵e-۱	۵/۲e-۳	۴/۴۵e-۲	۳e-۳
۶	۳۰	۱/۹۱	۸/۳۳e-۲	۱/۲۶e-۱	۶/۶۳e-۲
۷	۵	-۲/۲۹e+۳	-۲/۴۵e+۳	-۲/۳۲e+۳	-۲/۴۵e+۳
۷	۳۰	-۱/۲۷e+۴	-۱/۳۶e+۴	-۱/۲۸e+۴	-۱/۳۶e+۴
۸	۵	۶/۶۲	۶/۲۲e-۳	۶/۱۷	۶/۲۲e-۳
۸	۳۰	۵/۹۴e+۱	۱/۸۶	۵/۵۵e+۱	۱/۸۲
۹	۵	۱/۸۹	۴/۳۷e-۱۵	۱/۵۶	۴/۳۷e-۱۵
۹	۳۰	۲/۹۶	۱/۰۴	۲/۲۱	۱/۰۳
۱۰	۵	۱/۱۳	۴/۵۴e-۲	۵/۵۶e-۱	۴/۵۴e-۲
۱۰	۳۰	۳/۹	۱/۰۳	۱/۵۲e-۱	۱/۰۳

همانگونه که در جدول ۲ مشاهده می شود CSO دارای همگرایی بالایی روی توابع ۱ تا ۳ با ۵ متغیر می باشد.



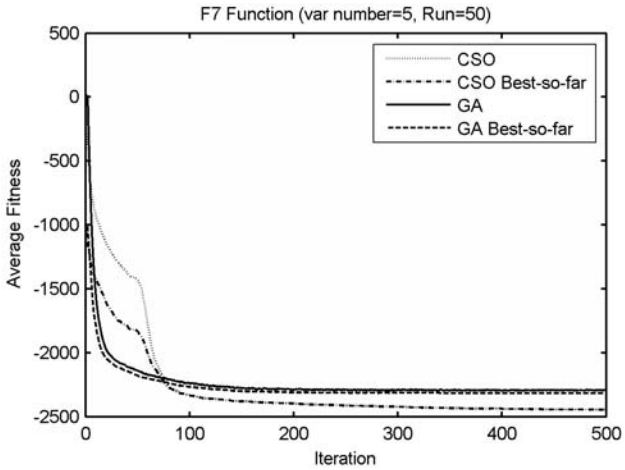
شکل (۲)

در تابع ۵ با ۵ متغیر الگوریتم CSO در نسل ۱۵۰ جواب نهایی را به دست آورده است و به آن همگرا شده است.



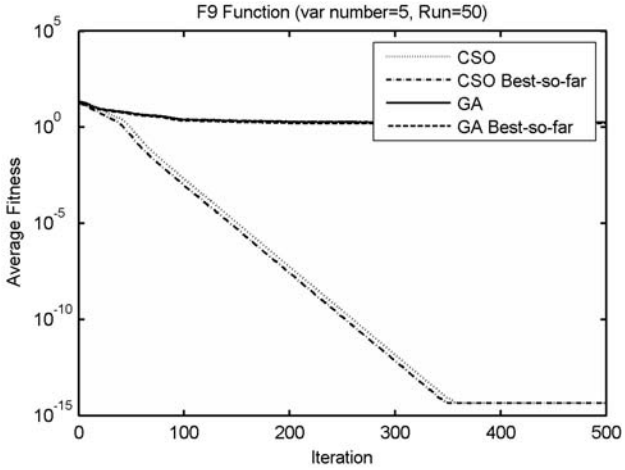
شکل (۳)

در تابع ۷ با ۵ متغیر الگوریتم ژنتیک دارای همگرایی بهتری در ۱۰۰ نسل اول دارد اما به تدریج الگوریتم CSO جوابهای بهتری را نزدیک به جواب نهایی تابع ارائه می دهد.



شکل (۴)

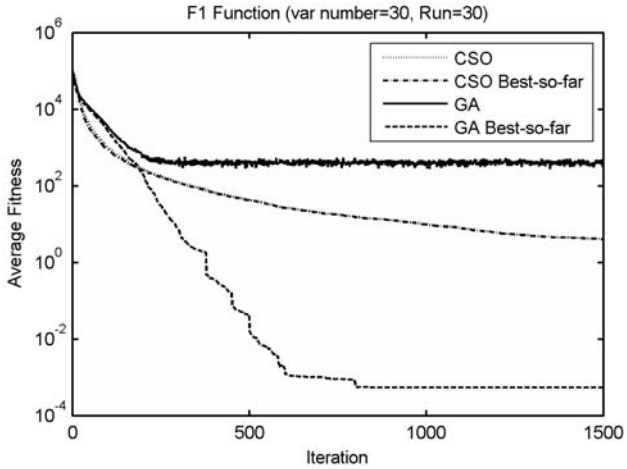
در تابع ۹ با ۵ متغیر برای الگوریتم CSO تا نسل ۳۵۰ همگرایی ادامه پیدا کرده و در این نقطه الگوریتم در یک بهینه محلی به دام افتاده است.



شکل (۵)

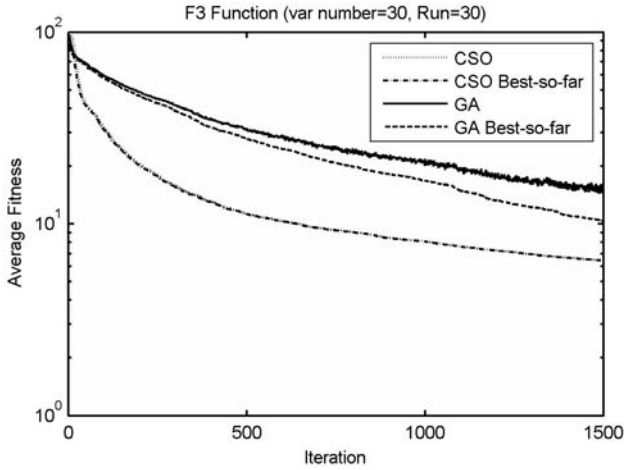
در توابع با ۳۰ متغیر همگرایی های شدید مرحله قبل برای CSO

وجود ندارد اما جوابهای میانگین به دست آمده همچنان بهتر است اما در برخی موارد بهترین جواب به الگوریتم ژنتیک بر می گردد.



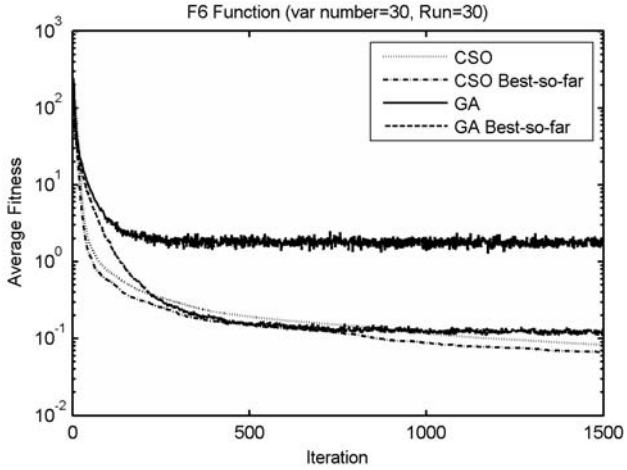
شکل (۶)

در تابع ۳ با ۳۰ متغیر هم بهترین جواب نسل و هم میانگین در الگوریتم CSO بهتر است.



شکل (۷)

در تابع ۶ با ۳۰ متغیر تقریبا بهترین جوابهای GA با میانگین جوابهای CSO هماهنگ گشته اند.



شکل (۸)

همانگونه که مشاهده می شود تفاوت بین مشخصه های بهترین و میانگین نسل ها در CSO به مراتب از GA کمتر است و این نشان دهنده این است که در الگوریتم CSO، نسل ها همگرا تر از GA به جستجو خود ادامه می دهند.

۷- نتیجه

با توجه به نتایج به دست آمده در بخش ۶ می توان عنوان کرد که الگوریتم بهینه ساز فضای منحنی در بسیاری از توابع شناخته شده عملکرد خوبی از خود نشان می دهد و با الگوریتم های معروفی همچون الگوریـتم ژنتیک قابل مقایسه می باشد. از جمله خصوصیات برتر این الگوریتم می توان به سادگی پیاده سازی آن، سرعت اجرای بالای آن، دوری از بهینه های محلی تا حد امکان و نا محدود بودن دایره انتخاب نقاط نسل بعد اشاره کرد.

سپاسگزاری

در اینجا از آقایان دکتر رضا فرهی مقدم، دکتر حسین نظام آبادی پور و دکتر سعید سریزدی که مرا در انجام این کار تشویق، یاری و راهنمایی نمودند و در مسیر تهیه مقاله و ویرایش آن از نظرات مفید خود بهره مند ساختند، تشکر می نمایم.

مراجع

[1] Wolpert, D. H., and W. G. Macready. 1997. “No free lunch theorems for optimization”. IEEE Trans. Evol. Comput. 1:67-82.
[2] Haupt, R. L., and S. E. Haupt. 2004. “Practical genetic algorithm”. Wiley Inc. Publication.
[3] Kirkpatrik, S., C. D. Gelatt Jr., and M. P. Vecchi. 1983. “Optimization by simulated annealing”. Science 220:671-680.
[4] Back, T. 1997. “Evolutionary computation: comments on the history and current state”. IEEE Trans. Evol. Comput. 1:3-17.1
[5] Holland, J. H. 1992. “Genetic algorithms”. Sci, Am. 267:66-72.
[6] Angeline, P. J. 1995. “Evolution revolution: An introduction to the special track on genetic and evolutionary programming”. IEEE Exp. Intell. Syst. Appl. 10:6-10.
[7] Adewuya, A. A. 1996. “New methods in genetic search with real-valued chromosomes”. Master’s thesis. Massachusetts Institute of Technology, Cambridge.

زیرنویس‌ها

[1] Fitness
[2] Exploration
[3] Exploitation