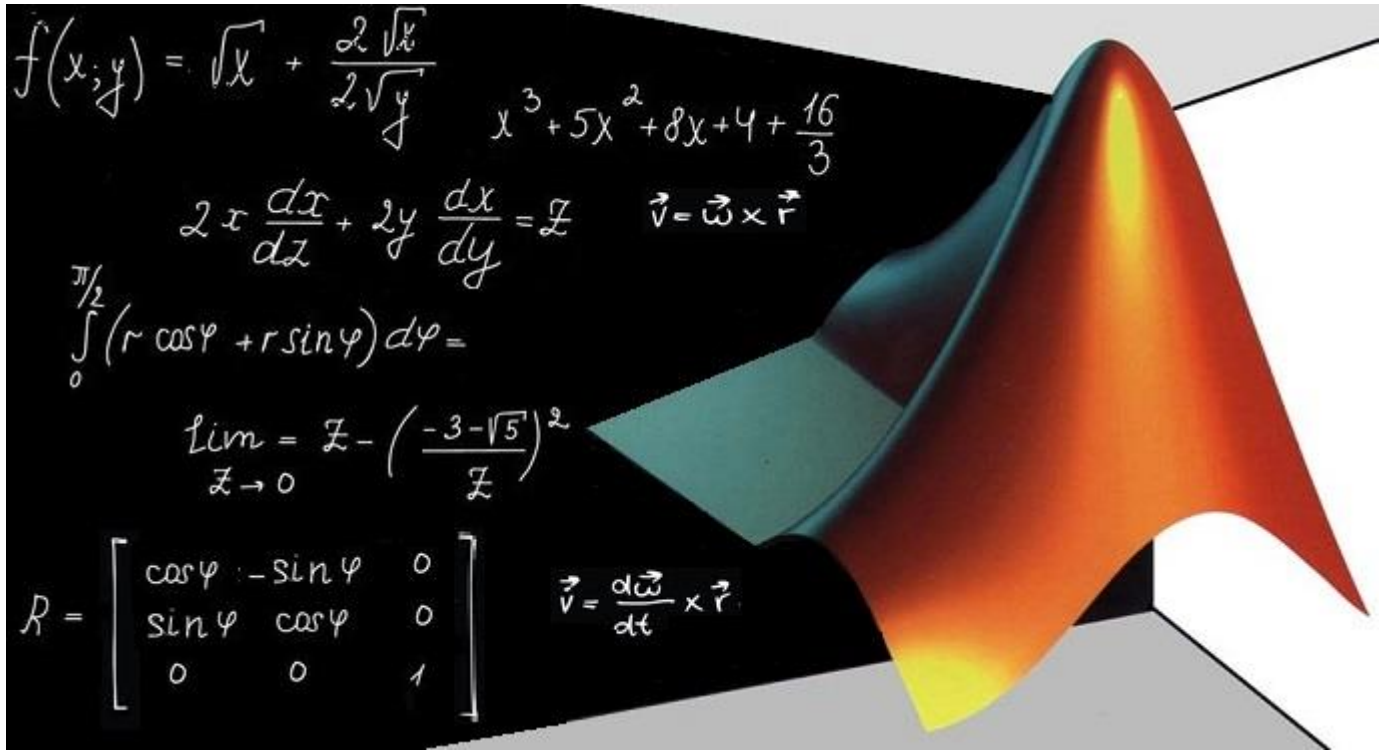
بهینه سازی پارامترهای منابع آب

آشنایی با الگوریتم بهینه‌سازی ژنتیک



نسخه 2022b

آشنایی با الگوریتم بهینه‌سازی ژنتیک

بهینه سازی در واقع ارائه بهترین نتیجه از یک مسأله تعریف شده تحت شرایط مشخص می باشد. جهت یافتن مقادیر بهینه مجهولات (متغییرهای تصمیم) روش های بهینه سازی متنوعی به تناسب نوع مدلسازی مسأله (همانند خطی، غیرخطی، با و بدون محدودیت، مسائل پیوسته و گسسته و ...) معرفی شده اند. این روش ها با وجود عملکرد مناسب، موانع و مشکلاتی نیز دارند که عبارتند از:

- یافتن جواب های بهینه محلی (Local solution) به خصوص زمانی که مقدار اولیه ارائه شده در نزدیکی بهینه محلی باشد.
- هر یک از الگوریتم های کلاسیک بهینه سازی باید متناسب با نحوه فرمول بندی تابع هدف و محدودیت ها اصلاح شوند.
- استفاده از پیش فرض هایی در خصوص طبیعت مسأله، که ممکن است برای تمامی مسائل صحیح نباشد. همانند مشتق پذیری، محدب (Convex) بودن و پیوستگی
- در صورتی که ابعاد مسأله مورد بررسی زیاد باشد، زمان اجرای الگوریتم های کلاسیک بشدت افزایش می یابد و در برخی موارد به جواب های بهینه همگرا نمی شود.

جهت غلبه بر مشکلات الگوریتم های بهینه سازی کلاسیک، رویکردهای نوینی از روش های بهینه سازی موسوم به الگوریتم‌های فراابتکاری یا فراتکاملی یا فرااکتشافی ابداع شدند که می توانند برای هر نوع مسأله بهینه سازی مورد استفاده قرار گیرند.

آشنایی با الگوریتم بهینه‌سازی ژنتیک

در واقع الگوریتم‌های فراابتکاری یا فراتکاملی یا فرااکتشافی نوعی از الگوریتم‌های تصادفی هستند که برای یافتن پاسخ بهینه نزدیک به بهینه کلی (Global solution) کاربرد دارند. با توجه به توضیحات ارائه شده می توان روش‌ها و الگوریتم‌های بهینه‌سازی را به دو دسته زیر تقسیم نمود:

✓ الگوریتم های دقیق یا تحلیلی (Exact or Analytical Algorithms)

الگوریتم‌های دقیق قادر به یافتن جواب بهینه به صورت دقیق هستند اما در مورد مسائل بهینه‌سازی سخت (مسائل با درجه غیرچندجمله‌ای) کارایی کافی ندارند و زمان اجرای آن‌ها متناسب با ابعاد مسائل به صورت نمایی افزایش می‌یابد.

✓ الگوریتم‌های تقریبی (Approximate Algorithms)

الگوریتم‌های تقریبی قادر به یافتن جواب‌های خوب (نزدیک به بهینه) در زمان حل کوتاه برای مسائل بهینه‌سازی سخت هستند. الگوریتم‌های تقریبی به سه دسته الگوریتم‌های ابتکاری (کاوشی) (Heuristic) و فراابتکاری (فراکاوشی) (Meta-heuristic) و فوق ابتکاری (Hyper-heuristic) تقسیم بندی می‌شوند.

آشنایی با الگوریتم بهینه‌سازی ژنتیک

مشکلات اصلی الگوریتم‌های ابتکاری عبارتند از:

- ✓ گیر افتادن جواب های در نقاط بهینه محلی
- ✓ همگرایی زودرس به جواب های محلی

الگوریتم‌های فراابتکاری برای حل مشکلات الگوریتم‌های ابتکاری ارائه شده‌اند. در واقع الگوریتم‌های فراابتکاری، یکی از انواع الگوریتم‌های بهینه‌سازی تقریبی هستند که دارای راهکارهای برون رفت از نقاط بهینه محلی هستند و قابلیت کاربرد در طیف گسترده‌ای از مسائل را دارند. رده‌های گوناگونی از این نوع الگوریتم در دهه‌های اخیر توسعه یافته‌است که همه این‌ها زیر مجموعه الگوریتم فراابتکاری می‌باشند.

الگوریتم های فوق ابتکاری (Hyper-heuristic) در واقع یک روش جستجوی اکتشافی است که با ترکیب روش های یادگیری ماشین (Machine learning) و چندین الگوریتم ابتکاری اقدام به بهبود فرآیند جستجوی جواب بهینه می نماید. در واقع در این الگوریتم ها بر مبنای قدرت و ضعف الگوریتم های ابتکاری و یا فراابتکاری، به گونه ای عمل می نمایند که در هر مرحله از جستجو با توجه به وضعیت مسأله مورد بررسی، الگوریتم ابتکاری متناسب با آن مورد استفاده قرار گیرد. در واقع الگوریتم های فوق ابتکاری در جهت دستیابی به جواب بهینه به طور هوشمندانه اقدام به استفاده از توانمندی سایر الگوریتم های ابتکاری و یا فراابتکاری می نمایند.

آشنایی با الگوریتم بهینه‌سازی ژنتیک

الگوریتم‌های فراابتکاری را بر اساس معیارهای زیر طبقه‌بندی می نمایند:

الف) الگوریتم های مبتنی بر یک جواب: در این الگوریتم‌ها در حین فرآیند جستجو یک جواب را تا دستیابی به مقدار بهینه تغییر می‌دهند.

ب) الگوریتم های مبتنی بر جمعیت: در الگوریتم‌های مبتنی بر جمعیت در حین جستجو، یک جمعیت از جواب‌ها مدنظر قرار می گیرد.

ج) الگوریتم های الهام گرفته شده از طبیعت و بدون الهام از طبیعت: بسیاری از الگوریتم‌های فراابتکاری از طبیعت الهام گرفته شده‌اند، در این میان برخی از الگوریتم‌های فراابتکاری نیز از طبیعت الهام گرفته نشده‌اند. الگوریتم های فراابتکاری الگوریتم هایی هستند که با الهام از طبیعت، فیزیک و انسان طراحی شده اند و در حل بسیاری از مسایل بهینه سازی استفاده می شوند. معمولاً از الگوریتم های فراابتکاری در ترکیب با سایر الگوریتم ها، جهت رسیدن به جواب بهینه یا خروج از وضعیت جواب بهینه محلی استفاده می‌گردد. در سال‌های اخیر یکی از مهمترین و امیدبخش‌ترین تحقیقات، روش‌های ابتکاری برگرفته از طبیعت بوده است. این روش‌ها شباهت‌هایی با سیستم‌های اجتماعی و یا طبیعی دارند.

د) الگوریتم های قطعی و احتمالی: یک الگوریتم فراابتکاری قطعی نظیر جستجوی ممنوعه (Tabu Search (TS))، مسئله را با استفاده از تصمیمات قطعی حل می‌کند. اما در الگوریتم‌های فراابتکاری احتمالی نظیر الگوریتم تبرید شبیه‌سازی شده (Simulated Annealing (SA))، یک سری قوانین احتمالی در حین جستجو مورد استفاده قرار می‌گیرد.

آشنایی با الگوریتم بهینه‌سازی ژنتیک

ه) الگوریتم های با حافظه و بدون حافظه: برخی از الگوریتم‌های فراابتکاری فاقد حافظه می‌باشند، به این معنا که این نوع الگوریتم‌ها از اطلاعات بدست آمده در حین جستجو استفاده نمی‌کنند (همانند الگوریتم تبرید شبیه‌سازی شده (SA)). این در حالی است که در برخی از الگوریتم‌های فراابتکاری نظیر جستجوی ممنوعه از حافظه استفاده می‌کنند. این حافظه اطلاعات بدست آمده در حین جستجو را در خود ذخیره می‌کند.

فرآیندهای حاکم بر الگوریتم های فراکاوشی

روش‌های مورد استفاده در الگوریتم های فراکاوشی برگرفته از فیزیک، زیست‌شناسی و جامعه‌شناسی هستند و از مؤلفه های زیر تشکیل شده‌اند:

- ❖ استفاده از تعداد مشخصی از سعی‌ها و کوشش‌های تکراری
- ❖ استفاده از یک یا چند عامل (نرون، ژن، کروموزوم، مورچه، زنبور، پرنده و ...)
- ❖ عملیات (در حالت چند عاملی) با یک سازوکار همکاری و رقابت
- ❖ ایجاد روش‌های خود تغییری و خود تبدیلی

آشنایی با الگوریتم بهینه‌سازی ژنتیک

این الگوریتم ها نیز مشابه طبیعت دارای دو تدبیر بزرگ در دستیابی به مقادیر بهینه می‌باشند:

الف) انتخاب پاداش برای خصوصیات فردی قوی و جزا برای فرد ضعیف‌تر

ب) جهش، که منجر به معرفی اعضای تصادفی و امکان تولد فرد جدید را میسر می‌سازد.

به طور کلی دو وضعیت وجود دارد که در روش‌های ابتکاری برگرفته از طبیعت دیده می‌شود، یکی انتخاب و دیگری جهش. انتخاب ایده‌ای مبنا برای بهینه‌سازی و جهش ایده‌ای مبنا برای جستجوی پیوسته می‌باشد.

از خصوصیات روش‌های ابتکاری برگرفته از طبیعت، می‌توان به موارد زیر اشاره کرد:

- ✓ پدیده‌ای حقیقی در طبیعت را مدل‌سازی می‌کنند.
- ✓ بدون قطع می‌باشند و تا دستیابی به هدف ادامه می یابند.
- ✓ اغلب بدون شرط، ترکیبی همانند (عامل‌های متعدد) را معرفی می‌نمایند.
- ✓ تطبیق‌پذیر هستند.

این خصوصیات باعث رفتاری معقول در جهت تأمین هوشمندی (قدرت حل مسائل مشکل) می‌شود.

آشنایی با الگوریتم بهینه‌سازی ژنتیک

ساختار کلی الگوریتم های فراکاوشی

الگوریتم‌های فراکاوشی از رویکرد مشابهی جهت استخراج مقادیر بهینه بهره می برند که به صورت مراحل زیر است:

۱- جستجوی جواب بهینه با ایجاد تعدادی جواب تصادفی (و یا یک جواب تصادفی در برخی از الگوریتم ها) در محدوده مجاز متغییرهای تصمیم: این مجموعه جواب ها در هر یک از الگویتم ها نام هایی همانند جمعیت، کلونی، گروه و ... دارند. همچنین به هر یک از این جواب ها به تنهایی اسامی مانند کروموزم، مورچه، پرنده، ذره و ... اختصاص می یابد.

۲- تولید مجموعه جواب های جدید با استفاده از عملگرها (این عملگرها عموماً بر مبنای اعداد تصادفی عمل می نمایند)

۳- انتخاب جواب هایی از میان مجموعه جواب های اولیه و جواب های جدید

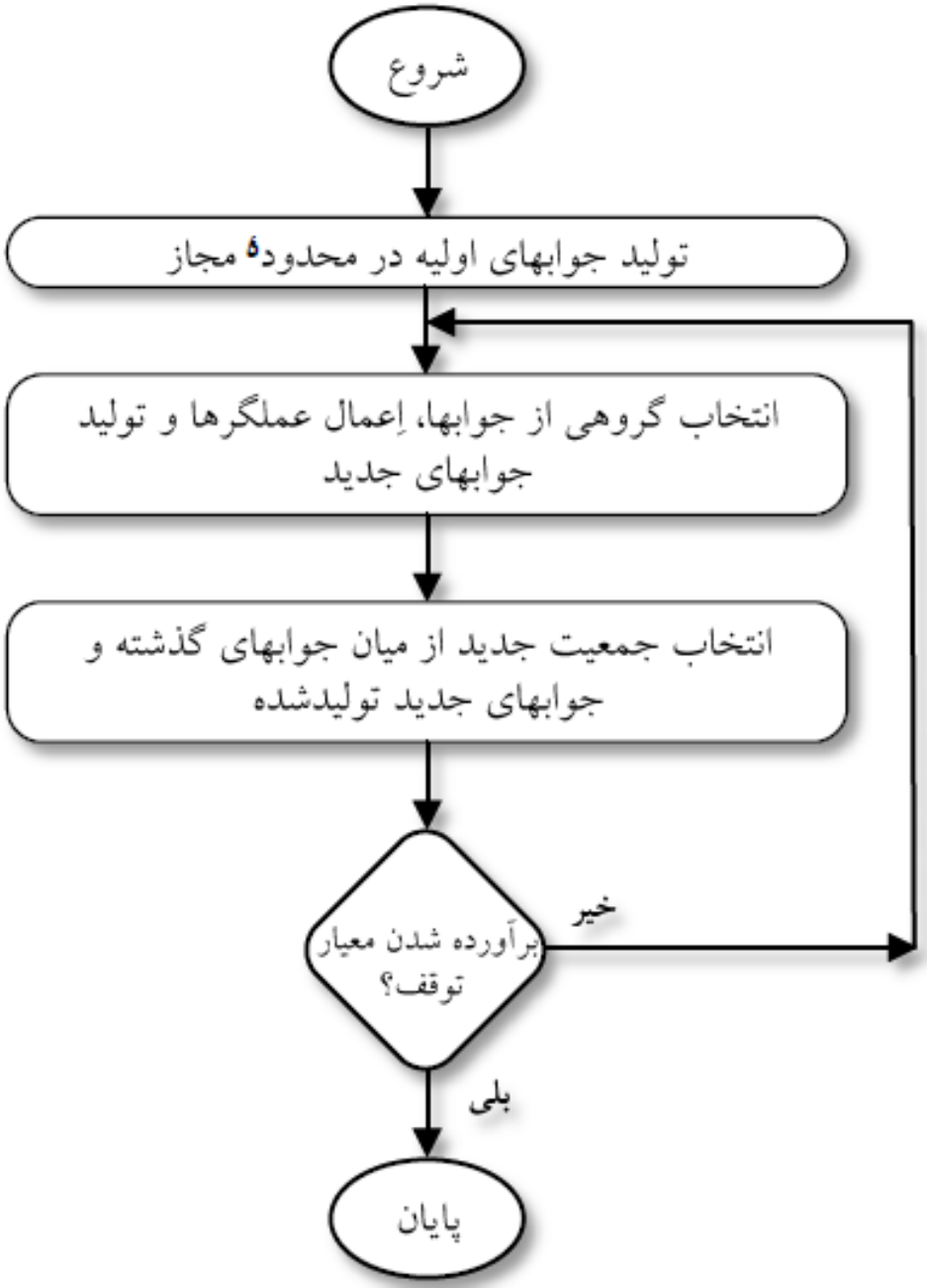
۴- تکرار مراحل ۲ و ۳ تا دستیابی به معیار توقف تعریف شده در الگوریتم بهینه سازی فراکاوشی

بنابراین هر الگوریتم فراکاوشی از دو بخش مهم تشکیل شده است:

الف) ساختار و روش عمل عملگرها جهت تولید جواب های جدید

ب) انتخاب جواب ها: این بخش بیانگر میزان هوشمندی الگوریتم است.

آشنایی با الگوریتم بهینه‌سازی ژنتیک



منظور از انتخاب جواب ها، چگونگی انتخاب تعدادی از جواب های فعلی جهت اعمال عملگرها و تولید جواب جدید است. لازم به ذکر است انتخاب در مرحله مقایسه بین جواب های گذشته و جدید اعمال می شود که به عنوان مرحله نخبه گزینی در نظر گرفته می شود. با تکرار تمرکز تولید جواب های جدید با استفاده از جواب های برتر (شایسته تر) و انتقال نسل شایسته تر به مرحله بعد، انتظار می رود که در هر مرحله، کیفیت جواب ها از نظر بهینگی نیز ارتقاء یابد.

در سال‌های اخیر الگوریتم‌های فرابتکاری جدیدی با توجه به موجودات زنده موجود در طبیعت توسعه داده شده‌اند که از معروف‌ترین آن‌ها می‌توان به الگوریتم بهینه سازی گرگ خاکستری، الگوریتم بهینه سازی سنجاقک، الگوریتم بهینه سازی گرده افشانی گل ها، الگوریتم بهینه سازی نهنگ یا وال، الگوریتم بهینه سازی ملخ و الگوریتم بهینه سازی کلونی پنگوئن های امپراتور و ... نام برد.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

در سال‌های اخیر با توجه به رشد رایانه‌ها، برای مسائل بهینه‌سازی بزرگ (با تعداد متغیرهای تصمیم زیاد) روش‌های جدید تدوین و پیشنهاد شده است. یکی از بهترین روش‌های بهینه‌سازی جدید، الگوریتم ژنتیک می‌باشد. الگوریتم ژنتیک از تکنیک‌های بهینه‌سازی نسبتاً جدیدی است که برای حل مسائل بزرگ با تابع یا محدودیت‌های غیرخطی بسیار مناسب می‌باشد. الگوریتم ژنتیک با ایده گرفتن از فرآیند تکامل موجودات و با توجه به واقعیت‌های زیر توسعه داده شده است:

- ✓ فرزندان دارای ترکیبی از خصوصیات ژنتیکی والدینشان هستند.
- ✓ همواره موجودات قوی تر شانس بیشتری برای بقا و تولید مثل دارند.
- ✓ احتمال اینکه نسل‌های جدید که از والدین برتر (مثلاً قوی‌تر) ایجاد می‌شوند، بهتر از نسل‌های قبلی باشند بیشتر از این است که فرزندان ضعیف‌تری ایجاد شوند.
- ✓ فرزندان حاصل از والدین ضعیف ممکن است قوی باشند. بنابراین برای تولید فرزندان قوی‌تر لازم است تا حدودی به والدین ضعیف نیز امکان بقا داده شود.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

الگوریتم ژنتیک اولین بار توسط Holland در سال ۱۹۷۵ پیشنهاد گردید. این تکنیک بر اصل تکامل داروین استوار است. بر پایه مکانیزم طبیعی انتخاب و تولید مثل، Holland یک ساختار هوشمند برای جستجوی تصادفی ایجاد نمود که می‌توانست در مجموعه جواب، یک نواحی محتمل را برای جواب‌های بهینه پیدا کند و در این نواحی به جستجو بپردازد. در الگوریتم ژنتیک ضمانتی برای اینکه نسل تولید شده از نسل قبل بهتر باشد، داده نمی‌شود ولی همواره احتمال تولید جواب‌های بهتر، بیشتر از تولید جواب‌های بدتر است.

Holland در سال ۱۹۷۹ با توجه به نکات فوق، روش الگوریتم ژنتیک را برای مسائل بهینه‌سازی پایه‌گذاری کرد. این روش در دو دهه بعد از ارائه الگوریتم، چندان مورد استقبال قرار نگرفت ولی در سال‌های اخیر با رشد قابلیت رایانه‌ها و نیاز بشر به حل مسائل پیچیده‌تر، مورد توجه ویژه محققین قرار گرفته است.

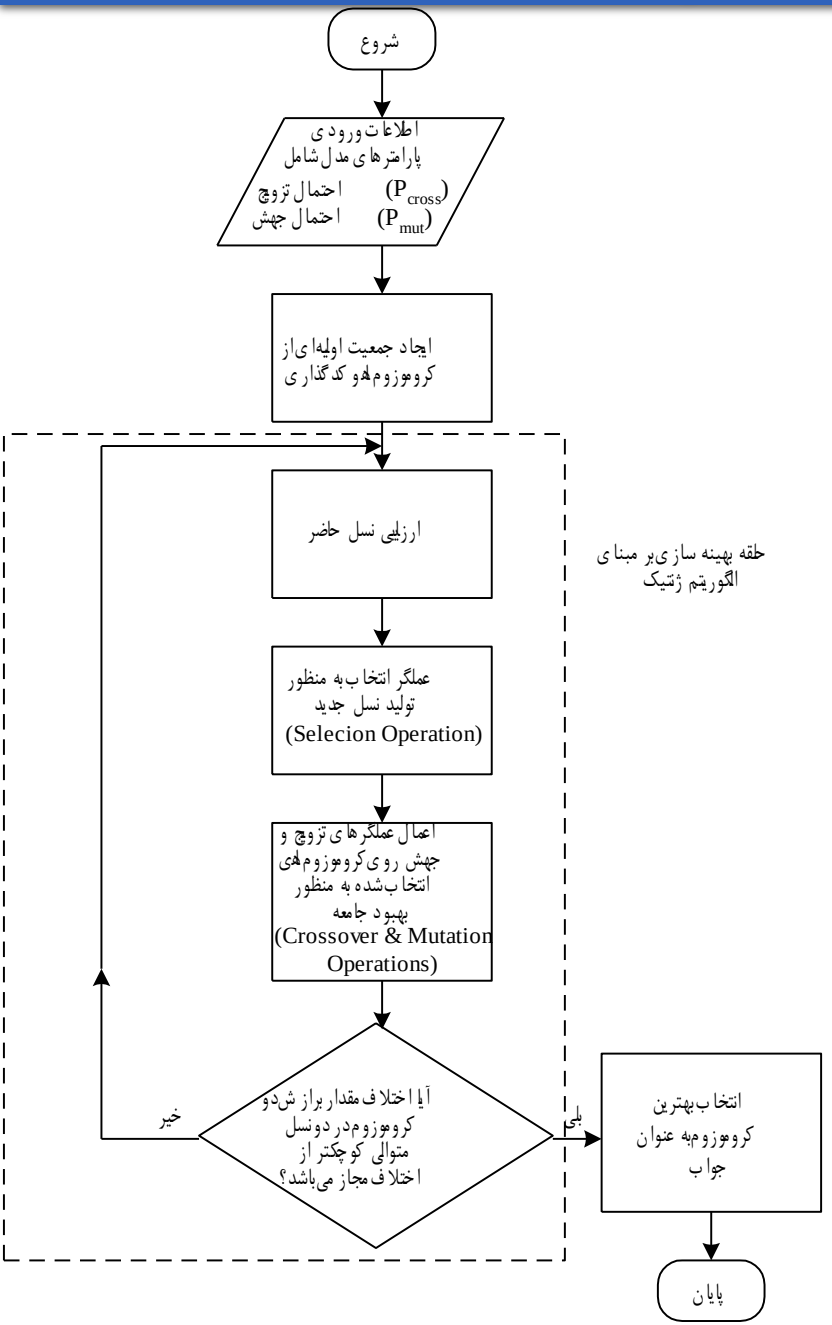
در الگوریتم‌های ژنتیک هر جواب یک کروموزوم (Chromosome) و هر مجموعه‌ای از جواب‌ها یک جامعه یا نسل (Population) نامیده می‌شود. هر کروموزوم مجموعه‌ای از پارامترهاست که ژن (Gene) نامیده می‌شوند و مقادیر آنها باید تعیین شود. عملکرد هر جواب (کروموزم)، برازش (Fitness) یا تطابق کروموزم نامیده می‌شود.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

مراحل انجام محاسبات در الگوریتم ژنتیک کلاسیک به صورت زیر می‌باشد:

1. متغیرهای تصمیم مسأله لازم است در کنار یکدیگر قرار داده شده و یک کروموزم را تشکیل دهند. به هر یک از متغیرهای تصمیم در کروموزوم یک ژن می‌گویند.
2. ژن‌های مسأله می‌توانند بسته به مشخصات مسأله به صورت اعداد حقیقی باشند یا کدگذاری شوند.
3. به صورت تصادفی مجموعه‌ای از جواب‌های مسأله تحت عنوان جمعیت اولیه (نسل اولیه) تولید می‌شود.
4. محاسبه میزان تابع هدف به ازای هر یک از اعضای جامعه (میزان تابع هدف را می‌توان تحت عنوان برازش هر کروموزوم در نظر گرفت).
5. انتخاب مجموعه‌ای از کروموزوم‌های برتر (کروموزوم‌های والد)، از جواب‌های قابل قبول مسأله، برای تولید نسل بعد
6. اعمال عملگر تزویج برای تولید کروموزوم‌های اولیه نسل بعد از کروموزوم‌های والد نسل قبل
7. تولید کروموزوم‌های نهایی نسل جدید با اعمال عملگر جهش بر روی کروموزوم‌های اولیه نسل جدید
8. تکرار مراحل انتخاب تا اعمال کروموزوم جهش
9. بررسی همگرایی روش (شرط توقف)

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن



مراحل اجرای الگوریتم ژنتیک کلاسیک را می توان در قالب فلوچارت مقابل ارائه نمود.

منطق حاکم بر الگوریتم‌های ژنتیک بسیار ساده است. ابتدا روشی که جواب‌های امکان‌پذیر را به ساختاری که الگوریتم ژنتیک با آن کار خواهد کرد، تبدیل می‌کند.

سپس یک مجموعه اولیه از جواب‌ها برای مسأله ایجاد می‌شود. در مرحله بعد چند جواب برتر انتخاب شده و از ترکیب آنها با استفاده از عملگرهای ویژه‌ای که جابجایی ژن‌ها و جهش نامیده می‌شوند یک مجموعه جدید از جواب‌ها ایجاد می‌شود. فرآیند انتخاب معمولاً فرآیندی تصادفی است ولی شانس بیشتری را به انتخاب جواب‌های بهتر می‌دهد.

عملگرهای جابجایی ژن‌ها و جهش زمانی برای کروموزم‌های انتخاب شده به‌کار گرفته می‌شوند که چندین آزمون احتمالاتی با موفقیت سپری شود. اگر هیچ یک از این آزمونها سپری نشود، کروموزوم های انتخاب شده بدون تغییر در جامعه جدید کپی می‌شوند که به آن تولید مجدد می‌گویند.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

به طور کلی بهینه‌سازی توسط الگوریتم ژنتیک توسط سه مرحله اساسی زیر صورت می‌پذیرد:

انتخاب (Selection)

منظور از انتخاب والدین، در واقع ایجاد امکان تولید مثل به اعضایی از جامعه است که عملکرد بهتری داشته‌اند. برای این منظور روش‌های مختلفی وجود دارد. در بین روش‌های مختلف انتخاب، روش های چرخ گردان (Roulette wheel) و تورنامنت (Tournament) نسبت به روش‌های دیگر بهتر عمل می‌نماید.

در این روش تعدادی از کروموزم‌های جامعه انتخاب شده و از بین آنها کروموزمی که بهترین برازش را دارد، استخراج می‌گردد. عملکرد موفق این روش انتخاب در مسئله پایش آبهای زیرزمینی، مدیریت کیفی رودخانه و بهره‌برداری چند هدفه مخزنه مشخص گردیده است. در این روش به طور تصادفی تعدادی از کروموزم‌های متوالی جامعه انتخاب می‌گردد و کروموزمی که بهترین مقدار تابع هدف (برازش) را از بین این کروموزم‌ها دارا می‌باشد، انتخاب می‌گردد. این عملیات به تعداد کروموزم‌های جامعه تکرار می‌شود تا یک جامعه از کروموزم‌های والد (Parents) ایجاد گردند. در این روش، کروموزم‌های برتر احتمال انتخاب بیشتری خواهند داشت.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

انتخاب کروموزوم‌های والد به روش چرخ گردان

در این روش انتخاب سعی می‌شود علاوه بر حفظ بهترین کروموزوم‌های نسل حاضر، ترکیبی از کروموزوم‌های برتر برای تولید نسل بعد انتخاب گردند. گام‌های اصلی روش به شرح زیر می‌باشد:

- ✓ میزان تابع هدف به ازای هر کروموزوم $\boldsymbol{j}$ (برازش هر کروموزوم) محاسبه می‌شود ($\boldsymbol{F(j)}$)
- ✓ میزان احتمال انتخاب هر یک از مقادیر برازش کروموزم‌ها به صورت رابطه مقابل محاسبه می‌شود:

$$p(j) = \frac{F(j)}{\sum_{i=1}^{p_{size}} F(i)}$$

- ✓ با توجه به موقعیت ظاهری کروموزوم‌ها در جامعه، میزان احتمال تجمعی $\boldsymbol{a_j}$ را برای هر کروموزوم $\boldsymbol{j}$ در جامعه به صورت زیر محاسبه می شود:

$$a_j = \sum_{1}^{j} p(j)$$

- ✓ یک عدد تصادفی ($\boldsymbol{r}$) با توزیع یکنواخت بین صفر ویک تولید می‌گردد. اگر $\boldsymbol{a_{p-1} \leq r \leq a_p}$ باشد، $\boldsymbol{p}$امین کروموزم انتخاب می شود.

مراحل فوق به تعداد اعضاء جامعه تکرار می‌گردد به طوری که کروموزم‌های والد نسل بعد به طور کامل انتخاب شوند.

در صورتی که بهترین کروموزوم نسل جاری انتخاب نشده باشد این کروموزم به طور تصادفی جایگزین یکی از کروموزم‌های والد انتخاب شده می‌گردد. این موضوع باعث می‌شود تا حداقل بهترین کروموزوم نسل جاری در نسل بعد هم حضور داشته باشد و از تخریب بهترین کروموزوم‌های نسل جاری جلوگیری شود.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

مراحل فوق باعث می‌شود تا احتمال انتخاب کروموزوم‌های برتر بیشتر باشد زیرا این کروموزوم‌ها منجر به ایجاد تغییرات بزرگی در تابع احتمال تجمعی پذیرش می‌شوند و احتمال این که اعداد تصادفی در بازه مربوط به آنها قرار گیرد و حتی این کروموزوم‌ها چندین بار انتخاب شوند زیاد است. بنابراین در این روش انتخاب، کروموزوم‌های برتر با احتمال بیشتری انتخاب می‌گردند و لذا الگوریتم به سمت جواب‌های برتر حرکت می‌کند. از طرف دیگر این روش انتخاب، جواب‌های نامناسب را به طور کامل حذف نمی‌کند بلکه ناحیه جستجوی جواب‌های بهینه وسیع بوده و در صورت برخورد با جواب های موضعی، حرکت به سوی جواب های بهینه ادامه خواهد یافت.

Fitness_Function (Chromosome1) = 1
Fitness_Function (Chromosome2) = 4
Fitness_Function (Chromosome3) = 3
Fitness_Function (Chromosome4) = 2

از آنجایی احتمال انتخاب کروموزوم اول برابر ۰/۱ است. در نتیجه بازه ۰ تا ۰/۱ را به کروموزوم یک نسبت می دهیم.

همچنین احتمال انتخاب کروموزوم دوم برابر ۰/۴ است. در نتیجه بازه ۰/۱ تا ۰/۵ را به کروموزوم دو نسبت می دهیم.

احتمال انتخاب کروموزوم سوم برابر ۰/۳ است. در نتیجه بازه ۰/۵ تا ۰/۸ به کروموزوم سوم اختصاص می یابد.

احتمال انتخاب کروموزوم چهارم برابر ۰/۲ است. در نتیجه بازه ۰/۸ تا ۱ را به کروموزوم چهارم نسبت می دهیم.

Probability (chromosomes C1) = Fitness(chromosomes C1) / Sum Fitness(All chromosomes) = 1/10 = 0.1
Probability (chromosomes C2) = 4/10 = 0.4
Probability (chromosomes C3) = 3/10 = 0.3
Probability (chromosomes C4) = 2/10 = 0.2

با تولید یک عدد تصادفی بین ۰ تا ۱ و مقایسه با بازه های ارائه شده فوق، می توان کروموزم منتخب را استخراج نمود. این عدد تصادفی در هر بازه ای قرار بگیرید یعنی آن کروموزوم انتخاب شده است. به عنوان مثال اگر عدد تصادفی ۰/۲۵ باشد، چون بین ۰/۱ تا ۰/۵ است در نتیجه کروموزوم شماره دو انتخاب می شود.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

انتخاب کروموزوم‌های والد به روش تورنامنت

روش تورنامنت یکی از روش‌های ساده ولی پرکاربرد انتخاب کروموزوم‌های والد است. در این روش بسته به بزرگی جمعیت کروموزوم‌ها در یک نسل، جمعیت موجود به چندین دسته تقسیم شده و سپس به طور تصادفی از بین دسته‌های موجود یک دسته انتخاب می‌گردد و بهترین کروموزوم آن به عنوان یکی از کروموزوم‌های والد نسل بعد انتخاب می‌گردد. با انتخاب یک کروموزوم، تعداد کروموزم جامعه جاری تغییری نمی‌نماید و عملاً امکان انتخاب یک کروموزم به دفعات وجود دارد.

در روش تورنامنت انتخاب تعداد دسته‌ها از اهمیت ویژه‌ای برخوردار است. تعداد دسته‌های کم، موجب کاهش شدید تنوع کرموزوم نسل‌ها خواهد شد و احتمال همگرایی به جواب‌های موضعی را به شدت افزایش خواهد داد. از طرف دیگر تعداد زیاد دسته‌ها نیز می‌تواند سرعت همگرایی روش به جواب‌های بهینه را به شدت کاهش دهد.

برتری اصلی این روش نسبت به روش چرخ گردان، انعطاف پذیری آن است به طوری که حتی طی فرآیند محاسبات و با توجه به مشخصات کروموزوم‌های تولید شده، الگوریتم می‌تواند خود را اصلاح کند و تعداد دسته‌های مناسب را برای تولید نسل‌های بعد انتخاب نماید.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

ارزیابی برازش و انتخاب کروموزم‌های برتر

ارزیابی هر کروموزوم بر اساس مقدار تابع هدف، به ازای مقادیر ژن‌های آن صورت می‌گیرد. برای بهبود تنوع کروموزوم‌ها در یک نسل، قبل از انتخاب کروموزوم‌های برتر، می‌توان توابع برازش را اصلاح نمود. یکی از روش‌هایی که کاربرد مؤثری در جلوگیری از همگرایی کروموزوم‌های یک نسل به یک یا چند کروموزوم داشته است، روش طبقه‌بندی می‌باشد که توسط Nafpliotis (1993) و Horn and Mahfoud (1995) پیشنهاد شده است. در این روش سعی می‌شود به نوعی مقادیر برازش کروموزوم‌ها اصلاح شوند تا همگرایی آنها به سمت بهترین کروموزوم‌های موجود کاهش داده شود. در روش طبقه‌بندی، تابع برازش محاسبه شده برای هر کروموزم به صورت زیر اصلاح می‌گردد:

$$\text{برازش اصلاح شده کروموزم } n = \frac{\text{برازش اولیه کروموزم } n}{\sum_{j=1}^{npop} Sh(d_{jn})} \qquad if \sum_{j=1}^{npop} Sh(d_{jn}) > 0$$

در این رابطه، $npop$: تعداد کروموزم های هر نسل، d_{jn}: شاخصی از اختلاف بین دو کروموزم j و n است که به صورت زیر محاسبه می شود:

$$d_{jn} = \frac{\sqrt{\sum_{i=1}^{ngens}\left(\left(P_{ij} - P_{in}\right)/\left(P_{max_i} - P_{min_i}\right)\right)^2}}{ngens} \quad , \qquad \forall\, j \in pop$$

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

$$d_{jn} = \frac{\sqrt{\sum_{i=1}^{ngens}\left(\left(P_{ij} - P_{in}\right)/\left(P_{max_i} - P_{min_i}\right)\right)^2}}{ngens} \quad , \quad \forall\, j \in pop$$

P_{ij}: مقدار ژن i در کروموزم j

P_{in}: مقدار ژن i در کروموزم n

$$Sh(d_{jn}) = \begin{cases} 1 - d_{jn}/L_{min} & if \quad d_{jn} < L_{min} \\ 0 \quad , & otherwise \end{cases} \quad , \quad \forall\ j,n$$

P_{max_i} و P_{min_i}: مقادیر حداکثر و حداقل ژن i در جامعه

$ngens$: تعداد ژن های هر کروموزم

L_{min}: ثابتی است که اندازه طبقات را کنترل می‌نماید. هرچه مقدار آن کوچکتر باشد میزان تغییر در مقادیر برازش کروموزوم‌ها کمتر است.

پس از تعیین مقادیر اصلاح شده برازش کروموزوم‌ها لازم است کروموزوم‌های والد برای نسل بعد با استفاده از روش های چرخ گردان و یا تورنامنت انتخاب گردند.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

عملگر تزویج (Crossover)

عملگرهای تولید مثل، که عملگرهای تزویج و جهش نامیده می‌شوند، از کروموزوم‌های والد انتخاب شده، کروموزم‌های نسل جدید را ایجاد می‌نمایند. عملگر تزویج با جابجایی ساختار ژنتیکی دو کروموزوم والد، کروموزوم‌های فرزند را به صورتی تولید می‌نماید که دارای مشخصات ژنتیکی مثبت یا منفی کروموزوم‌های والد باشند. از آنجا که کروموزوم‌های والد معمولاً کروموزوم‌های برتر نسل جاری از نظر میزان برازش هستند، کروموزم‌های فرزند نیز با احتمال خیلی زیاد این خصوصیت مثبت را از کروموزوم‌های والد به ارث خواهند برد.

Michalewicz (1992) کارایی روش‌های تزویج تک نقطه‌ای، دو نقطه‌ای و یکنواخت را تشریح نمود. در سال‌های اخیر نیز روش‌های تزویج متنوعی توسط محققان مختلف پیشنهاد شده‌اند ولی هنوز راهکار کاملاً برتری معرفی نشده است. انجام عمل تزویج بر روی کروموزوم‌های والد با احتمال P_C (این مقدار معمولاً بین ۰/۶ تا ۰/۹ می باشد) صورت می‌گیرد. برای این منظور برای هر زوج کروموزوم انتخاب شده برای عمل تزویج، یک عدد تصادفی با توزیع یکنواخت r تولید شده و با مقدار احتمال تزویج P_C مقایسه می‌گردد. در صورتی که r کوچکتر از احتمال تزویج باشد، عمل تزویج انجام می‌گیرد، در غیر این صورت کروموزم‌های والد به همان شکل برای انجام عمل جهش در نظر گرفته می‌شوند.

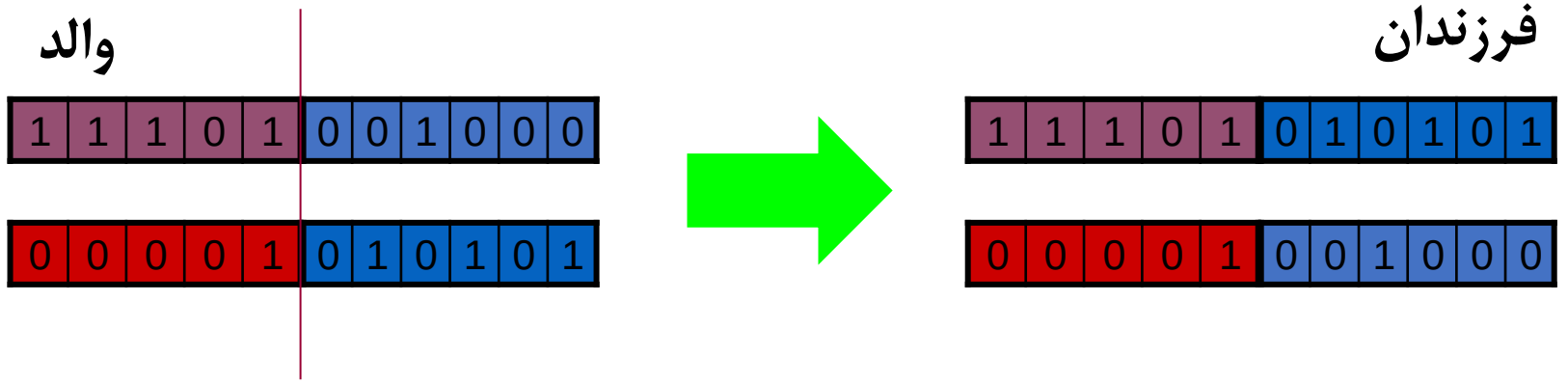
معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

عملگر تزویج تک نقطه ای (1-Point Crossover)

در این روش با توجه به طول کروموزوم‌ها، یک محل برش به طور تصادفی انتخاب می‌گردد و محتویات ژن‌های کروموزوم‌ها در بعد از نقطه برش با یکدیگر جابجا می‌شوند. محل برش حتماً باید در بین ژن‌ها قرار داشته باشد و در بین بیت‌های یک ژن قرار نمی‌گیرد. به عبارت دیگر در صورتی که $x_i\ (i = 1, \dots, n)$ و $y_i\ (i = 1, \dots, n)$ نشان‌دهنده ژن‌های مرتب شده دو کروموزوم والد $\boldsymbol{x}$ و y و محل برش که به صورت تصادفی تولید شده است برابر با $\boldsymbol{r}$ باشند، کروموزوم‌های فرزند $(\acute{x}, \acute{y})$ به صورت زیر خواهند بود:

$$x_i' = \begin{cases} x_i & i \le r \\ y_i & otherwise \end{cases}$$

$$y_i' = \begin{cases} y_i & i \le r \\ x_i & otherwise \end{cases}$$



نکته: اگر تعداد کروموزم‌های تزویج فرد باشد آنگاه بطور تصادفی یکی از کروموزم‌های تزویج از آن خارج می‌گردد و برای عملیات جهش باقی می‌ماند.

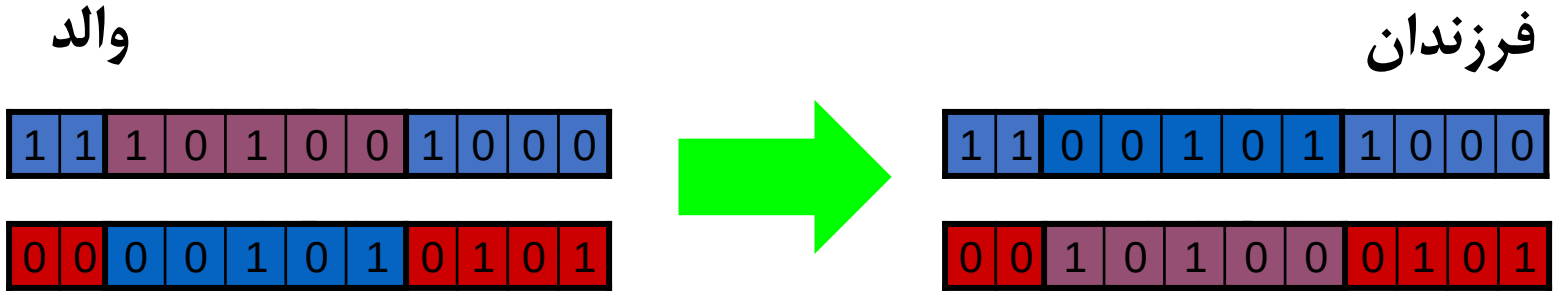
معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

عملگر تزویج دو نقطه ای (2-Point Crossover)

در این روش با توجه به طول کروموزوم‌ها، دو محل برش به طور تصادفی انتخاب می‌گردد و محتویات ژن‌های کروموزوم‌ها در قبل از نقطه برش اول و بعد از نقطه برش دوم با یکدیگر جابجا می‌شوند. به عبارت دیگر در صورتی که $x_i\ (i=1,\dots,n)$ و $y_i\ (i=1,\dots,n)$ نشان‌دهنده ژن‌های مرتب شده دو کروموزوم والد $\boldsymbol{x}$ و $\boldsymbol{y}$ و محل های برش که به صورت تصادفی تولید شده است برابر با $\boldsymbol{r}_1$ و $\boldsymbol{r}_2$ باشند، کروموزوم‌های فرزند $(\acute{x},\acute{y})$ به صورت زیر خواهند بود:

$$x_i' = \begin{cases} y_i & i \le r_1 \\ x_i & r_1 \le i \le r_2 \\ y_i & i \ge r_2 \end{cases}$$

$$y_i' = \begin{cases} x_i & i \le r_1 \\ y_i & r_1 \le i \le r_2 \\ x_i & i \ge r_2 \end{cases}$$



نکته: می توان ژن های واقع در بین دو برش را نیز جابجا نمود و ژن های قبل و بعد از برش را ثابت نگه داشت. لازم به ذکر است عملگرهای تزویج بیش از دو نقطه تا n نقطه نیز در مطالعات قابل استفاده است.

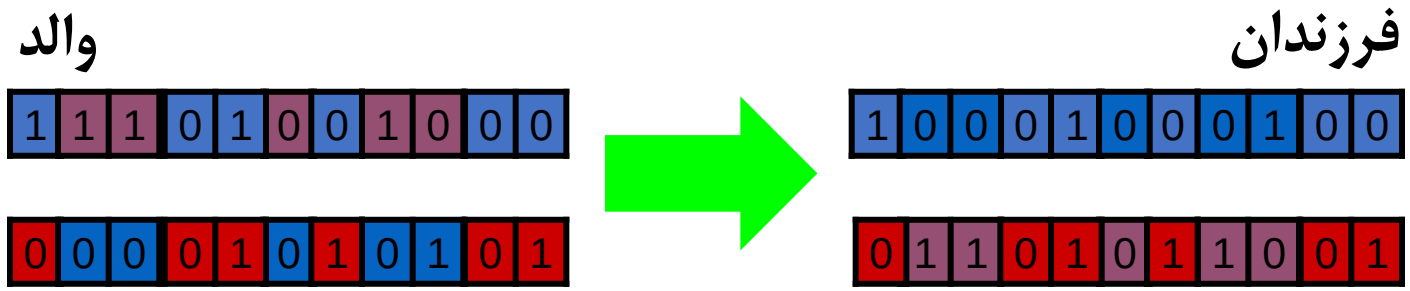
معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

عملگر تزویج یکنواخت (Uniform Crossover)

در این روش که حالت تعمیم داده شده حالات قبل می‌باشد، به جای اینکه محل‌های برش به صورت تصادفی تعیین شوند برای هر ژن i یک عدد تصادفی با توزیع یکنواخت بین صفر و یک تولید می‌گردد و با توجه به مقدار این عدد تصادفی در مورد جابجایی آن ژن بین کروموزوم‌های والد تصمیم‌گیری می‌شود. به عبارت دیگر در صورتی که $x_i\ (i = 1, \dots, n)$ و $y_i\ (i = 1, \dots, n)$ نشان‌دهنده ژن‌های مرتب شده دو کروموزوم والد $\boldsymbol{x}$ و $\boldsymbol{y}$ و همچنین $\boldsymbol{r_i}$ عدد تصادفی تولید شده برای ژن i در کروموزوم‌های والد باشد، کروموزوم‌های فرزند $(\acute{x}, \acute{y})$ به صورت زیر خواهند بود:

$$x_i' = \begin{cases} x_i & r_i \le 0.5 \\ y_i & r_1 > 0.5 \end{cases}$$

$$y_i' = \begin{cases} y_i & r_i \le 0.5 \\ x_i & r_1 > 0.5 \end{cases}$$



معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

عملگر جهش (Mutation)

هدف از استفاده از این عملگر، ایجاد برخی تغییرات در کروموزم‌هاست تا حالات مختلف در نظر گرفته شوند. و از دستیابی به جواب‌های بهینه غیر محلی اطمینان حاصل شود. جهش شامل ایجاد یک تغییر تصادفی در اطلاعات یک ژن می‌باشد. عملگر جهش دارای یک پارامتر است که آستانه احتمال P_{mut} نامیده می‌شود. احتمال اینکه یک متغیر تصمیم در هر گام تکاملی تغییر نماید برابر با P_{mut} در نظر گرفته می‌شود. هر چه این عدد کوچکتر باشد احتمال تغییر متغیرهای تصمیم طی فرآیند تکاملی کاهش می‌یابد. معمولاً این مقدار کاهش می‌یابد تا جستجوهای الگوریتم ژنتیک از کل محدوده جواب‌های امکان‌پذیر به سمت نواحی با جواب‌های بهتر تمایل پیدا می‌کند. این کاهش مقدار آستانه احتمال نباید آنقدر باشد که میزان همگرایی ناشی از فرآیند انتخاب و تزویج را تشدید نماید.

در عملگر جهش ابتدا یک عدد تصادفی برای هر ژن تولید می‌گردد اگر عدد تصادفی کوچکتر از P_{mut} باشد، جهش برای آن ژن از کروموزم اتفاق می‌افتد. عملگر جهش برای ژن‌های کلیه کروموزم‌ها بکار برده می‌شود. عملیات جهش از آن جهت حائز اهمیت است که گوناگونی در جمعیت رشته‌ها را به منظور جلوگیری از همگرایی قبل از موقع و بدست آمدن جواب غیر بهینه، باعث می‌شود.

نکته: مقدار P_{mut} به طور معمول بین $1/$(طول کروموزم) و $1/$(اندازه نسل) در نظر گرفته می شود.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

الگوریتم ژنتیک برای تعداد مناسبی نسل تا رسیدن به همگرایی انجام می‌شود. برای تعداد نسل‌ها هیچگونه قانون یا راهنمایی وجود ندارد. تعداد نسل‌ها باید آنقدر زیاد باشد که با اجرای مدل، جواب بهینه غیر محلی حاصل شود و از طرفی نباید آنقدر زیاد باشد که مدت زمان اجرای برنامه را بی‌دلیل طولانی نماید.

نحوه اجرای الگوریتم ژنتیک با استفاده از دستورات MATLAB

جهت اجرای یک مدل بهینه سازی خطی و یا غیرخطی با و بدون محدودیت با استفاده از الگوریتم فراکاوشی ژنتیک از دستور ga به صورت زیر استفاده می شود:

```
[x,fval,exitflag,output,population,scores]=ga(fun,nvars,A,b,Aeq,beq,lb,ub,nonlcon,IntCon,options)
```

ورودی های دستور ga به شرح زیر می باشند:

fun: نام تابع هدف خطی و یا غیرخطی است که با استفاده از تابع function و یا با تعریف آن توسط @ می تواند مشخص شود. لازم به ذکر است این تابع باید یک بردار به طول تعداد متغییر تصمیم (nvars) بپذیرد و اقدام به محاسبه مقدار تابع هدف نماید. به عنوان مثال:

```
fun = @(x)(x(1:3)-x(4:6)-[1 3 5]).^2
```

در این تابع هدف ۶ متغییر تصمیم (۳ تا برای x1 و ۳ تا برابر x2) به عنوان ورودی لحاظ می شود.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

nvars: تعداد متغییرهای تصمیم است که یک عدد صحیح مثبت می باشد.

A: ماتریس ضرایب نامعادلات با ابعاد M (تعداد نامعادلات) در nvars (تعداد متغییرهای تصمیم) حاوی اعداد حقیقی.

$A*x <= b$

b: بردار ستونی با M ستون که نشان دهنده اعداد ثابت سمت راست نامعادلات می باشد.

$$x_1 + 2x_2 \leq 10$$
$$3x_1 + 4x_2 \leq 20$$
$$5x_1 + 6x_2 \leq 30,$$

```
A = [1,2;3,4;5,6];
b = [10;20;30];
```

جهت نمایش محدودیت $x_1 + x_2 + x_3 + \cdots + x_N \leq 1$ از ماتریس های زیر استفاده می شود:

```
A = ones(1,N)
b = 1
```

Aeq: ماتریس ضرایب محدودیت های تساوی (معادلات) با ابعاد Me (تعداد معادلات) در nvars (تعداد متغییرهای تصمیم) حاوی اعداد حقیقی و beq: بردار ستونی با Me ستون بیانگر مقادیر ثابت سمت راست معادلات است.

$Aeq*x = beq$

$$x_1 + 2x_2 + 3x_3 = 10$$
$$2x_1 + 4x_2 + x_3 = 20,$$

```
Aeq = [1,2,3;2,4,1];
beq = [10;20];
```

نکته: در صورتی که بردارهای b و beq به صورت ردیفی ارائه شوند، این دستور از شکل ستونی آن ها استفاده می کند و مشکلی در روند اجرای برنامه ایجاد نمی شود.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

lb و ub: بردار و یا ماتریس از اعداد حقیقی که به ترتیب نشان دهنده حدود پایین و بالای هر متغییر تصمیم می باشد. اگر تعداد اعداد واقع در این ماتریس ها کمتر از متغییرهای تصمیم باشد، برنامه با warning مواجه می شود. در صورتی که متغییرهای تصمیم بزرگتر از صفر باشند، می توان از دستور lb = zeros(nvars,1) جهت تعیین حد پایین آن ها استفاده نمود. همچنین در صورتی که متغییرهای تصمیم کوچکتر از یک عدد مشخصی (a) باشند از دستور ub = a.*ones(nvars,1) جهت تعیین حد بالای آن ها استفاده می شود.

$x_1 \geq 0 , x_3 \geq 3$ ⟹ lb = [0;-Inf;3] $x_2 \leq 4 , x_3 \leq 1$ ⟹ ub = [Inf;4;1]

Nonlcon: جهت ارائه محدودیت های غیرخطی نامساوی و مساوی کاربرد دارد. برای این منظور لازم است با تعریف تابعی مشخص، محدودیت های نامساوی در یک پارامتر (به عنوان مثال C) و محدودیت های تساوی در یک پارامتر مجزای دیگر (به عنوان مثال Ceq) ارائه شوند.

```
function [c,ceq] = mycon(x)
c = ...     % Compute nonlinear inequalities at x.
ceq = ...   % Compute nonlinear equalities at x.
```

محدودیت های غیرخطی نامساوی

محدودیت های غیرخطی مساوی

nonlcon = @mycon

نحوه قرار دادن تابع تعریف شده جهت معرفی محدودیت های غیرخطی در دستور ga

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

به عنوان مثال جهت معرفی محدودیت های غیرخطی زیر به دستور ga، لازم است تابع زیر تعریف شود:

$$x_1^2 + \frac{x_2^2}{2} \leq 1$$
$$x_1^2 \geq x_2^2 + 1$$
$$x_1 x_2 + x_2 = 3$$

```
function [C Ceq] = Nonlcons(x)
C(1) = x(1)^2 + x(2)^2/2 - 1;
C(2) = x(2)^2 - x(1)^2+1;
Ceq =x(1)*x(2)+x(2)-3;
```

```
nonlcon = @Nonlcons
```

در صورتی که مسأله دارای محدودیت های خطی مساوی و یا نامساوی نمی باشد، لازم است nonlcno=[] قرار داده شود.

نکته ۱: در دستور ga در صورتی که در options، نوع متغییرهای تصمیم (PopulationType) از نوع 'bitString' باشد (ژن ها به صورت بیت تعریف شوند)، نمی توان از محدودیت های غیر خطی استفاده نمود.

نکته ۲: در صورت وجود متغییرهای تصمیم از نوع عدد صحیح (IntCon تهی نباشد)، جهت اعمال محدودیت های غیرخطی نامساوی باید Ceq=[] در نظر گرفته شود.

IntCon: برداری از شماره متغییرهای تصمیم که به صورت عدد صحیح می باشند. طول این بردار از یک تا numel(nvars) متغییر می باشد. به عنوان مثال، بردار مقابل نشان می دهد که متغییرهای تصمیم سوم، ششم و نهم به عنوان متغییرهای عدد صحیح در نظر گرفته شده اند.

```
IntCon = [3,6,9]
```

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

options: با استفاده از دستور options = optimoptions(SolverName,Name,Value) که قبلاً معرفی گردید، می توان گزینه های اضافی مرتبط با الگوریتم فراکاوشی ژنتیک را تعریف نمود. با توجه به اینکه حل کننده مدنظر ga می باشد لذا باید بجای Solvername از 'ga' استفاده نمود.

options = optimoptions('ga',Name,Value)

پارامترهایی که توسط Name و Value قابل تنظیم می باشد، عبارتند از:

- با انتخاب 'ConstraintTolerance' در Name می توان حد موجه بودن محدودیت های خطی و غیرخطی را با ارائه یک مقدار مثبت حقیقی تغییر داد. لازم به ذکر است مقدار پیش فرض این حد برابر با ۰/۰۰۱ می باشد. به عبارت دیگر در صورتی که بر مبنای جواب بهینه بدست آمده، میزان تخطی از محدودیت ها از حد تعیین شده بیشتر باشد اجرای الگوریتم ادامه می یابد.
- جهت تولید نسل اولیه می توان از 'CreationFcn' در Name و 'gacreationuniform' و یا 'gacreationlinearfeasible' (مقدار پیش فرض که برای محدودیت های خطی کاربرد دارد) در Value استفاده نمود. همچنین می توان با معرفی توابع دلخواه در value نیز اقدام به تولید نسل اولیه نمود.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

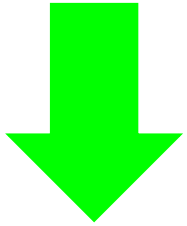
- با انتخاب 'CrossoverFcn' در Name می توان نوع تابع مرتبط با عملگر تزویج را تغییر داد. توابعی که کاربر می تواند آن ها را توسط Value انتخاب نماید عبارتند از: 'crossoverscattered'، 'crossoverheuristic'، 'crossoversinglepoint'، 'crossovertwopoint'، 'crossoverintermediate' و 'crossoverarithmetic'. همچنین می توان توابع دلخواه مدنظر جهت انجام عملیات تزویج را تعریف و به Value معرفی نمود.

نحوه عملکرد عملگر تزویج پراکنده (crossoverscattered): این روش برای مسائل بدون محدودیت خطی کاربرد دارد. به ازای هر جفت کروموزم یک بردار باینری (صفر و یک) به تعداد متغییرهای تصمیم تولید می کند. در موقعیت های از این بردار که حاوی عدد یک است از کروموزم والد اول و در موقعیت های با عدد صفر از کروموزم والد دوم جهت تولید کروموزم فرزندان استفاده می نماید.

بردار باینری تولید شده=[1 1 0 0 1 0 0 0]

کروموزم های والد

p1 = [a b c d e f g h]
p2 = [1 2 3 4 5 6 7 8]



کروموزم فرزند

child1 = [a b 3 4 e 6 7 8]

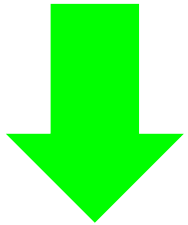
معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

نحوه عملکرد عملگر تزویج تک نقطه ای (crossoversinglepoint): این روش برای مسائل بدون محدودیت خطی کاربرد دارد. به ازای هر جفت کروموزم یک عدد صحیح بین یک و تعداد متغییر تصمیم تولید می شود. ژن های قبل از این عدد از کروموزم والد اول و ژن های کروموزم بعد از آن از کروموزم والد دوم انتخاب و با هم ترکیب شده و کروموزوم فرزندان را تولید می کنند.

کروموزم های والد

p1 = [a b c d e f g h]
p2 = [1 2 3 4 5 6 7 8]

عدد صحیح تولید شده=۳



کروموزم فرزند

child1 = [a b c 4 5 6 7 8]

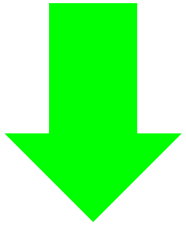
نحوه عملکرد عملگر تزویج دو نقطه ای (crossovertwopoint): این روش برای مسائل بدون محدودیت خطی کاربرد دارد. به ازای هر جفت کروموزم دو عدد صحیح m و n بین یک و تعداد متغییر تصمیم تولید می شود. ژن های قبل از عدد m و بعد از n از کروموزم والد اول و ژن های کروموزم بین m+1 و n از کروموزم والد دوم انتخاب و با هم ترکیب شده و کروموزوم فرزندان را تولید می کنند.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

کروموزم های والد

p1 = [a b c d e f g h]
p2 = [1 2 3 4 5 6 7 8]

اعداد صحیح تولید شده= ۳ و ۶



کروموزم فرزند

child1 = [a b c 4 5 6 g h]

نحوه عملکرد عملگر تزویج واسط (crossoverintermediate): این روش به عنوان روش پیش فرض برای مسائل با محدودیت خطی کاربرد دارد. این روش به صورت وزنی اقدام به ترکیب کروموزم های والد می نماید. برای این منظور لازم است از پارامتر Ratio (که می تواند به صورت یک عدد و یا یک بردار به طول تعداد متغییر تصمیم باشد) و رابطه زیر استفاده شود. مقدار پیش فرض Ratio عدد یک برای تمامی متغییرهای تصمیم است.

child = parent1 + rand * Ratio * (parent2 - parent1)

در صورتی که Ratio بین صفر و یک باشد، کروموزم های فرزند بین کروموزم های والد قرار می گیرند. در غیر اینصورت ممکن است کروموزم های فرزند خارج از بازه کروموزم های والد واقع شوند. جهت تنظیم این پارامتر می توان از دستور زیر استفاده نمود:

```
options = optimoptions('ga','CrossoverFcn', {@crossoverintermediate, ratio});
```

بجای ratio از مقدار عددی استفاده می شود:

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

نحوه عملکرد عملگر تزویج اکتشافی (crossoverheuristic): در این روش، کروموزم های فرزند در فاصله نزدیکی از کروموزم والد با تابع هدف بهتر و در فاصله دورتر از کروموزم والد با تابع هدف بدتر قرار می گیرد. جهت تعیین میزان دوری و یا نزدیکی به کورموزم برتر از پارامتر Ratio استفاده می شود. مقدار پیش فرض این پارامتر، ۱/۲ می باشد. رابطه بین کرموزوم فرزند و والد به صورت زیر است که در آن کرموزوم والد یک، از تابع هدف بهتری برخوردار است:

```
child = parent2 + Ratio * (parent1 - parent2)
```

جهت تغییر مقدار پیش فرض پارامتر Ratio از دستور زیر استفاده می شود:

```
options = optimoptions('ga','CrossoverFcn',{@crossoverheuristic,ratio});
```

نحوه عملکرد عملگر تزویج حسابی (crossoverarithmetic): در این روش، کروموزم های فرزند از میانگین حسابی کروموزم والد بدست می آیند. در صورت خطی بودن محدودیت ها، در این حالت کروموزم های فرزند تولید شده همیشه موجه خواهند بود.

نکته: جهت تعریف عملگر تزویج دلخواه لازم است با استفاده از function، تابع مدنظر با ورودی های زیر تعریف شود:

```
xoverKids = myfun(parents, options, nvars, FitnessFcn,unused,thisPopulation)
```

در این تابع، parents: بردار ردیفی از کرموزم های والد، nvars: تعداد متغییرهای تصمیم، FitnessFcn: تابع هدف

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

xoverKids = myfun(parents, options, nvars, FitnessFcn,unused,thisPopulation)

thisPopulation: ماتریس با ابعاد اندازه جمعیت در تعداد متغییر تصمیم می باشد. خروجی نیز ماتریسی با ابعاد اندازه جمعیت در تعداد متغییر تصمیم است که نشان دهنده کروموزم های حاوی فرزندان می باشد.

لازم به ذکر است در صورتی که محدودیت ها مسأله مورد بررسی به صورت خطی باشند، لازم است تابع مرتبط با عملگر تزویج به گونه ای تهیه شود که کروموزم های فرزند تولید شده این محدودیت ها را ارضاء نمایند در غیر اینصورت (وجود محدودیت های غیرخطی)، ارضاء محدودیت ها ضرورت ندارد.

- با انتخاب ‘CrossoverFraction’ در Name می توان درصدی از جمعیت نسل بعد که شامل کروموزم های برتر نمی باشند و توسط عملگر تزویج تولید می شوند را در Value توسط یک مقدار مثبت تعیین نمود. مقدار پیش فرض این پارامتر ۰/۸ است. لازم به ذکر است در صورت انتخاب مقدار یک برای این پارامتر، عملیات جهش انجام نمی شود و انتخاب صفر به منزله انجام صرفاً عملیات جهش بر روی کروموزم ها می باشد.
- در صورت انتخاب ‘Display’ در Name امکان مشاهده روند دستیابی به جواب بهینه با ارائه عبارت 'iter‘ در Value و یا صرفاً مقدار نهایی جواب با ارائه 'final‘ در Value و یا عدم نمایش جواب ها در طی دوره بهینه سازی با انتخاب 'off‘ در Value فراهم می شود.

options = optimoptions('ga','Display','final');

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

- با انتخاب ’EliteCount‘ در Name می توان مشخص نمودن که چه تعداد از کروموزم های نسل فعلی بدون تغییر در نسل بعدی مورد استفاده قرار گیرند. جهت ارائه تعداد کروموزم ها از یک عدد صحیح مثبت در Value استفاده می شود. مقدار پیش فرض آن ceil(0.05*PopulationSize) می باشد.
- در صورت انتخاب ’FitnessLimit‘ در Name می توان با رسیدن مقدار تابع هدف به مقدار مشخص شده در Value، از ادامه فرآیند بهینه سازی جلوگیری به عمل آورد. مقدار پیش فرض این گزینه، -Inf است که به معنی یافتن حداقل ترین مقدار تابع هدف است.

پدیده crowding: پدیده ای است که در آن کروموزمی که سازگاری بسیار بیشتری از بقیه افراد جمعیت دارد (یعنی دارای تابع هدف بهتری است) بطور مرتب تولید نسل کرده و با تولید اعضای مشابه درصد عمده ای از جمعیت را اشغال می کند. این کار باعث کاهش پراکندگی جمعیت شده و سرعت GA را کم می کند. راه حل برطرف نمودن این مشکل استفاده از روش مقیاس سازی تابع هدف می باشد. برای این منظور لازم است در Name از ’FitnessScalingFcn‘ و در Value نام یکی از روش های مقیاس سازی تابع هدف استفاده شود.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

روش های مقیاس سازی تابع هدف در واقع بازه تغییرات تابع هدف اولیه را جهت استفاده عملگر انتخاب مناسب می نمایند. با استفاده از این روش کروموزم هایی که دارای مقدار تابع هدف مقیاس شده بالاتری می باشند، از احتمال انتخاب بالایی برخوردار خواهند بود. بازه تغییرات تابع هدف مقادیر مقیاس شده بر روی عملکرد الگوریتم ژنتیک مؤثر است. در صورتی که این مقادیر مقیاس شده دارای بازه وسیعی باشند، کروموزم هایی با مقادیر بالایی تابع هدف مقیاس شده به سرعت انتخاب می شوند و در نتیجه الگوریتم امکان جستجوی سایر فضاهای موجه را از دست می دهد. به طور عکس نیز در صورت کوچک بودن بازه تغییرات تابع هدف مقیاس شده، تمامی کروموزم ها از شانس یکسانی در انتخاب برخوردار خواهند بود و لذا عملیات جستجوی زمان بر و با سرعت کمتری انجام می شود.

روش های مقیاس سازی تابع هدف عبارتند از:

'fitscalingrank': در این روش بر اساس رتبه (n) هر کروموزم مرتب شده با توجه به تابع هدف، با استفاده از رابطه $1/\sqrt{n}$ مقیاس سازی می شود. این عمل منجر ایجاد شانس تقریباً برابر برای انتخاب کروموزم های نامطلوب می شود. این رویکرد به عنوان پیش فرض روش مقیاس سازی است. این روش اثرات مرتبط با گسترده بودن تابع هدف را از بین می برد.

```
options = optimoptions('ga','FitnessScalingFcn','fitscalingrank')
```

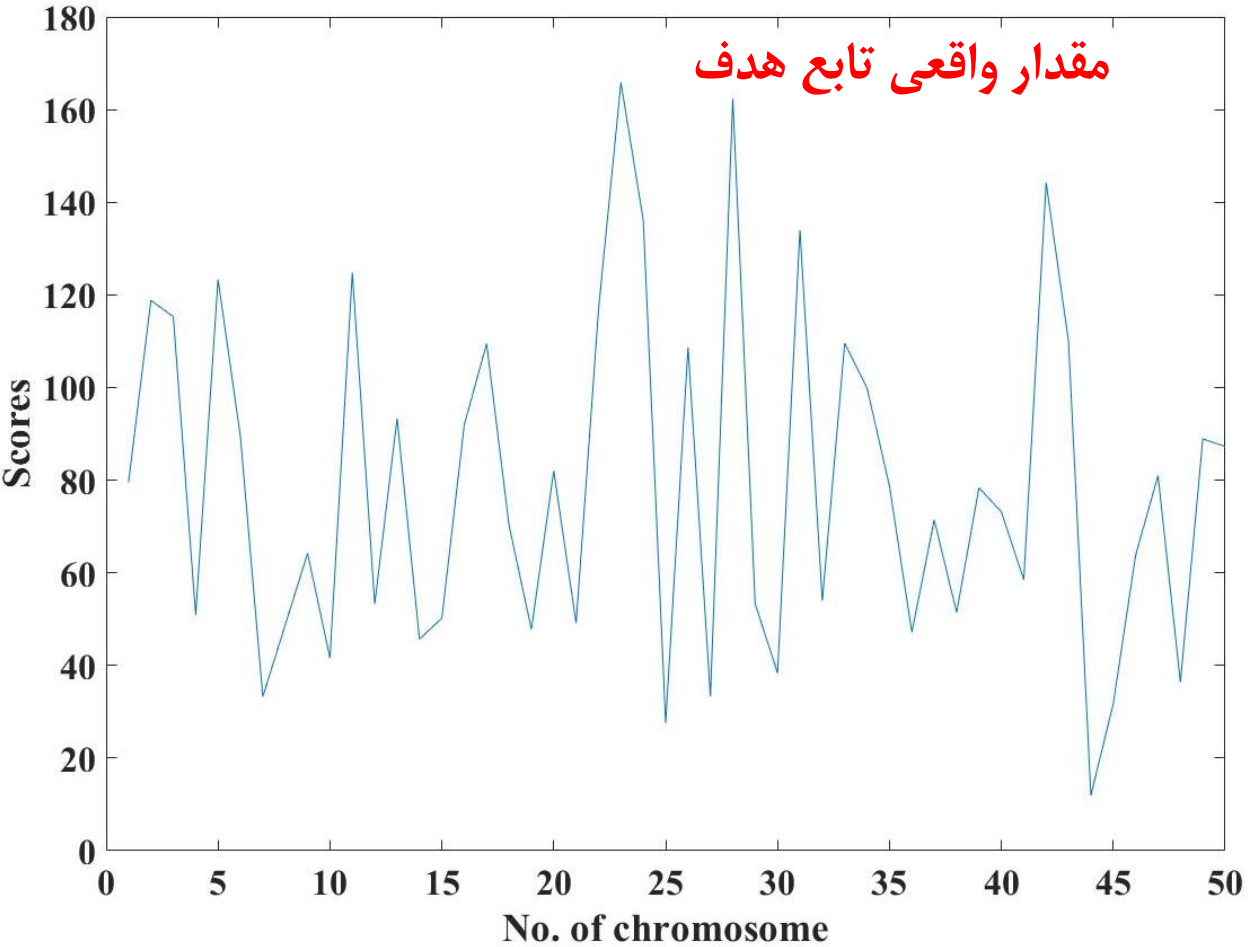
معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

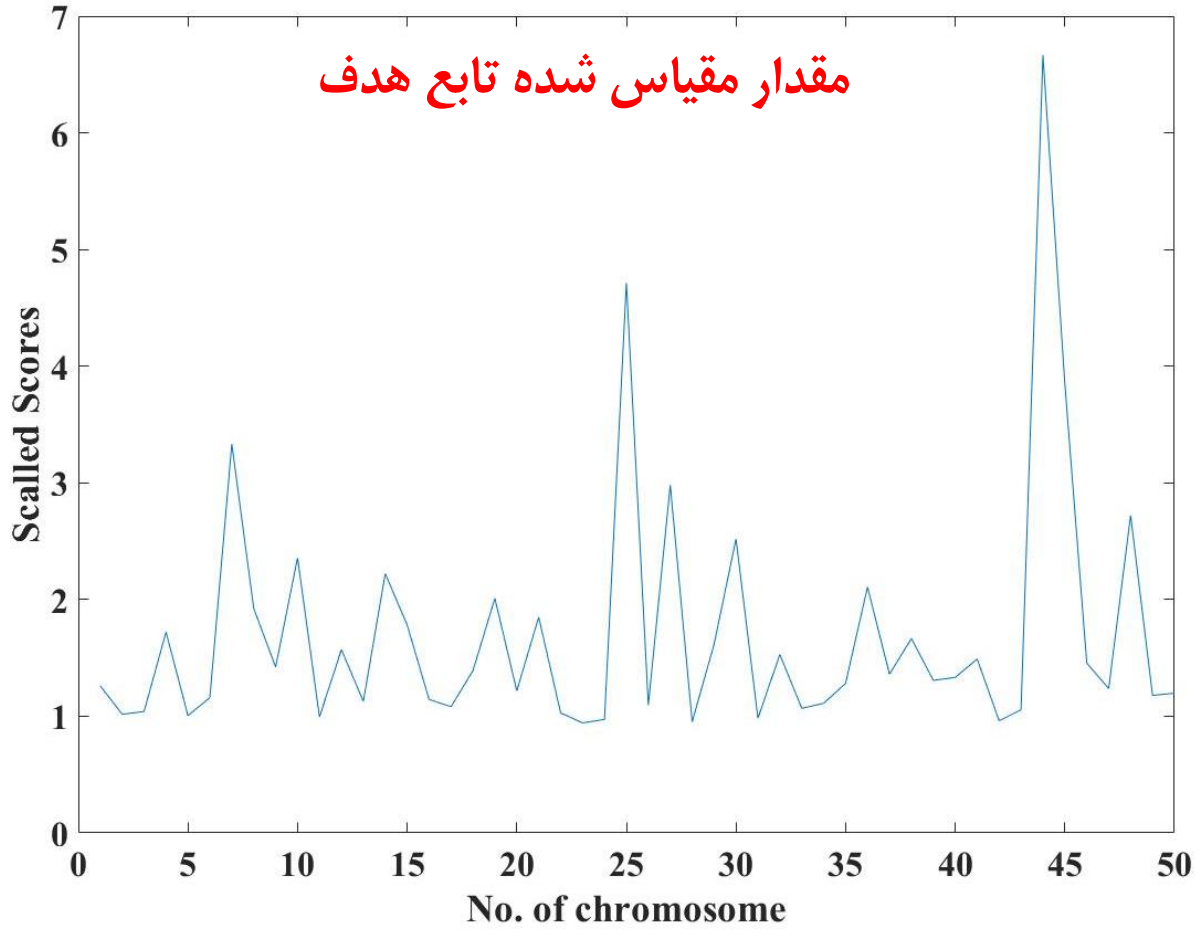
```
function expectation = fitscalingrank(scores,nParents)
scores = scores(:); [~,i] = sort(scores);
expectation = zeros(size(scores));
expectation(i) = 1 ./ ((1:length(scores))  .^ 0.5);
expectation = nParents * expectation ./ sum(expectation);
```

کد مرتبط با این روش به صورت زیر می باشد:

آدرس فایل:

C:\Program Files\MATLAB\R2022b\toolbox\globaloptim\globaloptim





معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

'fitscalingprop': این روش با توجه به مقدار تابع هدف هر کروموزم اقدام به مقیاس سازی می نماید.

```
options = optimoptions('ga','FitnessScalingFcn','fitscalingprop')
```

```
function expectation = fitscalingprop(scores,nParents)
scores = scores(:);
scores = 2 * mean(scores) - scores;
m = min(scores);
if(m < 0)
    scores = scores - m;
end
expectation = nParents * scores ./ sum(scores);
```

'fitscalingtop': در این روش درصدی از کروموزم های برتر نسل فعلی از یک مقیاس بالا و ثابت برخوردار می باشند و به سایر کروموزم ها، مقیاس صفر اختصاص می یابد. مقدار پیش فرض این درصد برابر با ۰/۴ است که می توان با استفاده از پارامتر quantity تغییر داد.

```
options = optimoptions('ga','FitnessScalingFcn',{@fitscalingtop,quantity})
```

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

```
function expectation = fitscalingtop(scores,nParents,quantity)
if nargin < 3 || isempty(quantity)
    quantity = 0.4;
end
scores = scores(:);
 if quantity < 1
    quantity = round(quantity * length(scores));
end
 expectation = zeros(size(scores));
[~,i] = sort(scores);
expectation(i(1:quantity)) = nParents / quantity;
```

'fitscalingshiftlinear': در این روش بر مبنای حاصلضرب عدد ثابت rate در متوسط تابع هدف کروموزم ها نسبت به تابع هدف سایر کروموزم ها عملیات مقیاس سازی انجام می شود. بر این اساس مقیاس متناسب با حاصلضرب عدد ثابت rate (با مقدار پیش فرض ۲) در متوسط تابع هدف کروموزم ها به بهترین کروموزم اختصاص می یابد.

```
options = optimoptions('ga','FitnessScalingFcn',{@fitscalingshiftlinear, rate})
```

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

```
function expectations = fitscalingshiftlinear(scores,nParents,MaximumSurvivalRate)
if nargin < 3 || isempty(MaximumSurvivalRate)
    MaximumSurvivalRate = 2;
end
 scores = -scores(:);
 maxScore  = max(scores);
meanScore = mean(scores);
minScore  = min(scores);
 if(~isfinite(meanScore))
    error(message('globaloptim:FITSCALINGSHIFTLINEAR:finiteScore'));
end
if(maxScore == minScore)
    expectations = ones(length(scores),1) ./ length(scores);
    return;
end
desiredMean = nParents/length(scores);
 scale = desiredMean * (MaximumSurvivalRate  - 1) / (maxScore - meanScore);
 offset = desiredMean - (scale * meanScore);
if(offset + scale * minScore < 0)
    scale = desiredMean / (meanScore - minScore);
    offset = desiredMean - (scale * meanScore);
end
expectations = offset + scale * scores;
```

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

- با انتخاب ’FunctionTolerance‘ در Name می توان آستانه متوسط تغییرات نسبی تابع هدف در طی یک تعداد تکرار مشخص را در Value تعیین نمود. بر این اساس چنانچه متوسط تغییرات نسبی تابع هدف در طی یک تعداد تکرار مشخص (که با ارائه ’MaxStallGenerations‘ در Name و یک عدد صحیح مثبت در Value تعیین می شود. مقدار پیش فرض آن ۵۰ تکرار است.) از این آستانه کمتر باشد، الگوریتم بهینه سازی متوقف می شود. مقدار پیش فرض آستانه تغییرات 10^{-6} است. لازم به ذکر است در صورت انتخاب StallTest در Name و استفاده از ‘geometricWeighted' در Value، بجای محاسبه متوسط تغییرات نسبی تابع هدف از متوسط وزنی تغییرات نسبی تابع هدف استفاده می شود. مقدار پیش فرض StallTest در Name، ‘averageChange' در Value می باشد.

- با انتخاب ’HybridFcn‘ در Name می توان جهت اطمینان از یافتن مقدار بهینه کلی از یک الگوریتم بهینه سازی دیگر همانند ‘fminsearch'، ‘patternsearch'، ‘fminunc' و ‘fmincon' در Value استفاده نمود.

```
options = optimoptions('ga','HybridFcn','patternsearch')
```

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

- با انتخاب ’InitialPenalty‘ در Name می توان مقدار جریمه اولیه مرتبط با جواب هایی که فضای موجه را رعایت نمی کنند، توسط یک عدد مثبت در Value معرفی نمود. مقدار پیش فرض، عدد ۱۰ است.
- جهت تولید نسل اولیه از ’InitialPopulationMatrix‘ در Name و ارائه یک ماتریس با ابعاد اندازه جمعیت در تعداد متغییر تصمیم در Value استفاده می شود. لازم به ذکر است در صورتی که ماتریس ارائه شده دارای ردیف هایی کمتر از اندازه جمعیت اعلام شده به الگوریتم باشد، سایر ردیف های باقیمانده توسط الگوریتم ساخته می شود.

```
sravn ,ncfssentif@)ag = [pop_lanif,tuptuo,galftixe,lavf,x]);
```
اجرای اولیه الگوریتم ژنتیک

```
options = optimoptions('ga','InitialPop', final_pop);
```
استفاده از جمعیت بهینه اولیه تولید شده به عنوان جمعیت اولیه برای اجرای مجدد

```
[x,fval,exitflag,output,final_pop2]= ga(@fitnessfcn,nvars,[],[],[],[],[],[],[],options);
```

- استفاده از ’InitialPopulationRange‘ در Name این امکان را به کاربر می دهد تا بازه مجاز مرتبط با هر یک از ژن های کروموزم را جهت تولید نسل اولیه در value به صورت یک بردار و یا ماتریس مشخص نماید. برای مسائل بدون محدودیت این بازه [-10;10] است و برای مسائل دارای محدودیت، لازم است از [lb;ub] استفاده شود.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

- با انتخاب ’MaxGenerations‘ در Name می توان تعداد تکرارهای مدنظر جهت اجرای الگوریتم بهینه سازی را در Value با ارائه یک عدد صحیح مثبت مشخص نمود. مقدار پیش فرض آن، ۱۰۰ برابر تعداد متغییرهای تصمیم می باشد.
- در صورتی که ’MaxStallTime‘ در Name استفاده شود، کاربر می توان با ارائه یک عدد مثبت در Value (به معنای مدت زمان به ثانیه می باشد) اقدام به ارائه شرط برای الگوریتم بهینه سازی نماید. در این حالت اگر بهبودی در تابع هدف در طی مدت مشخص شده در Value ایجاد نشد، اجرای الگوریتم بهینه سازی متوقف می شود.
- با انتخاب ’MaxTime‘ در Name و ارائه یک عدد مثبت در value (به عنوان مدت زمان اجرا الگوریتم به ثانیه) امکان تعیین مدت زمان اجرای الگوریتم توسط کاربر فراهم می شود.
- در صورتی که اندازه جمعیت به صورت یک بردار باشد و در طی فرآیند بهینه سازی بتواند جمعیت های مختلفی را تجربه نماید، می توان با انتخاب ’MigrationDirection‘ در Name و استفاده از ’forward' و یا ’both' در Value جهت حرکت کروموزم ها را در بین جمعیت های مختلف کنترل نمود. در forward، اطلاعات جمعیتی آخرین نسل برای نسل بعدی مورد استفاده قرار می گیرد اما در both، اطلاعات جمعیتی نسل های قبل و بعد از نسل فعلی در نسل فعلی استفاده می گردند.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

- با انتخاب ’MigrationFraction‘ در Name و ارائه یک مقدار بین صفر و یک در Value، برای مسائل با اندازه جمعیت های متغییر، می توان درصدی از نسل فعلی که در نسل بعدی مورد استفاده قرار می گیرند را تعیین نمود. به عنوان مثال اگر اندازه جمعیت نسل قبلی و فعلی به ترتیب ۵۰ و ۱۰۰ باشند و Value، ۰/۴ انتخاب گردد، تنها ۲۰ کروموزم نسل قبل به نسل فعلی منتقل می شوند. مقدار پیش فرض برابر با ۰/۲ است.

نکته: در فرآیند مهاجرت کروموزم ها در بین نسل ها، همیشه بهترین کروموزم ها مهاجرت کرده و جایگزین بدترین کروموزم ها می شوند.

- با انتخاب ’MigrationInterval‘ در Name و ارائه یک عدد صحیح مثبت در Value، می توان فرآیند مهاجرت کروموزم بین نسل را پس از یک تعداد تکرار مشخص تعیین نمود. به عنوان مثال اگر مقدار Value برابر با ۵ در نظر گرفته شود، پس از هر ۵ تکرار، عملیات مهاجرت کروموزم بین نسل ها انجام می شود. مقدار پیش فرض برابر با ۲۰ است.
- با انتخاب ’MutationFcn‘ در Name و ارائه نام یکی از توابع مرتبط با عملگر جهش در Value، می توان نحوه اعمال عملگر جهش را بر روی کروموزم ها تعیین نمود. در واقع این عملگر منجر به ایجاد تنوع و گستردگی در کروموزم ها شده که موجبات جستجوی در فضای گسترده تری از منطقه موجه می شود. لازم به ذکر است این توابع برای مسائل برنامه ریزی عدد صحیح استفاده نمی شود.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

نحوه عملکرد عملگر جهش زنگوله ای (mutationgaussian): این روش که برای مسائل بدون محدودیت کاربرد دارد، اعداد تصادفی را بر اساس توزیع زنگوله ای (گوسی) با میانگین صفر و انحراف معیاری که بر مبنای دو پارامتر Scale و Shrink تعیین می شود، به مقادیر ژن های کروموزم والد اضافه می کند. پارامتر Scale مقدار انحراف معیار مرتبط با اولین تکرار را تعیین می کند و بر اساس بازه تغییرات ژن ها (Initial Range) به صورت Scale*(v(2)-v(1)) محاسبه می شود. در این رابطه v(2) و v(1) به ترتیب برداری حاوی حدود بالا و پایین تعریف شده برای هر ژن با ابعاد دو ردیف در تعداد متغییرهای تصمیم می باشد.

پارامتر Shrink که میزان جمع شدگی انحراف معیار در طی تکرارهای مختلف را تعیین می کند، بر اساس رابطه زیر اقدام به کنترل نحوه تغییرات انحراف معیار در طی فرآیند بهینه سازی می نماید:

$$\sigma_k = \sigma_{k-1}\left(1 - Shrink\frac{k}{Generations}\right)$$

در این رابطه σ_k، انحراف معیار تکرار K ام و $Generations$، تعداد کل تکرارها است. با ارائه مقدار یک به پارامتر Shrink مقدار انحراف معیار به صورت خطی تا دستیابی به مقدار صفر در آخرین تکرار تغییر می کند. جهت تنظیم این دو پارامتر از دستور زیر استفاده می شود (مقادیر پیش فرض این دو پارامتر برابر با یک می باشد):

```
options = optimoptions('ga','MutationFcn',{@mutationgaussian, scale, shrink})
```

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

نکته: در صورت استفاده از عملگر جهش زنگوله ای، افزایش قابل توجه تعداد تکرار می تواند منجر به بدتر شدن جواب بهینه نهایی شود. علت این امر کاهش مقدار متوسط جهش بر روی کروموزم ها در طی تکرارها می باشد.

نحوه عملکرد عملگر جهش یکنواخت (mutationuniform): این روش که برای مسائل بدون محدودیت کاربرد دارد، از دو مرحله تشکیل شده است:

۱- انتخاب ژن هایی که در هر کروموزم باید تغییر داده شوند. برای این منظور برای هر ژن یک عدد تصادفی تولید می شود. در صورتی که این عدد از نرخ احتمال (Probability rate) مشخص شده توسط کاربر (با مقدار پیش فرض ۰/۰۱) کمتر بود، آن ژن جهت اعمال تغییرات مدنظر مورد توجه قرار می گیرد.

۲- جهت اعمال تغییرات در ژن های منتخب بر اساس بازه تغییرات مجاز هر ژن (متغییر تصمیم)، یک مقدار تصادفی در این بازه تولید و جایگزین مقدار ژن موجود می گردد.

جهت تغییر مقدار نرخ احتمال از دستور زیر استفاده می شود:

```
options = optimoptions('ga','MutationFcn', {@mutationuniform, rate})
```

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

نحوه عملکرد عملگر جهش موجه تطبیقی (mutationadaptfeasible): این روش که برای مسائل دارای محدودیت کاربرد دارد، در ابتدا ژن هایی را به صورت تصادفی جهت اعمال جهش انتخاب می کند. با توجه به میزان بهبود تابع هدف (جواب برتر) در طی تکرارهای قبل و رعایت محدودیت های تعریف شده، تغییرات بر روی ژن های منتخب انجام می شود.

- با انتخاب 'PenaltyFactor' در Name می توان مقدار جریمه اولیه داده شده مرتبط با راه حل هایی که فضای موجه را رعایت نمی کنند، با ارائه یک عدد مثبت در Value بروز نمود. این مقدار جایگزین جریمه اولیه می شود. مقدار پیش فرض، عدد ۱۰۰ است.

- جهت ترسیم پارامترهای محاسبه شده توسط الگوریتم ژنتیک در طی فرآیند بهینه سازی از 'PlotFcn' در Name و توابع معرفی شده زیر همراه با @ (و یا با ارائه نام تابع در بین دو کوتیشن) در Value استفاده می شود. لازم به ذکر است جهت ترسیم همزمان چند پارامتر باید نام توابع را در داخل {} قرار داد. همانند: {'gaplotstopping','gaplotscorediversity'}

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

۱- تابع @gaplotscorediversity (یا 'gaplotscorediversity'):

هیستوگرامی از مقدار تابع هدف مرتبط با کروموزم های هر تکرار را نشان می دهد. بر اساس این نمودار می توان میزان تنوع کروموزم ها (تغییرات در تابع هدف کروموزم ها) را مورد بررسی قرار داد.

به عنوان مثال بر اساس این شکل می توان دریافت که مقدار تابع هدف اکثر کروموزم های آخرین نسل تقریباً یکسان و برابر با ۴ است و تنوع و پراکندگی در این مقادیر نمی توان یافت.

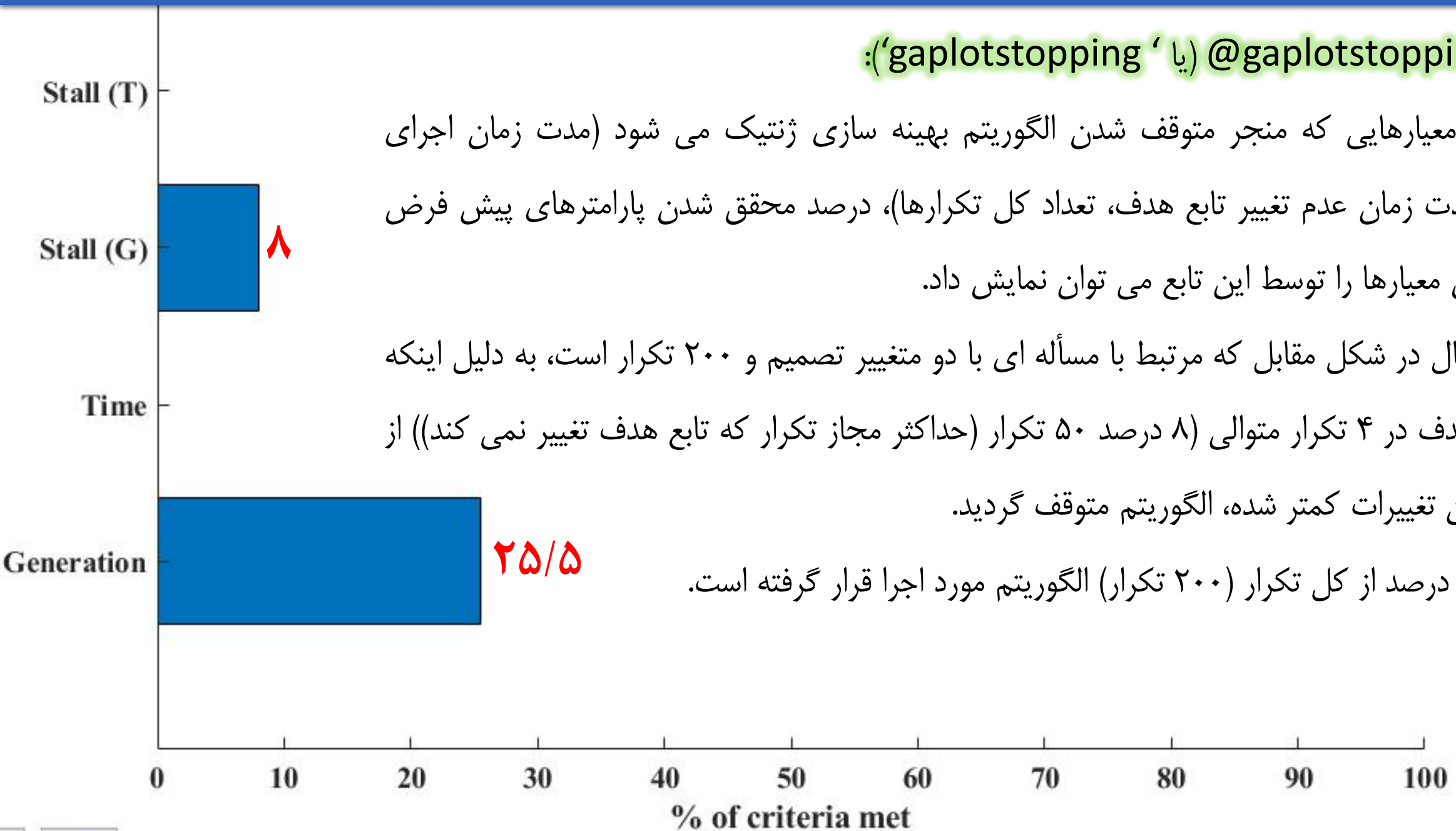
معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

۲- تابع @gaplotstopping (یا 'gaplotstopping'):

با توجه به معیارهایی که منجر متوقف شدن الگوریتم بهینه سازی ژنتیک می شود (مدت زمان اجرای الگوریتم، مدت زمان عدم تغییر تابع هدف، تعداد کل تکرارها)، درصد محقق شدن پارامترهای پیش فرض مرتبط با این معیارها را توسط این تابع می توان نمایش داد.

به عنوان مثال در شکل مقابل که مرتبط با مسأله ای با دو متغییر تصمیم و ۲۰۰ تکرار است، به دلیل اینکه تغییر تابع هدف در ۴ تکرار متوالی (۸ درصد ۵۰ تکرار (حداکثر مجاز تکرار که تابع هدف تغییر نمی کند)) از حداقل میزان تغییرات کمتر شده، الگوریتم متوقف گردید.

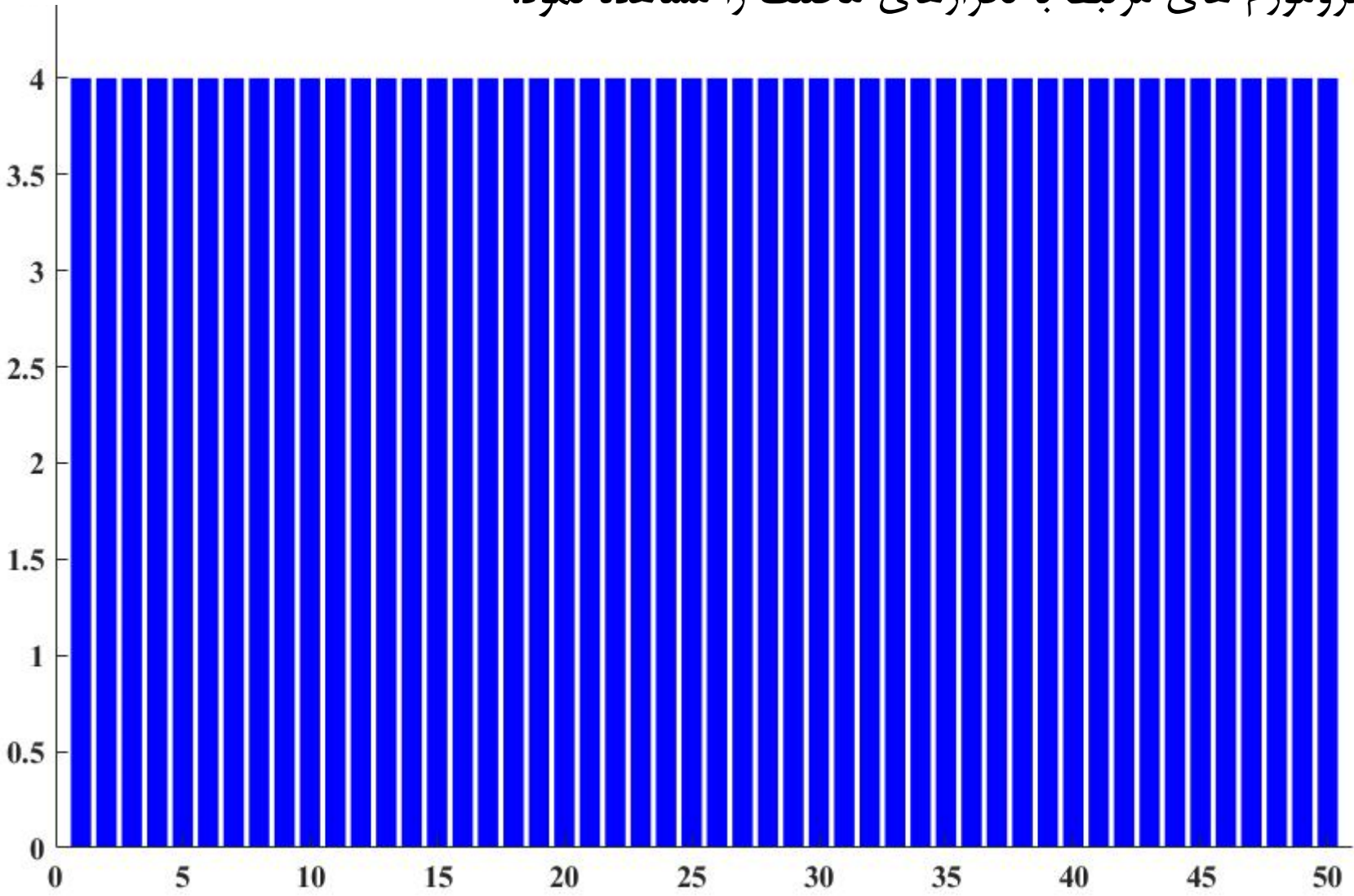
لذا در ۲۵/۵ درصد از کل تکرار (۲۰۰ تکرار) الگوریتم مورد اجرا قرار گرفته است.



معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

۳- تابع @gaplotscores (یا 'gaplotscores'):

با استفاده از این تابع می توان مقدار تابع هدف کروموزم های مرتبط با تکرارهای مختلف را مشاهده نمود.



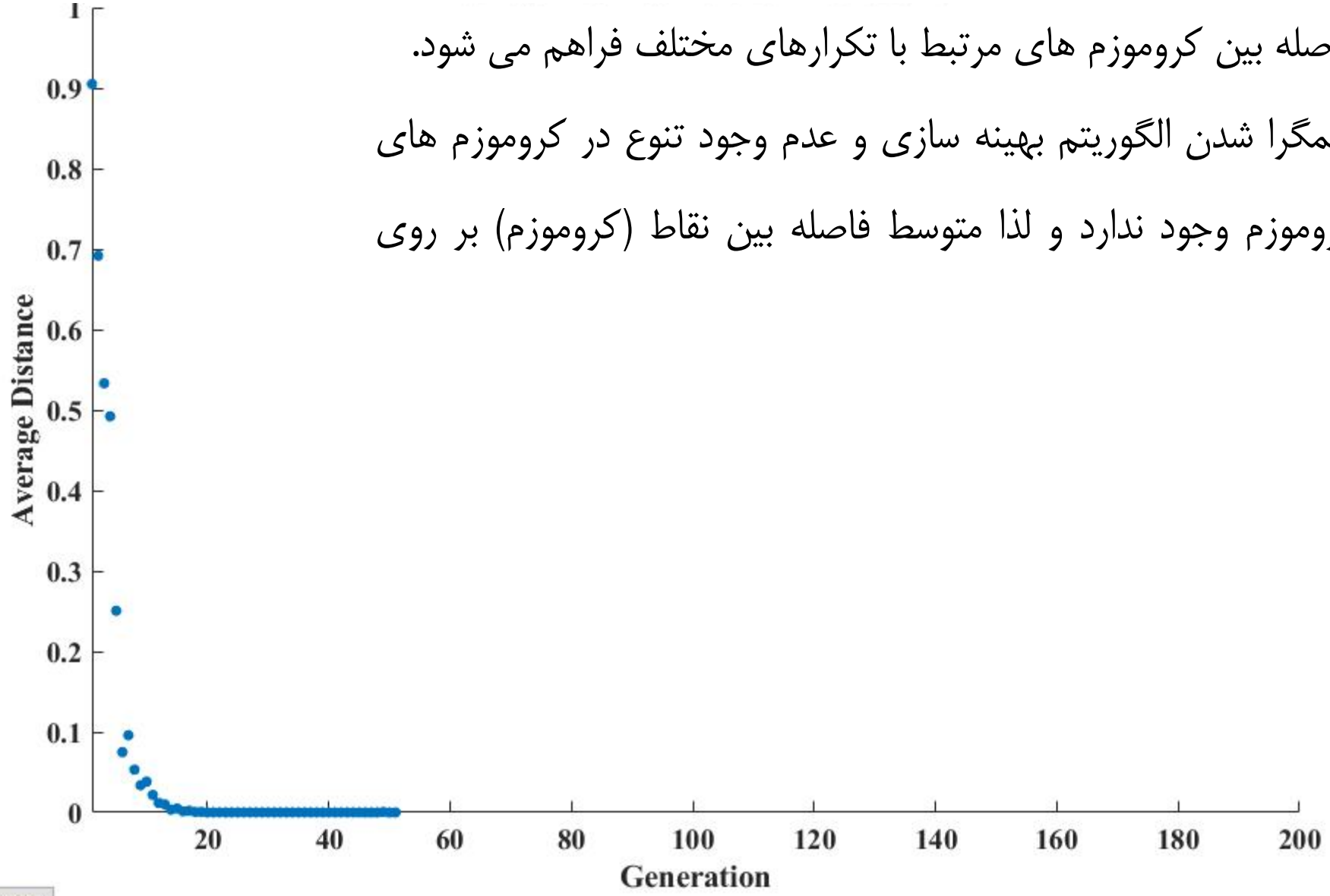
مطابق این شکل می توان دریافت که تغییرات تابع هدف به ازای کروموزم های مختلف بسیار ناچیز است و در واقع این مسأله به مقدار تابع هدف ۴ همگرا شده است.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

۴- تابع @gaplotscores (یا 'gaplotscores'):

بر اساس این تابع امکان مشاهده متوسط فاصله بین کروموزم های مرتبط با تکرارهای مختلف فراهم می شود.

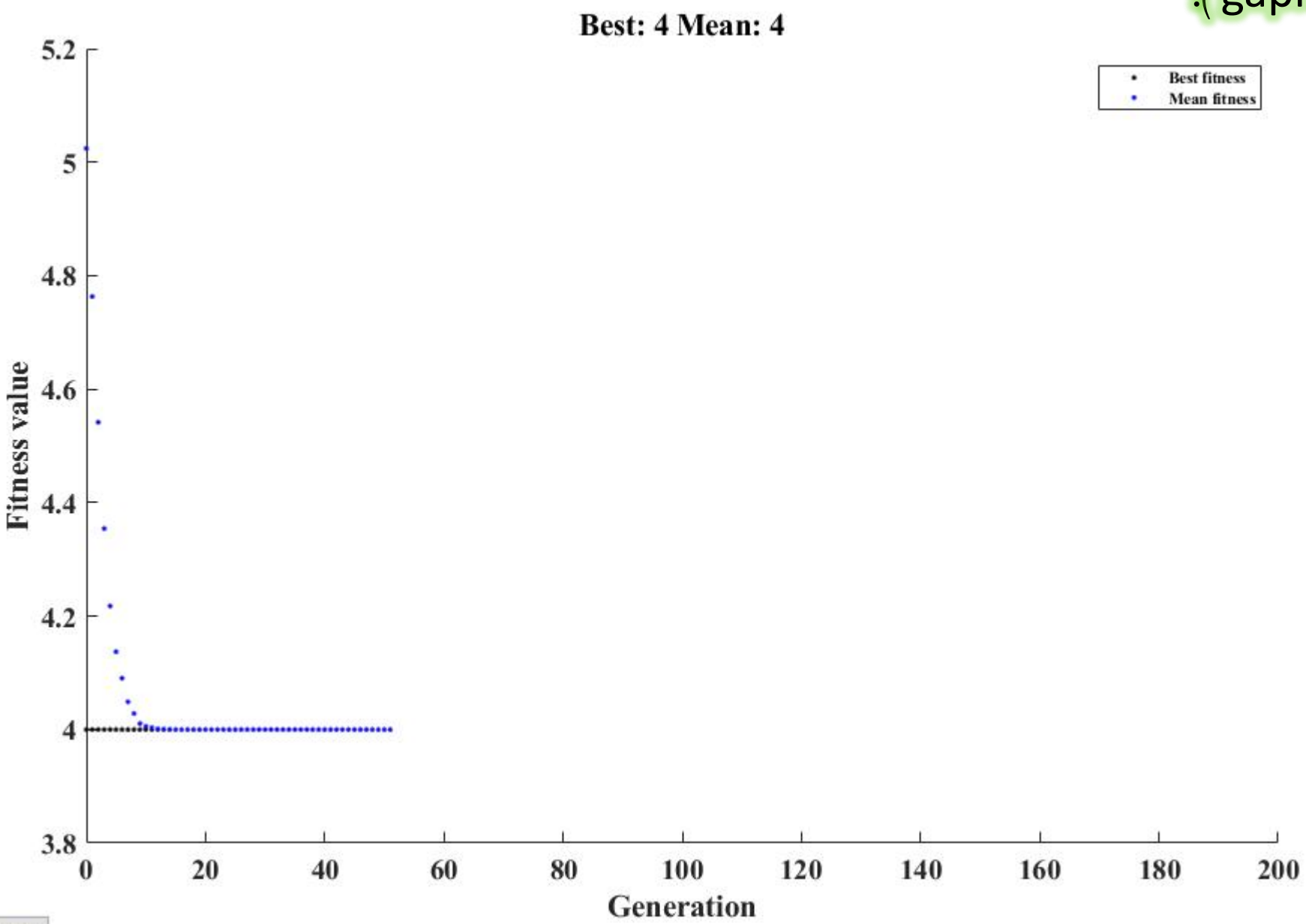
همان طور که مشاهده می شود به دلیل همگرا شدن الگوریتم بهینه سازی و عدم وجود تنوع در کروموزم های آخرین نسل، عملاً هیچ فاصله ای بین کروموزم وجود ندارد و لذا متوسط فاصله بین نقاط (کروموزم) بر روی محور افقی (عدد صفر) قرار گرفته است.



معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

۵- تابع @gaplotbestf (یا 'gaplotbestf'):

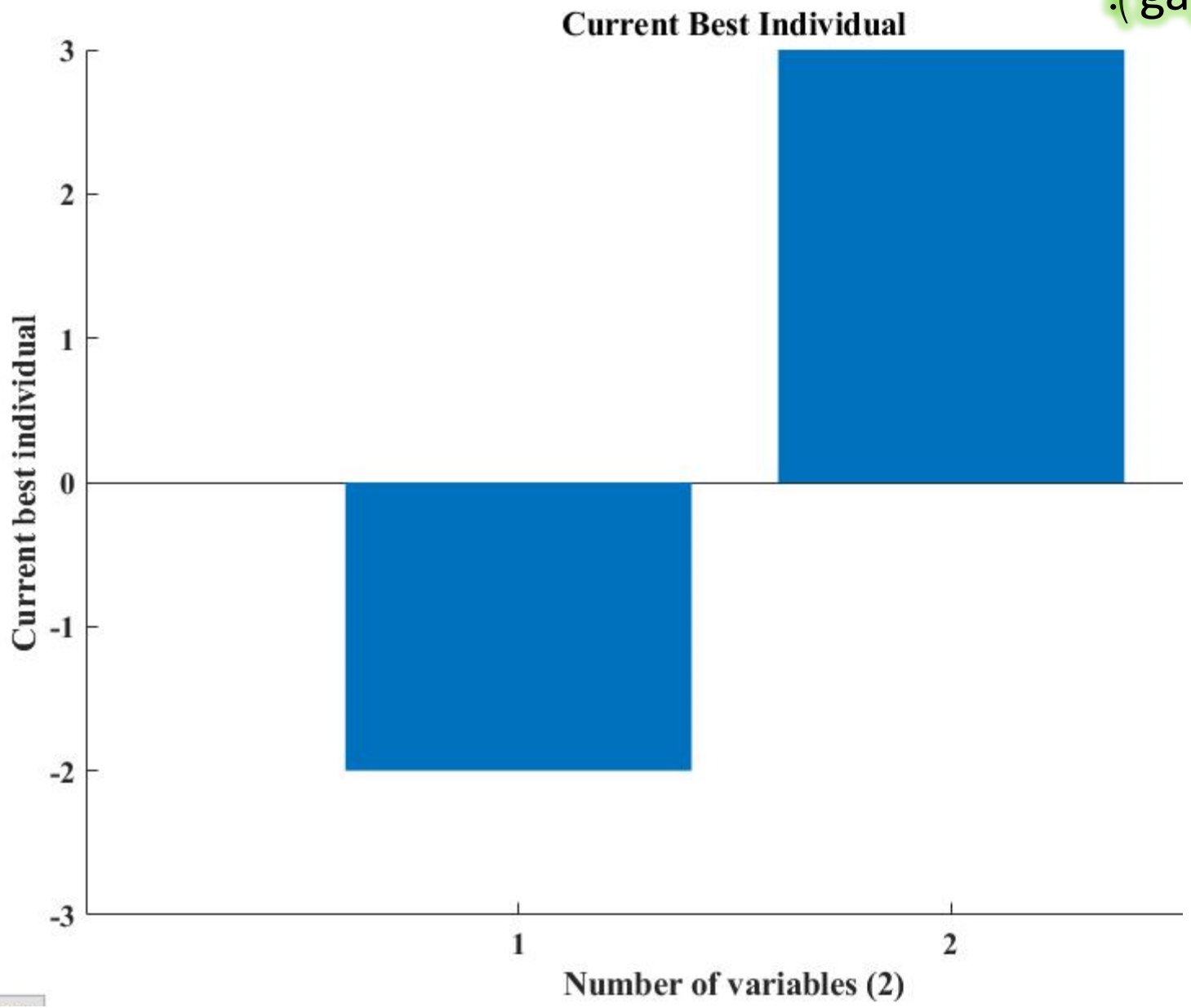
جهت مشاهده بهترین مقدار هر تکرار به همراه متوسط تابع هدف کروموزم های آن تکرار از این تابع استفاده می شود.



معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

۶- تابع @gaplotbestindiv (یا 'gaplotbestindiv'):

برای استخراج متغییرهای تصمیم بهینه مرتبط با بهترین کروموزم، از این تابع استفاده می شود. در محور افقی، به تعداد متغییرهای تصمیم مقدار وجود دارد و در روی محور عمودی، مقدار بهینه متغییر تصمیم ارائه شده است.

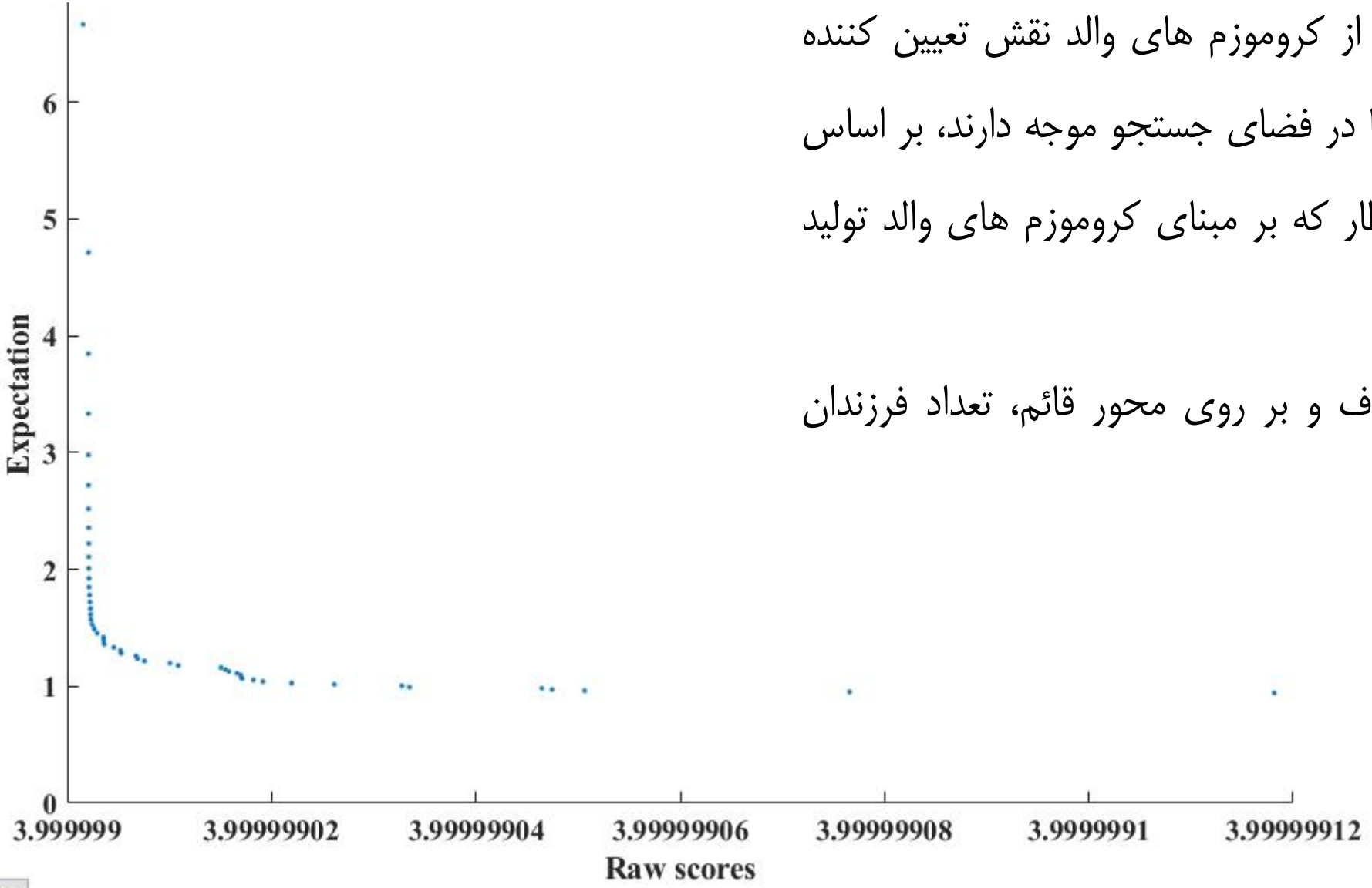


معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

۷- تابع @gaplotexpectation (یا 'gaplotexpectation'):

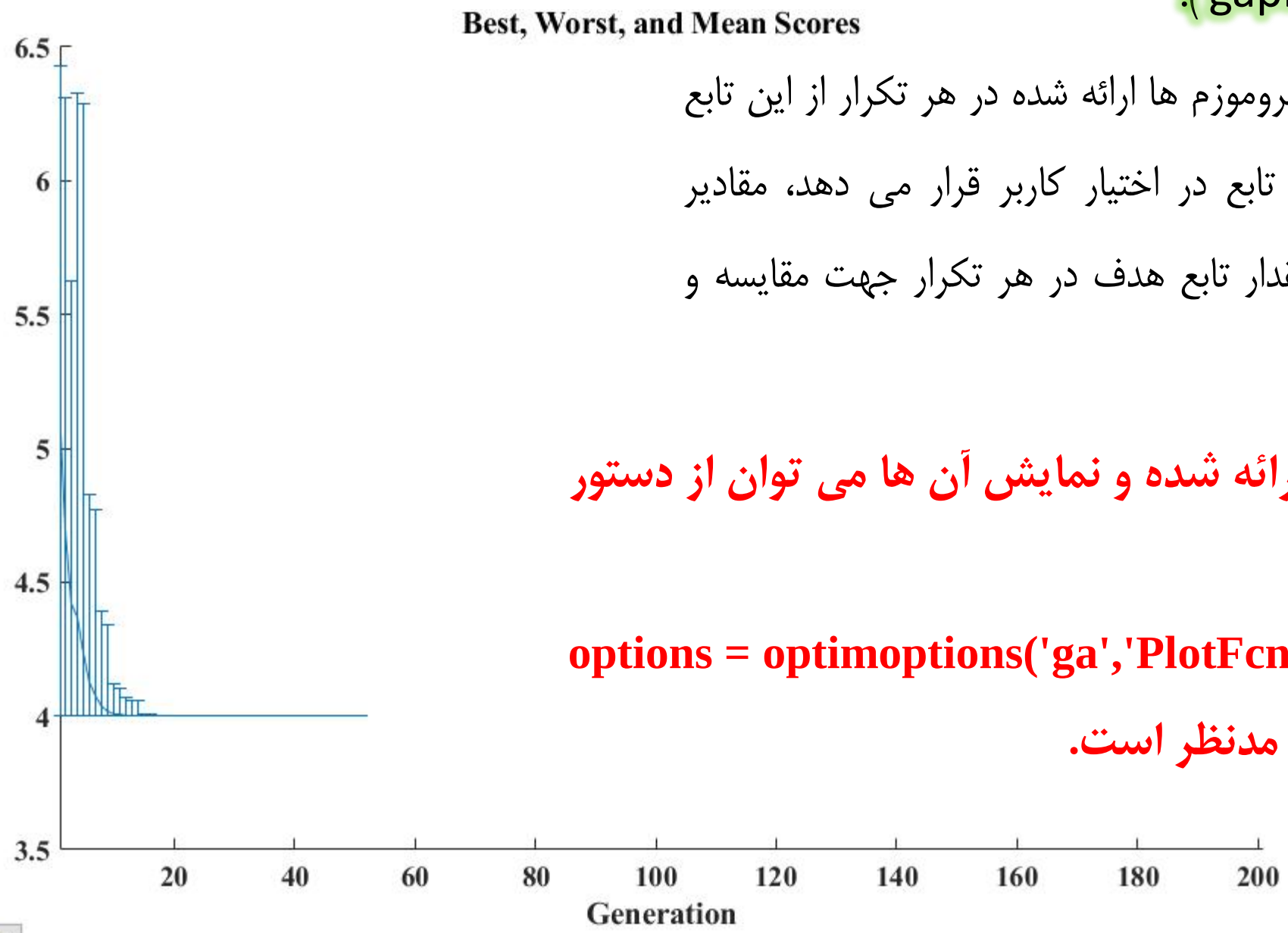
با توجه به اینکه تعداد فرزندان تولید شده از کروموزم های والد نقش تعیین کننده ای در تنوع و پراکندگی مجموعه جواب ها در فضای جستجو موجه دارند، بر اساس این تابع می توان تعداد فرزندان مورد انتظار که بر مبنای کروموزم های والد تولید می شوند را هر تکرار نمایش داد.

در محور افقی این نمودار، مقدار تابع هدف و بر روی محور قائم، تعداد فرزندان مورد انتظار ارائه شده است.



معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

۸- تابع @gaplotrange (یا 'gaplotrange'):



برای ارائه برخی از مشخصات آماری بهترین کروموزم ها ارائه شده در هر تکرار از این تابع استفاده می شود. بر مبنای نموداری که این تابع در اختیار کاربر قرار می دهد، مقادیر حداقل، حداکثر و میانگین مرتبط با بهترین مقدار تابع هدف در هر تکرار جهت مقایسه و بررسی روند بهبود جواب ها ارائه می شود.

نکته: جهت استفاده همزمان از توابع ارائه شده و نمایش آن ها می توان از دستور زیر استفاده نمود:

```
options = optimoptions('ga','PlotFcn', {@plotfun1, @plotfun2, ...});
```

در این دستور، plotfun1 و ... نام تابع مدنظر است.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

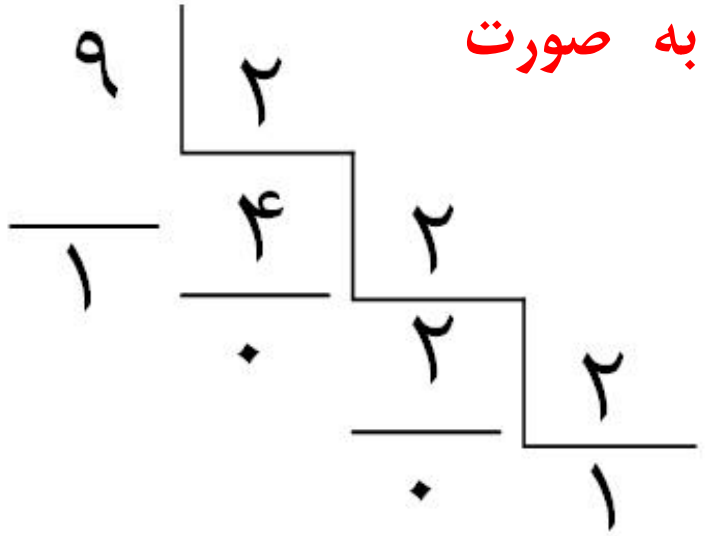
- جهت انتخاب تعداد جمعیت کروموزم ها، از 'PopulationSize' در Name و یک عدد صحیح مثبت در Value استفاده می شود. لازم به ذکر است در صورتی که تعداد متغییرهای تصمیم کمتر از ۵ باشند، مقدار پیش فرض اندازه جمعیت برابر با ۵۰ است و در غیر اینصورت این مقدار پیش فرض به ۲۰۰ افزایش می یابد. همچنین برای مسائل برنامه ریزی عدد صحیح اندازه جمعیت از رابطه زیر بدست می آید. در این رابطه nvars، تعداد متغییرهای تصمیم است.

$$\{\min(\max(10*nvars,40),100)\}$$

- برای انتخاب نوع ژن ها در کروموزم ها از 'PopulationType' در Name و توابع 'bitstring'، 'doubleVector' و یا یک تابع دلخواه در Value استفاده می شود. در تابع 'bitstring'، مقادیر ژن ها به صورت اعداد صفر و یک ارائه می شوند. در واقع هر عدد حقیقی به صورت یک بردار دودویی با طول ثابت در نظر گرفته می شود. در این روش کدگذاری مقادیر ژن ها، ابتدا با توجه به بزرگترین مقدار متغییر تصمیم، طول بیت هر ژن به صورت ثابت تعیین شده و سپس معادل دودویی آن ارائه می شود. با قرار دادن این معادل های دودویی، یک کروموزم شکل می گیرد. نکته اینکه در دستور ga، در صورت استفاده از تابع 'bitstring' و یا تعریف تابع دلخواه، تمامی محدودیت های در نظر گرفته شده در طی فرآیند بهینه سازی نادیده گرفته می شود.

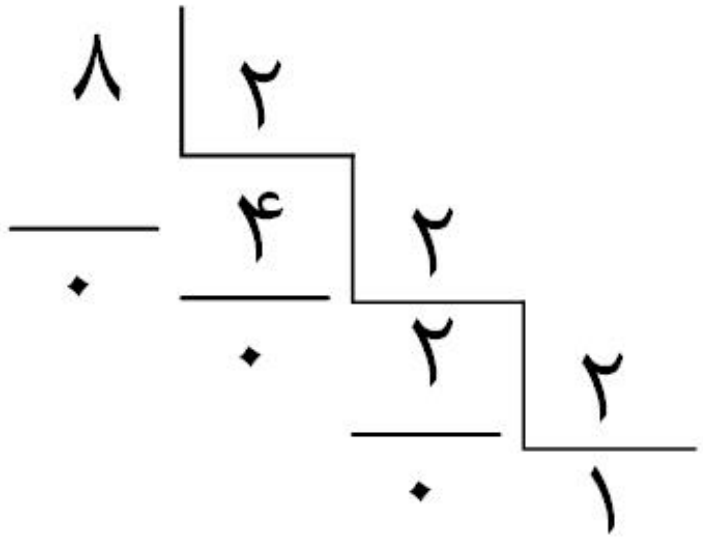
معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

نحوه تعریف اعداد حقیقی به صورت مقادیر دودویی:



بردار دودویی معادل با عدد ۹: ۱۰۰۱

$$1001 = 2^3 \times 1 + 2^2 \times 0 + 2^1 \times 0 + 2^0 \times 1$$



بردار دودویی معادل با عدد ۸: ۱۰۰۰

$$1000 = 2^3 \times 1 + 2^2 \times 0 + 2^1 \times 0 + 2^0 \times 0$$

- برای تعیین نوع تابع مورد استفاده در عملگر انتخاب لازم است از یکی از توابع زیر در Name استفاده شود:

نحوه عملکرد عملگر انتخاب یکنواخت تصادفی (selectionstochunif): در این روش بر اساس رویکرد چرخ گردان و انتخاب یکنواخت اقدام به تولید کروموزم فرزندان بر مبنای کروموزم والدین می شود. لازم به ذکر است این روش به عنوان پیش فرض عملگر انتخاب در نظر گرفته شده است.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

```
expectation = expectation(:,1);
wheel = cumsum(expectation) / nParents;
 parents = zeros(1,nParents);
 % we will step through the wheel in even steps.
stepSize = 1/nParents;
 % we will start at a random position less that one full step
position = rand * stepSize;
 % a speed optimization. Position is monotonically rising.
lowest = 1;
 for i = 1:nParents % for each parent needed,
    for j = lowest:length(wheel) % find the wheel position
        if(position < wheel(j)) % that this step falls in.
            parents(i) = j;
            lowest = j;
            break;
        end
    end
    position = position + stepSize; % take the next step.
end
```

فراخوانی مقدار تابع هدف مقیاس شده

محاسبه مقدار تجمعی تابع هدف مقیاس شده

تعریف اندازه گام

تعریف موقعیت

تا زمانی که موقعیت کروموزم از مقدار تابع هدف مقیاس شده آن کمتر است، آن کروموزم انتخاب می شود.

افزایش موقعیت جهت انتخاب سایر کروموزم با توجه به اندازه گام تعریف شده

بر اساس مثال Example1_GA و تابع selectionstochunif (واقع در مسیر C:\Program Files\MATLAB\R2022b\toolbox\globaloptim\globaloptim) می توان با روند این عملگر انتخاب آشنا شد.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

نحوه عملکرد عملگر انتخاب باقیمانده (selectionremainder): در این روش به تعداد مقادیر صحیح تابع هدف هر کروموزم، آن کروموزم جهت ایجاد نسل بعد انتخاب می شود. برای کروموزم ها با تابع هدف غیرصحیح، بر اساس مقیاس سازی عمل می شود.

```
parents = zeros(1,nParents);
expectations = expectations(:,1);
% First we assign the integral parts deterministically.
% Load up the sure parents and leave the fractional remainder in newScores.
next = 1;
for i = 1:length(expectations)
   while(expectations(i) >=1)
      parents(next) = i;
      next = next + 1;
      expectations(i) = expectations(i) - 1;
   end
end
 % if all newScores were integers, we are done!
if(next > nParents)
   return
end
```

انتخاب کروموزم های حاوی تابع هدف مقیاس شده صحیح به تعداد عدد صحیح موجود در تابع هدف

بر اساس مثال Example1_GA و تابع selectionremainder (واقع در مسیر C:\Program Files\MATLAB\R2018b\toolbox\globaloptim\globaloptim) می توان با روند این عملگر انتخاب آشنا شد.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

انتخاب کروموزم های باقیمانده حاوی تابع هدف غیرعدد صحیح به صورت تصادفی و بر مبنای مقیاس نمودن تابع هدف

نحوه عملکرد عملگر انتخاب یکنواخت (selectionuniform): در این روش به صورت تصادفی از کروموزم های والد جهت تولید نسل بعدی استفاده می شود. این روش امکان جستجوی مناسبی را در فضای موجه فراهم نمی کند.

```
% scale the remaining scores to be probabilities...
intervals = cumsum(expectations);
intervals = intervals / intervals(end);
 % take the rest by chance.
for k = next:nParents
   r = rand;
   for i = 1:length(expectations)
      if(r <= intervals(i))
         parents(k) = i;
         % make sure this one doesn't get picked again
         expectations(i) = 0;
         intervals = cumsum(expectations);
         if(intervals(end) ~= 0)
            intervals = intervals / intervals(end);
         end
         break;
      end
   end
end
```

```
expectation = expectation(:,1);
% nParents random numbers
parents = rand(1,nParents);
 % integers on the interval [1, populationSize]
parents = ceil(parents * length(expectation));
```

```
options = optimoptions('ga','SelectionFcn','selectionuniform');
```

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

نحوه عملکرد عملگر انتخاب چرخ گردان (selectionroulette): در این روش بر مبنای رویکرد چرخ گردان اقدام به تولید کروموزم های نسل بعد می نماید. برای این منظور لازم است از دستور زیر استفاده شود:

```
options = optimoptions('ga','SelectionFcn','selectionroulette');
```

```
function parents = selectionroulette(expectation,nParents,options)
 expectation = expectation(:,1);
wheel = cumsum(expectation) / nParents;
 parents = zeros(1,nParents);
for i = 1:nParents
    r = rand;
    for j = 1:length(wheel)
        if(r < wheel(j))
            parents(i) = j;
            break;
        end
    end
end
```

فراخوانی مقدار تابع هدف مقیاس شده

محاسبه مقدار تجمعی تابع هدف مقیاس شده

نحوه عملکرد عملگر انتخاب تورنامنت (selectiontournament): در این روش با استفاده از رویکرد تورنامنت اقدام به تولید کروموزم های نسل بعد می نماید. با توجه به اینکه در این روش، کروموزم ها به چندین دسته تقسیم می شوند و بهترین مقدار هر دسته در فرآیند تولید نسل بعد انتخاب می گردد، لذا ورود تعداد دسته ها (size) در این روش ضروری است:

```
options = optimoptions('ga','SelectionFcn',{@selectiontournament,size})
```

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

```
function parents = selectiontournament(expectation,nParents,options,tournamentSize)
if nargin < 4 || isempty(tournamentSize)
    tournamentSize = 4;
end
 playerlist = ceil(size(expectation,1) * rand(nParents,tournamentSize));
parents = tournament(playerlist,expectation);
function champions = tournament(playerlist,expectation)
playerSize = size(playerlist,1);
champions = zeros(1,playerSize);
for i = 1:playerSize
    players = playerlist(i,:);
    winner = players(1);
    for j = 2:length(players)
        score1 = expectation(winner,:);
        score2 = expectation(players(j),:);
        if score2(1) > score1(1)
            winner = players(j);
        elseif score2(1) == score1(1) && ... (length(score1) > 1 && score2(2) > score1(2))
            winner = players(j);
        end
    end
    champions(i) = winner;
end
```

انتخاب کروموزم ها به صورت تصادفی و با توجه به تعداد دسته های در نظر گرفته شده

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

در صورتی که مسأله بهینه سازی از نوع Problem-Base باشد، لازم است از دستور زیر جهت حداقل نمودن مقدار تابع هدف برای تعیین مقدار بهینه متغییرهای تصمیم با استفاده از الگوریتم GA استفاده نمود:

x = ga(problem)

در probem لازم است پارامترهای جدول زیر همانند problem.options، problem.solver، problem.fitnessfcn، problem.nvars و ... تعریف شود.

جدول پارامترهای موردنیاز جهت تعریف مسأله بهینه سازی و معرفی آن به مدل GA

fitnessfcn	Fitness functions
nvars	Number of design variables
Aineq	A matrix for linear inequality constraints
Bineq	b vector for linear inequality constraints
Aeq	Aeq matrix for linear equality constraints
Beq	beq vector for linear equality constraints
lb	Lower bound on x
ub	Upper bound on x
nonlcon	Nonlinear constraint function
rngstate	Optional field to reset the state of the random number generator
solver	'ga'
options	Options created using optimoptions or exported from the Optimization app

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

- با انتخاب 'UseVectorized' در Name و 'true' در Value می توان در صورتی که متغییر تصمیم به صورت برداری باشد، محاسبات را به صورت برداری انجام داد.

پس از اجرای الگوریتم بهینه سازی ژنتیک، شش خروجی در اختیار کاربر قرار می گیرد:

[x,fval,exitflag,output,population,scores]=ga(fun,nvars,A,b,Aeq,beq,lb,ub,nonlcon,IntCon,options)

x: بهترین مقدار متغییرهای تصمیم را به صورت یک بردار نشان می دهد.

fval: مقدار تابع هدف متناسب با بهترین مقدار متغییر تصمیم (x) را ارائه می نماید.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

exitflag: دلایل توقف اجرای الگوریتم ژنتیک به صورت یک عدد صحیح نمایش داده می شود. تفسیر این اعداد به صورت زیر است:

- **عدد یک**: در صورتی که مسأله بدون محدودیت غیرخطی باشد. علت توقف: کمتر بودن متوسط تغییرات تجمعی تابع هدف در طی MaxStallGenerations تکرار از مقدار FunctionTolerance و میزان تخطی محدودیت ها نیز از ConstraintTolerance کمتر است.
- **عدد یک**: در صورتی که مسأله دارای محدودیت غیرخطی است. علت توقف: کمتر بودن شاخص complementarity measure (حاصلضرب مقدار عددی محدودیت غیرخطی نامساوی در ضریب لاگرانژ ($C_i\lambda_i$)) از جذر ConstraintTolerance، متوسط تغییرات تجمعی تابع هدف در طی MaxStallGenerations تکرار از مقدار FunctionTolerance و میزان تخطی محدودیت ها نیز از ConstraintTolerance کمتر است.
- **عدد سه**: مقدار تابع هدف در طی MaxStallGenerations بدون تغییر بوده و FunctionTolerance و میزان تخطی محدودیت ها نیز از ConstraintTolerance کمتر است.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

- **عدد چهار**: میزان تغییرات متغییر تصمیم جهت بررسی وضعیت جواب بهینه بسیار کوچک شده و میزان تخطی محدودیت ها نیز از ConstraintTolerance کمتر است.
- **عدد پنج**: مقدار تابع هدف از مقدار تعریف شده در FitnessLimit (که به طور پیش فرض، منفی بی نهایت است) و میزان تخطی محدودیت ها نیز از ConstraintTolerance کمتر است.
- **عدد صفر**: تعداد تکرارها از مقدار تعریف شده در MaxGenerations بیشتر شده است.
- **عدد ۲-**: هیچ جواب موجه ایی پیدا نشده است.
- **عدد ۴-**: مدت زمانی که مقدار تابع هدف بهبودی نداشته است از مقدار مدت زمان تعیین شده در MaxStallTime بیشتر است.
- **عدد ۵-**: مدت زمان اجرای برنامه بهینه سازی از مقدار مدت زمان تعیین شده در MaxTime بیشتر شده است.

scores: مقدار تابع هدف متناظر با نسل برتر را به صورت یک بردار ستونی ارائه می دهد.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

output: اطلاعاتی در خصوص روند دستیابی به مقدار بهینه در اختیار کاربر قرار می دهد. این اطلاعات به شرح زیر می باشند:

- Problemtype: نوع مسأله حل شده را (بدون محدودیت (unconstrained)، با محدودیت بر روی متغییرهای تصمیم (boundconstraints)، محدودیت های خطی (linearconstraints)، محدودیت های غیرخطی (nonlinearconstr) و محدودیت عدد صحیح (integerconstraints)) نشان می دهد.
- Generations: تعداد تکرار انجام شده را نشان می دهد.
- Funccount: تعداد بار محاسبه تابع هدف را ارائه می نماید.
- Message: دلایل توقف الگوریتم بهینه سازی نشان داده می شود.
- Maxconstraint: حداکثر میزان تخطی از محدودیت های ارائه شده را در صورت وجود نشان می دهد.

population: برترین نسل حاوی کروموزم های بهینه را در قالب ماتریسی با ابعاد اندازه جمعیت (PopulationSize) در تعداد متغییر تصمیم نشان می دهد.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

مثال) مسأله بهینه سازی زیر را با استفاده از الگوریتم ژنتیک حل نمائید.

$$Minimize\ f(x) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$$

$x_1 . x_2 + x_1 - x_2 + 1.5 \leq 0$

$10 - x_1 . x_2 \leq 0$

$0 \leq x_1 \leq 1$

$0 \leq x_2 \leq 13$

```
clc
clear
ObjectiveFunction = @simple_fitness;
nvars = 2;    % Number of variables
LB = [0 0];   % Lower bound
UB = [1 13];  % Upper bound
ConstraintFunction = @simple_constraint;
X0 = [0.5 0.5]; % Start point (row vector)
options = optimoptions('ga','PlotFcn',{@gaplotbestf,@gaplotmaxconstr},...
    'Display','iter','ConstraintTolerance',1e-7,'InitialPopulationMatrix',X0);
[x,fval,exitflag,output,population,scores]=ga(ObjectiveFunction,nvars,[],[]
,...
    [],[],LB,UB,ConstraintFunction,options)
```

تعریف تابع هدف

تعریف تعداد متغییرهای تصمیم

تعریف محدودیت های غیرخطی

تعریف مقدار اولیه

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

در صورت وجود پارامترهای ثابت در تابع هدف، می توان تابع هدف را به صورت پارامتری به الگوریتم ژنتیک معرفی نمود.

مثال) مسأله بهینه سازی زیر را با استفاده از الگوریتم ژنتیک حل نمائید.

$$Minimize\ f(x) = a(x_1^2 - x_2)^2 + (b - x_1)^2$$

$$-3 \leq x_1 \leq 3$$

$$-3 \leq x_2 \leq 3$$

تعریف تابع هدف به صورت پارامتری

```
function y = parameterized_fitness(x,p1,p2)
y = p1 * (x(1)^2 - x(2)) ^2 + (p2 - x(1))^2;
```

```
clc
clear
a = 100;
b = 2; % define constant values
FitnessFunction = @(x) parameterized_fitness(x,a,b);
numberOfVariables = 2;
lb = [-3,-3];
ub = [3,3];
[x,fval] = ga(FitnessFunction,numberOfVariables,[],[],[],[],lb,ub)
```

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

جهت افزایش سرعت سازی می توان تابع هدف را به صورت برداری تهیه نمود. در واقع در این روش، به طور همزمان مقدار تابع هدف تمامی کروموزم ها مورد محاسبه قرار می گیرد. برای این منظور لازم است مراحل زیر بر روی متغییرهای تصمیم و تنظیمات دستور ga اعمال شود:

- تبدیل هر یک از متغییرهای تصمیم به مقادیر برداری. به عبارت دیگر متغییر $x(i)$ به $x(:,i)$ تبدیل می شود. در واقع ردیف ها بیانگر مقدار متغییر تصمیم iام در تمامی کروموزم ها می باشد.
- تبدیل عملگر های ضرب، توان و تقسیم از *، ^ و / به * . ، ^. و /. جهت محاسبات برداری متغییرها
- تبدیل UseVectorized به true در options. (options = optimoptions(@ga,'UseVectorized',true);)

مثال) مسأله بهینه سازی زیر را با استفاده از الگوریتم ژنتیک کلاسیک و برداری حل نمائید و زمان اجرای آن ها را با هم مقایسه نمائید.

$$Minimize\ f(x) = a(x_1^2 - x_2)^2 + (b - x_1)^2$$

$$-3 \le x_1 \le 3$$

$$-3 \le x_2 \le 3$$

تعریف تابع هدف پارامتری به صورت برداری

```
function y = vectorized_fitness(x,p1,p2)
y = p1 * (x(:,1).^2 - x(:,2)).^2 + (p2 - x(:,1)).^2;
```

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

```
clc
clear
VFitnessFunction = @(x)vectorized_fitness(x,100,1);
FitnessFunction = @(x)parameterized_fitness(x,100,1);
lb = [-3,-3];
ub = [3,3];
numberOfVariables = 2;
tic
options = optimoptions(@ga,'UseVectorized',true,'PopulationSize’,10000);
[x,fval] = ga(VFitnessFunction,numberOfVariables,[],[],[],[],lb,ub,[],options);
% options = optimoptions('ga','PopulationSize’,10000);
% [x,fval] = ga(FitnessFunction,numberOfVariables,[],[],[],[],lb,ub,[],options);
toc
```

در صورتی که محدودیت های غیرخطی در مسأله بهینه سازی وجود داشته باشند، جهت اجرای برنامه به صورت برداری، لازم است این محدودیت ها نیز به صورت برداری ارائه شوند. برای محدودیت های خطی در حل به صورت برداری، دستور ga به صورت خودکار از شکل برداری آن ها استفاده می کند.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

مثال) مسأله بهینه سازی زیر را با استفاده از الگوریتم ژنتیک حل نمائید.

$$Minimize\ f(x) = a(x_1^2 - x_2)^2 + (b - x_1)^2$$

$$x_1^2 + x_2^2 \leq 1$$

$$-3 \leq x_1 \leq 3$$

$$-3 \leq x_2 \leq 3$$

تعریف محدودیت غیرخطی به صورت برداری

```
function [c ceq] =nonlcon(x)
c(:,1)=x(:,1).^2+x(:,2).^2-1;
ceq=[];
```

```
clc
clear
VFitnessFunction = @(x)vectorized_fitness(x,100,1);
lb = [-3,-3];
ub = [3,3];
numberOfVariables = 2;
options = optimoptions(@ga,'UseVectorized',true,'PopulationSize',1000);
[x,fval] =
ga(VFitnessFunction,numberOfVariables,[],[],[],[],lb,ub,@nonlcon,options)
```

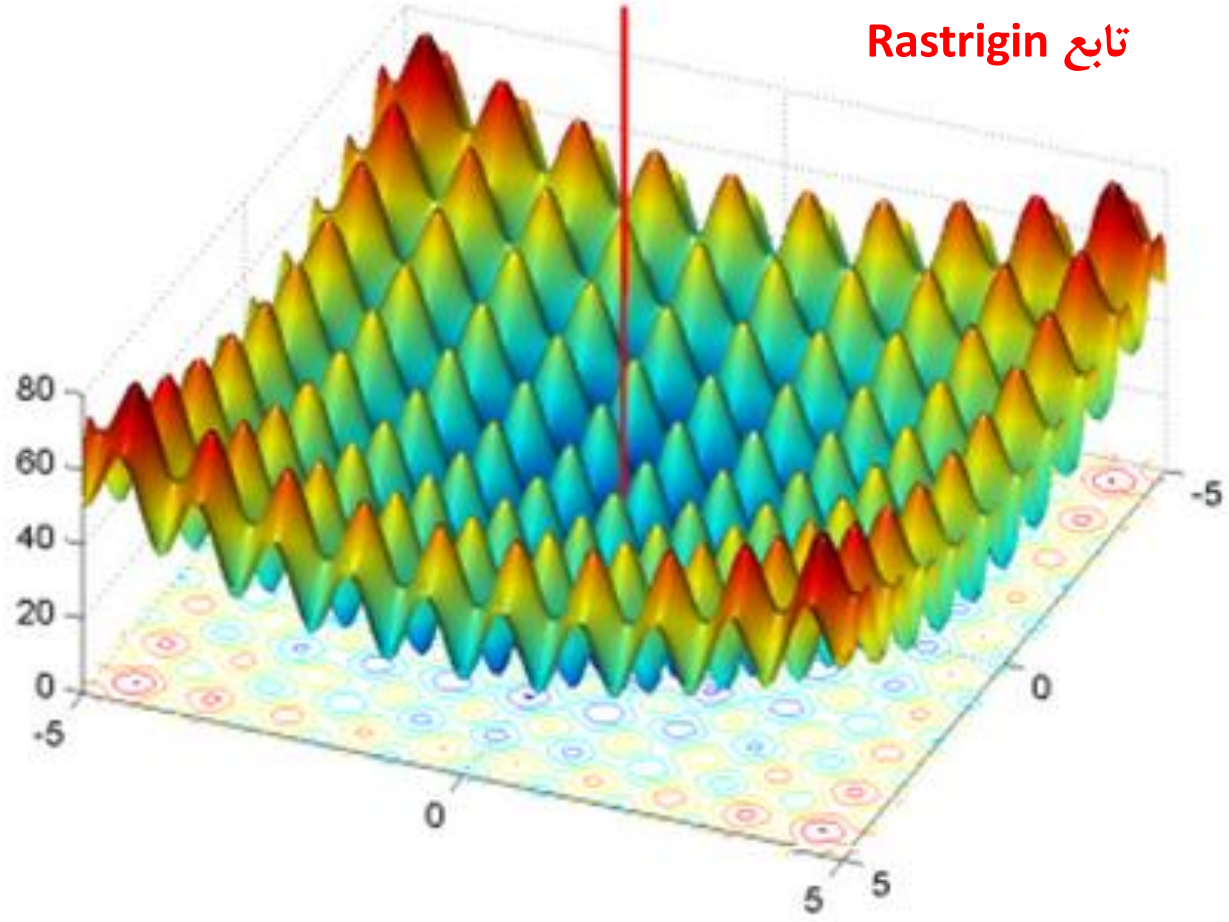
معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

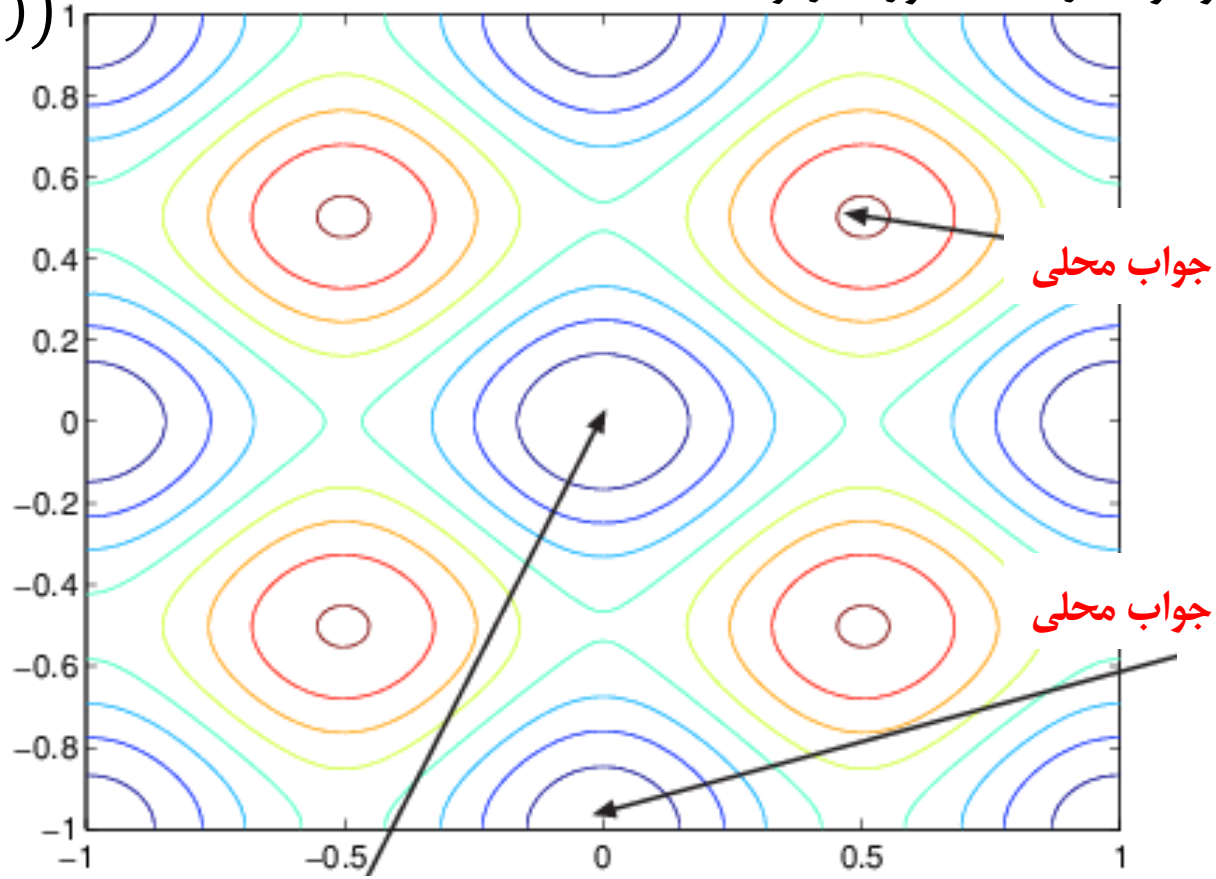
اثرات پارامترهای مختلف مؤثر در الگوریتم ژنتیک بر روی جواب بهینه:

جهت بررسی اثر پارامترهای مختلف مؤثر در الگوریتم ژنتیک بر روی تعیین جواب بهینه کلی از تابعی با مشخصات ریاضی زیر که حاوی تعداد قابل توجهی جواب محلی است، استفاده می شود. شکل این تابع برای دو متغییر تصمیم، که قابلیت افزایش تعداد متغییرهای تصمیم وجود دارد، به صورت زیر است:

$$f(x) = 20 + x_1^2 + x_2^2 - 10\big(cos(2\pi x_1) + cos(2\pi x_2)\big)$$

تابع Rastrigin





جواب کلی

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

۱- اثرات تعداد تکرار بر روی جواب بهینه محلی:

با توجه به اینکه مقدار پیش فرض تعداد تکرار در الگوریتم ژنتیک، ۱۰۰ برابر تعداد متغییرهای تصمیم است و همچنین در صورت عدم تغییر مقدار تابع هدف در طی ۵۰ تکرار متوالی، فرآیند بهینه سازی متوقف می شود، لذا لازم است این دو پارامتر یعنی MaxGenerations (حداکثر تعداد تکرار مجاز) و MaxStallGenerations (حداکثر تعداد تکرار مجاز در شرایطی که تغییری در تابع هدف ایجاد نمی شود) تغییر داده شوند.

لازم به ذکر است با توجه به اینکه تابع عملگر جهش پیش فرض از نوع جهش زنگوله ای است لذا افزایش تعداد تکرار مجاز ممکن است منجر به بهبود مقدار تابع هدف نشود. علت این امر در رابطه زیر مشخص است که با افزایش تعداد تکرار، متوسط مقدار جهش در هر تکرار و به عبارت دیکر تنوع کروموزم ها کاهش می یابد.

$$\sigma_k = \sigma_{k-1}\left(1 - Shrink\frac{k}{Generations}\right)$$

پارامتر Shrink، میزان جمع شدگی انحراف معیار (σ_k) در طی تکرارهای مختلف را تعیین می کند. مقدار انحراف معیار بر اساس بازه تغییرات ژن ها و به صورت Scale*(v(2)-v(1)) محاسبه می شود. در این رابطه v(2) و v(1) به ترتیب حدود بالا و پایین هر ژن می باشد.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

```
clc
clear
% load x;
%% Default options
options = optimoptions('ga','Display','iter','PlotFcn',@gaplotbestf);
x = ga(@rastriginsfcn,5,[],[],[],[],[],[],[],options);

%% Limit the MaxGenerations and MaxStallGenerations
% options = optimoptions('ga','Display','iter','PlotFcn',@gaplotbestf,...
%     'MaxGenerations',500,'MaxStallGenerations',Inf);
% x = ga(@rastriginsfcn,5,[],[],[],[],[],[],[],options);

%% Use InitialPopulationMatrix in optimization process
% options = optimoptions('ga','Display','iter','PlotFcn',@gaplotbestf,...
%     'MaxGenerations',500,'MaxStallGenerations',Inf,'InitialPopulationMatrix',x);
% x = ga(@rastriginsfcn,5,[],[],[],[],[],[],[],options);
% save x x;
```

استفاده از مقادیر پیش فرض جهت حل مسأله

تغییر در مقادیر مجاز تعداد تکرار و تعداد تکراری که الگوریتم علی رغم ثابت بودن تابع هدف می‌تواند ادامه یابد.

استفاده از آخرین مقدار بهینه تعیین شده به عنوان مقدار اولیه متغییرهای تصمیم جهت ادامه فرآیند بهینه سازی

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

۲- اثرات تنوع جمعیت کروموزم ها بر روی جواب بهینه محلی (Population Diversity):

یکی از پارامترهای مهم در تعیین عملکرد الگوریتم ژنتیک، تنوع جمعیت در کروموزم ها است. در صورتی که متوسط فاصله بین کروموزم ها زیاد باشد، گفته می شود که تنوع کروموزم ها بالا است و در صورت کم بودن متوسط فاصله بین کروموزم ها، تنوع پایین در نظر گرفته می شود. اعمال مقدار صحیح تنوع در الگوریتم ژنتیک بسیار مهم و کلیدی است. **اگر تنوع کروموزم ها خیلی زیاد و یا خیلی کم باشد، الگوریتم ژنتیک ممکن است بدرستی عمل نکند.** برای این منظور دو پارامتر بازه تغییرات اولیه ژن های کروموزم (Initial range) و مقدار جهش که بر روی تنوع کروموزم ها تأثیرگذار می باشند، مورد بررسی قرار می گیرد.

به طور معمول الگوریتم ژنتیک، نسل اولیه را بر مبنای بازه تغییرات ژن ها به صورت تصادفی تولید می کند. در صورتی که تنوع در جمعیت کروموزم ها به اندازه کافی وجود داشته باشد، الگوریتم ژنتیک می تواند، حتی بر اساس کروموزم هایی که حاوی ژن هایی خارج از بازه تغییرات اولیه ژن ها می باشند، اقدام به تعیین جواب بهینه نماید.

جهت بررسی اثرات بازه تغییرات اولیه مقادیر ژن ها بر روی تنوع و مقدار جواب بهینه از تابع Rastrigin با دو متغییر استفاده شده است.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

مثال) تابع Rastrigin را با اعمال دو متغییر تصمیم با استفاده از الگوریتم ژنتیک حل نمائید.

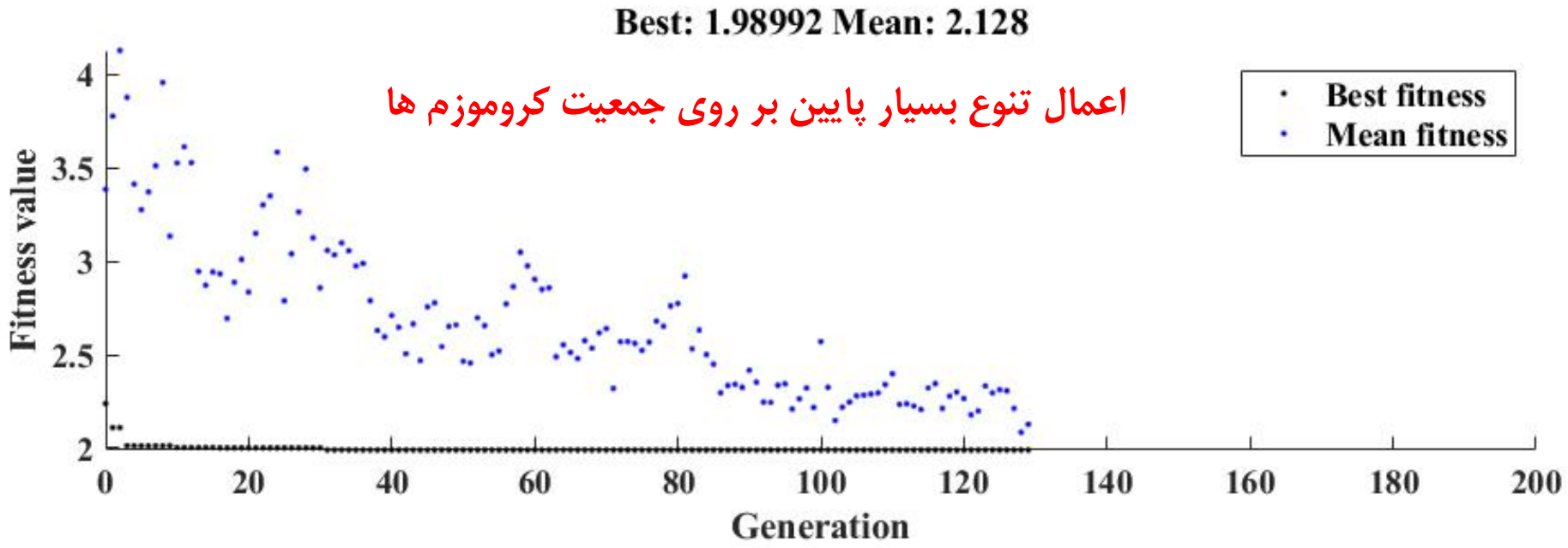
```
clc
clear
%% Too little diversity
options = optimoptions('ga','Display','iter','PlotFcn',{@gaplotbestf,...
    @gaplotdistance},'InitialPopulationRange',[1;1.1]);
x = ga(@rastriginsfcn,2,[],[],[],[],[],[],[],options);

%% Suitable diversity
% options = optimoptions('ga','Display','iter','PlotFcn',{@gaplotbestf,...
%     @gaplotdistance},'InitialPopulationRange',[1;100]);
% x = ga(@rastriginsfcn,2,[],[],[],[],[],[],[],options);

%% Too little diversity
% options = optimoptions('ga','Display','iter','PlotFcn',{@gaplotbestf,...
%     @gaplotdistance},'InitialPopulationRange',[1;1.5]);
% x = ga(@rastriginsfcn,2,[],[],[],[],[],[],[],options);
```

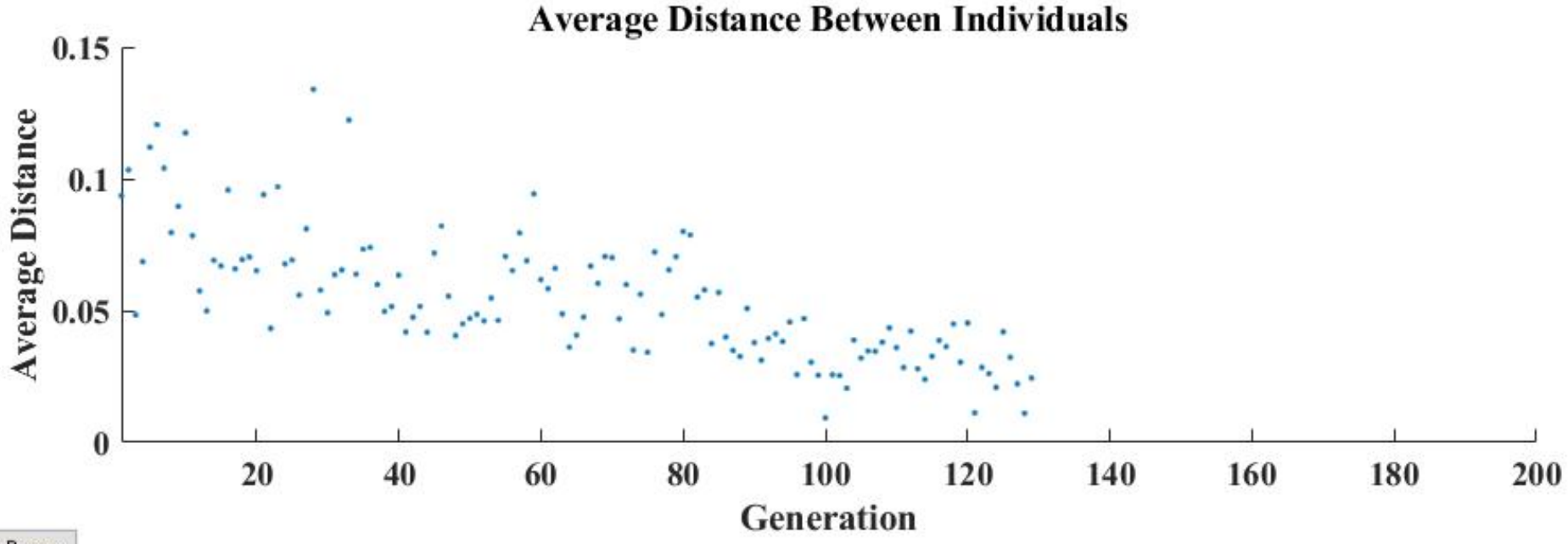
اعمال تنوع بسیار پایین بر روی جمعیت کروموزم ها

اعمال تنوع بسیار بالا بر روی جمعیت کروموزم ها

اعمال تنوع مناسب بر روی جمعیت کروموزم ها

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

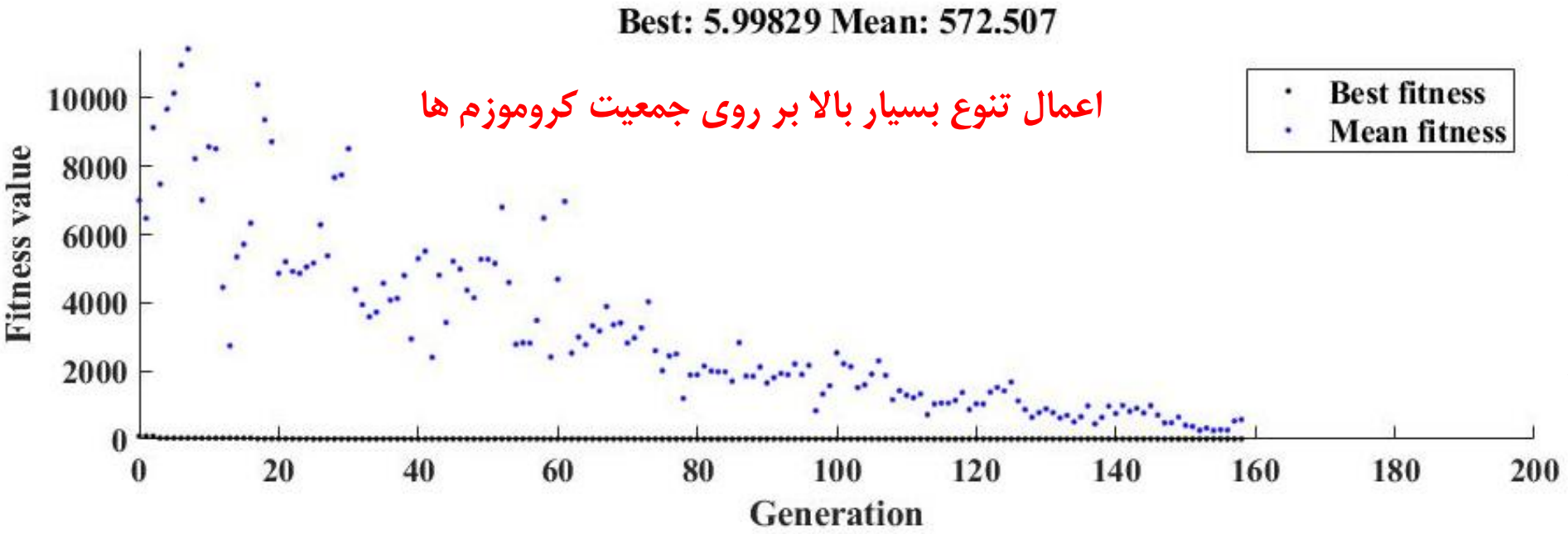




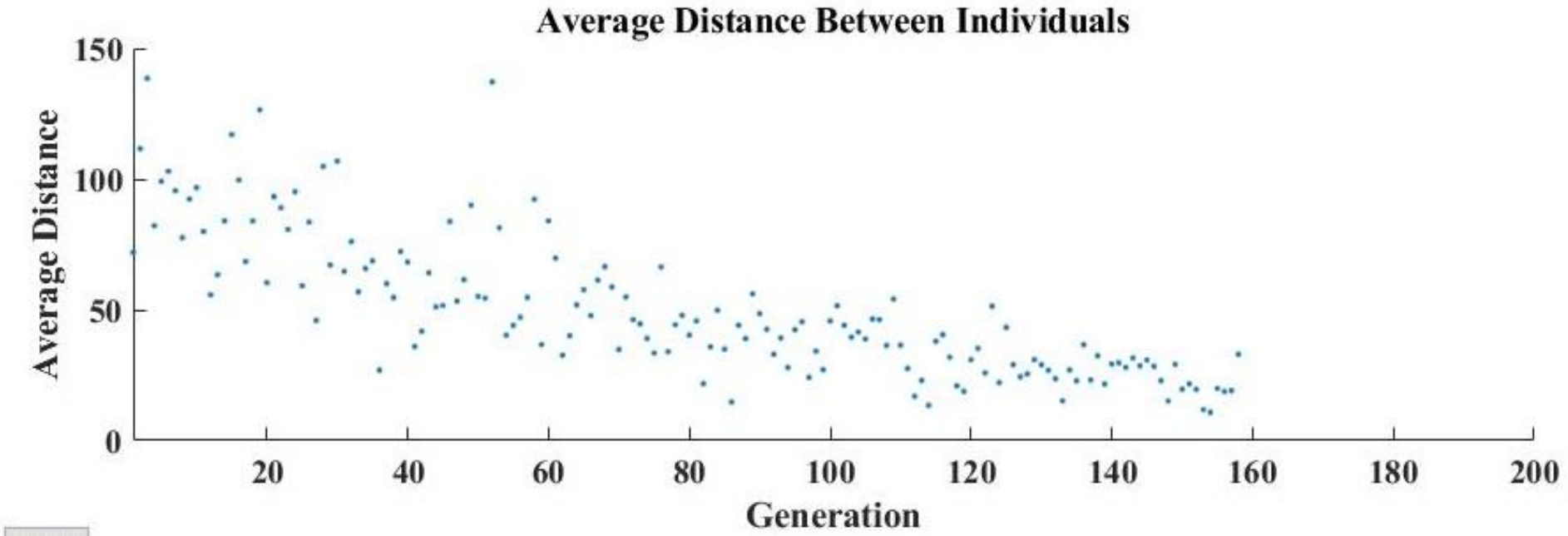
با توجه به فاصله بین کروموزم ها در هر تکرار می توان دریافت که به دلیل تنوع بسیار پایین مقادیر ژن ها، جواب بهینه بدست آمده فاصله زیادی با مقدار بهینه کلی (صفر) دارد.

این امر در نتیجه پایین بودن بازه تغییرات نسل اولیه تولید شده که بین ۱ و ۱/۱ است، می باشد.

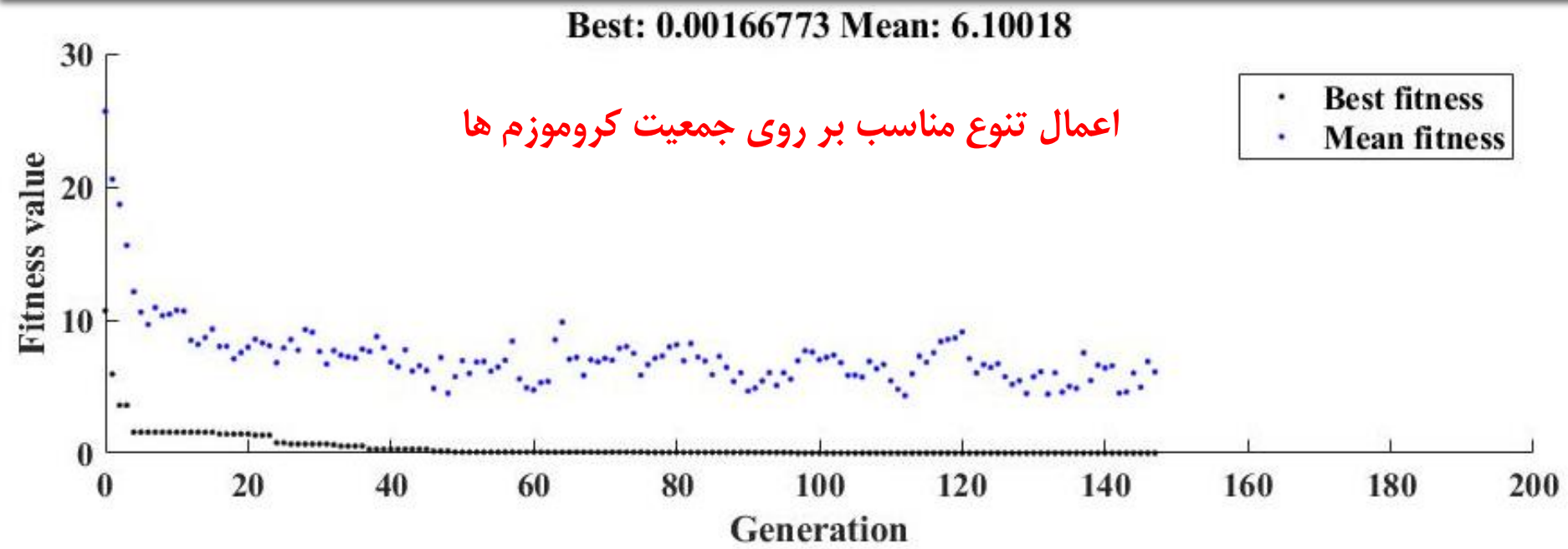
معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

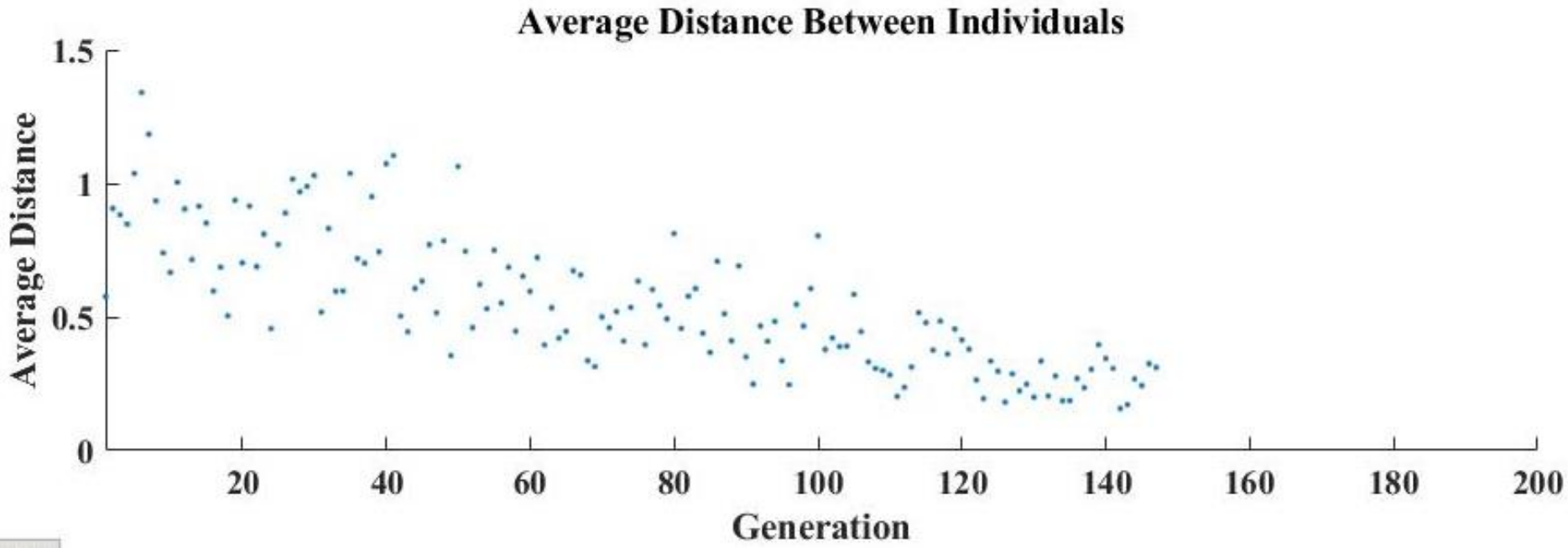


به دلیل فاصله زیاد بین کروموزم ها در هر تکرار، تنوع بسیار بالا در جمعیت کروموزم ها حاصل شده است. این تنوع بسیار بالا نیز منجر به دور شدن جواب بهینه بدست آمده از مقدار بهینه کلی (صفر) دارد.



معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن





اعمال تنوع مناسب به جمعیت اولیه کروموزم ها منجر به تولید جواب بهینه نزدیک به بهینه کلی شده است.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

۳- اثرات مقیاس نمودن تابع هدف بر روی جواب بهینه محلی (Fitness Scores):

با توجه به اینکه روش های مقیاس سازی تابع هدف، اقدام به ایجاد تغییرات در تابع هدف اولیه می نمایند لذا انتخاب صحیح نوع روش مقیاس سازی بر سرعت اجرای الگوریتم ژنتیک و همچنین نحوه جستجوی فضای موجه تأثیر زیادی دارد. بر این اساس لازم است حساسیت مدل بهینه سازی تهیه شده به این روش ها که مشتمل بر fitscalingrank، fitscalingprop، fitscalingtop و fitscalingshiftlinear می باشد، مورد بررسی قرار گیرد.

مثال) تابع Rastrigin را با اعمال چهار متغییر تصمیم با استفاده از الگوریتم ژنتیک حل نمائید.

```
clc
clear
options = optimoptions('ga','Display','iter','PlotFcn',{@gaplotbestf,...
   @gaplotdistance},'FitnessScalingFcn',{@fitscalingshiftlinear,40});

% options = optimoptions('ga','Display','iter','PlotFcn',{@gaplotbestf,...
%    @gaplotdistance},'FitnessScalingFcn','fitscalingrank');

% options = optimoptions('ga','Display','iter','PlotFcn',{@gaplotbestf,...
%    @gaplotdistance},'FitnessScalingFcn','fitscalingprop');
 %
% options = optimoptions('ga','Display','iter','PlotFcn',{@gaplotbestf,...
%    @gaplotdistance},'FitnessScalingFcn','fitscalingtop');

x=ga(@rastriginsfcn,4,[],[],[],[],[],[],[],options);
```

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

۴- اثرات عملگر جهش بر روی جواب بهینه محلی:

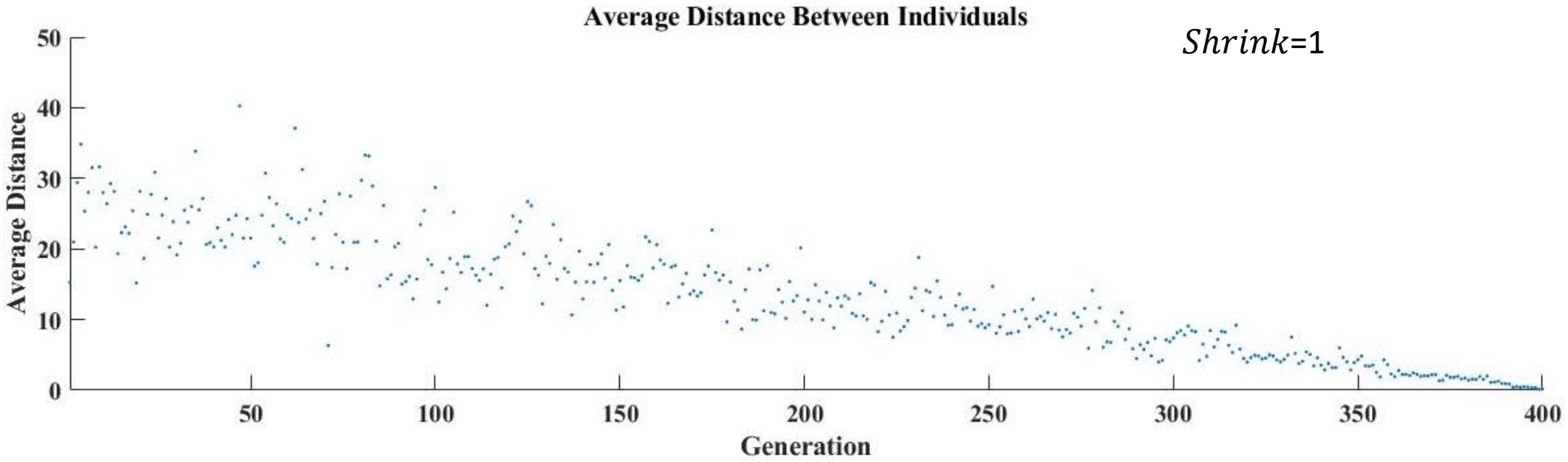
عملگر جهش منجر به ایجاد تغییرات در مقادیر ژن ها می شود. با توجه به اینکه مقدار پیش فرض این عملگر، که از نوع جهش زنگوله ای است، انتخاب صحیح دو پارامتر Scale و Shrink در دستیابی به جواب بهتر بسیار مؤثر خواهد بود. پارامتر Scale با ضرب شدن در بازه تغییرات مجاز ژن ها، انحراف معیار جهش را در اولین تکرار تعیین می کند. همچنین پارامتر Shrink مقدار متوسط کاهش جهش را مطابق رابطه زیر کنترل می کند. مطابق این رابطه مقدار انحراف معیار در طی تکرارهای مختلف به صورت خطی کاهش می یابد به طوری که در آخرین تکرار، مقدار $(1 - Shrink)$ در انحراف معیار ضرب می شود. با توجه به اینکه مقدار پیش فرض این دو پارامتر برابر با یک است، عملاً در آخرین تکرار، جهش انجام نمی شود.

$$\sigma_k = \sigma_{k-1}\left(1 - Shrink\frac{k}{Generations}\right)$$

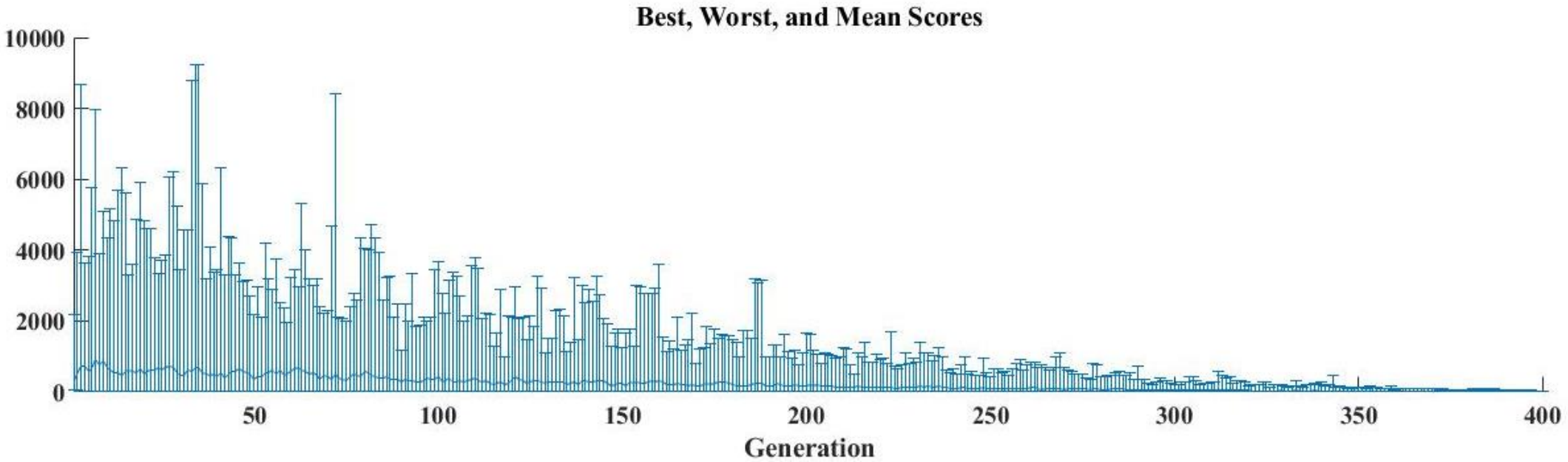
مثال) تابع Rastrigin را با اعمال چهار متغیر تصمیم با استفاده از الگوریتم ژنتیک و با اعمال مقادیر یک و ۰/۳ برای پارامتر Shrink حل نموده و نتایج را مورد مقایسه قرار دهید.

```
options = optimoptions('ga','Display','iter','PlotFcn',{@gaplotdistance,@gaplotrange},...
   'MaxStallGenerations',200);
x=ga(@rastriginsfcn,4,[],[],[],[],[],[],[],options);
```

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

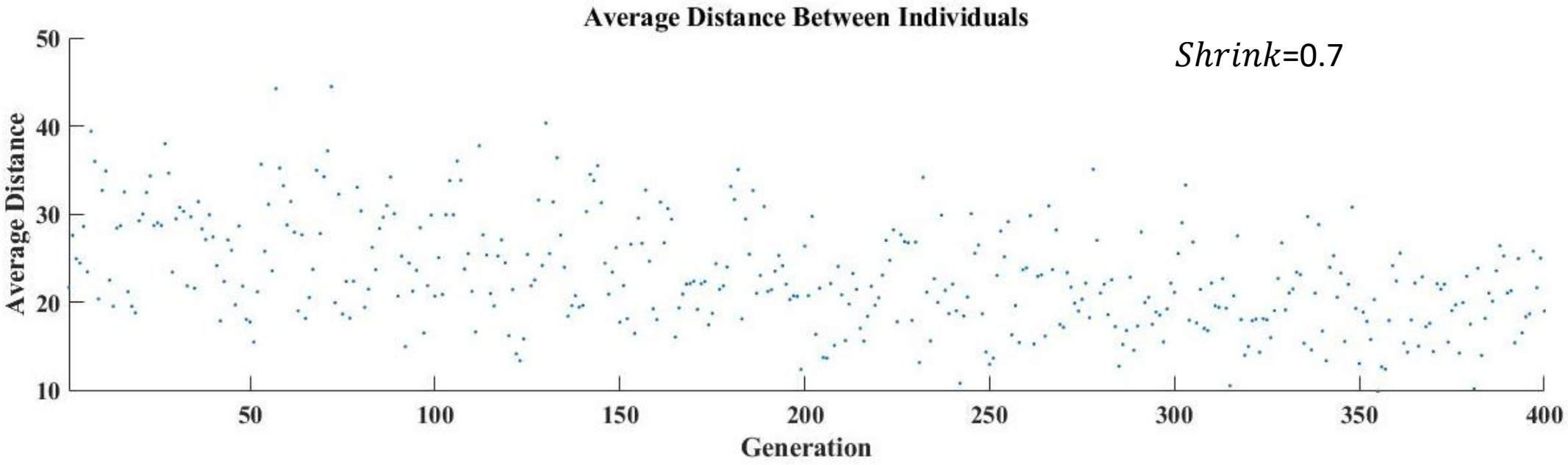


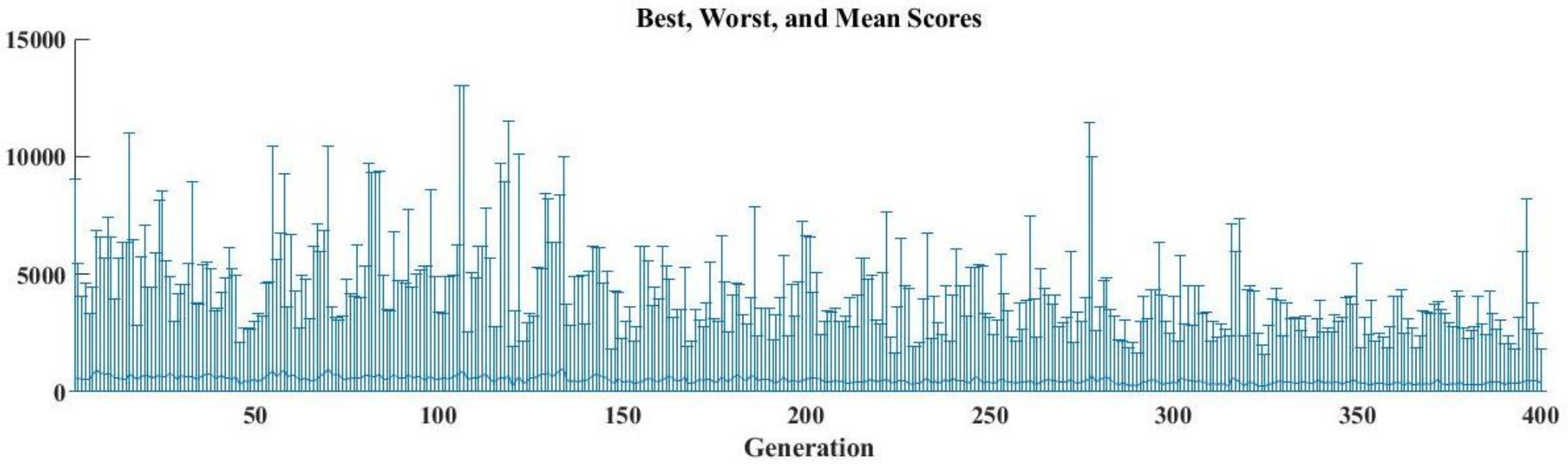
$Shrink$=1



مطابق این شکل، متوسط فاصله بین کروموزم ها با افزایش تکرارها کاهش می یابد به طوری که در آخرین تکرار به صفر می رسد. بر این اساس تغییرات تابع هدف نیز کاهش می یابد.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن





مقایسه این حالت با شکل قبل نشان می دهد که متوسط فاصله بین کروموزم ها با افزایش تکرارها کمتر کاهش می یابد. لذا تابع هدف در طی تکرارهای مختلف تغییراتی را تجربه می نماید.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

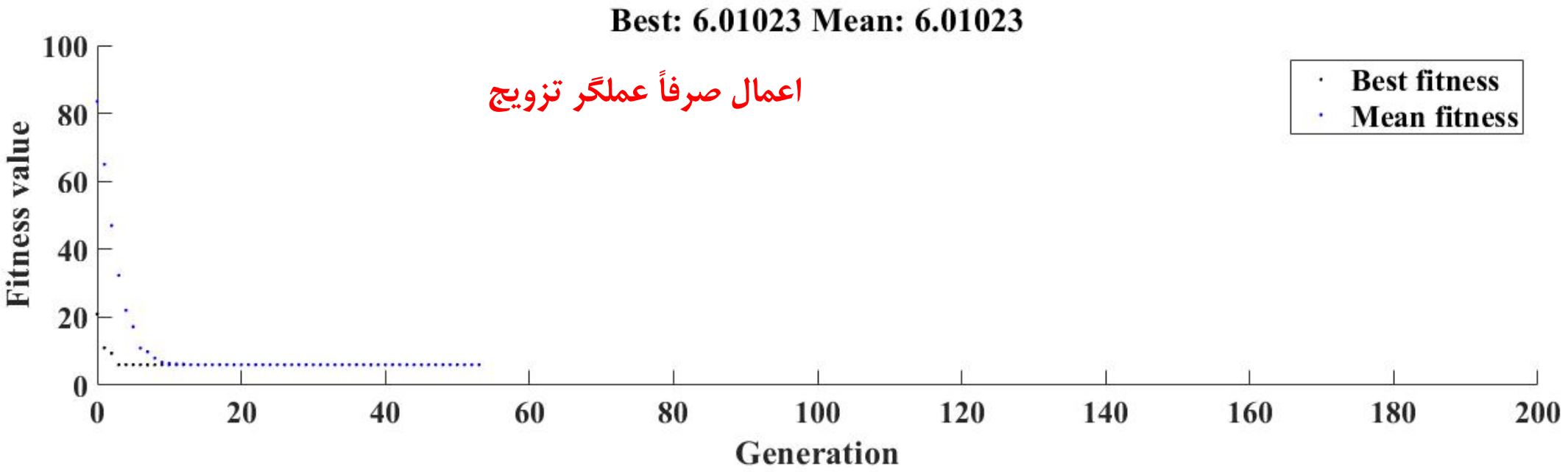
۵- اثرات عملگر تزویج بر روی جواب بهینه محلی:

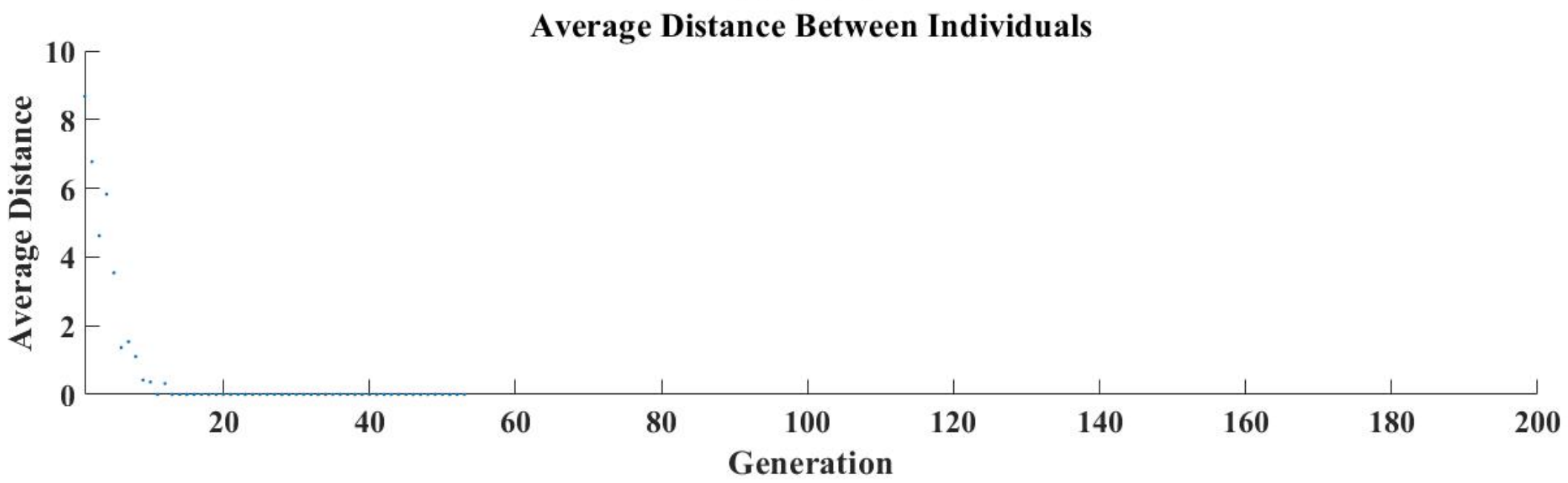
یکی از پارامترهای مهم و اثرگذاری بر روی جواب بهینه، پارامتر CrossoverFraction است. این پارامتر که تعیین کننده نحوه تولید کروموزم های در طی تکرارهای مختلف است، بیان می کند که چند درصد از جمعیت نسل بعد توسط عملگر تزویج تولید شوند. مقدار پیش فرض این پارامتر ۰/۸ است. در صورت انتخاب مقدار یک برای این پارامتر، عملیات جهش انجام نمی شود و انتخاب صفر به منزله انجام صرفاً عملیات جهش بر روی کروموزم ها می باشد.

مثال) تابع Rastrigin را با اعمال دو متغییر تصمیم با استفاده از الگوریتم ژنتیک تحت شرایطی که عملگر جهش و تزویج هر کدام به تنهایی اعمال شود، حل نمائید. همچنین بهترین مقدار پارامتر CrossoverFraction را بر اساس تحلیل حساسیت تعیین نمائید.

بر مبنای نتایج بدست آمده از این مثال می توان دریافت که عدم اعمال هر یک از دو عملگر تزویج و جهش می تواند به ترتیب منجر به عدم ترکیب ژن های بهبود یافته با سایر ژن های موجود جهت ایجاد کروموزم هایی با مقادیر بهتر تابع هدف و عدم ایجاد تغییرات در مقدار تابع هدف به دلیل عدم تغییر مقادیر ژن ها خواهد شد. لذا ضروری است این دو عملگر به صورت توأمان مورد استفاده قرار گیرند و پارامترهای آن به درستی تنظیم شوند.

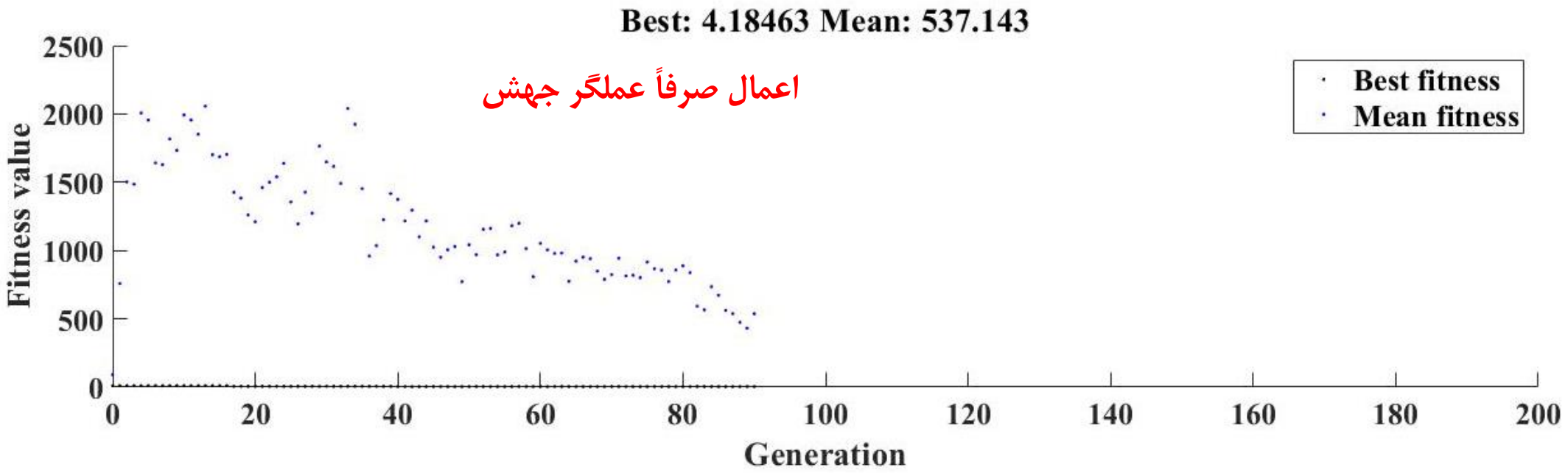
معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

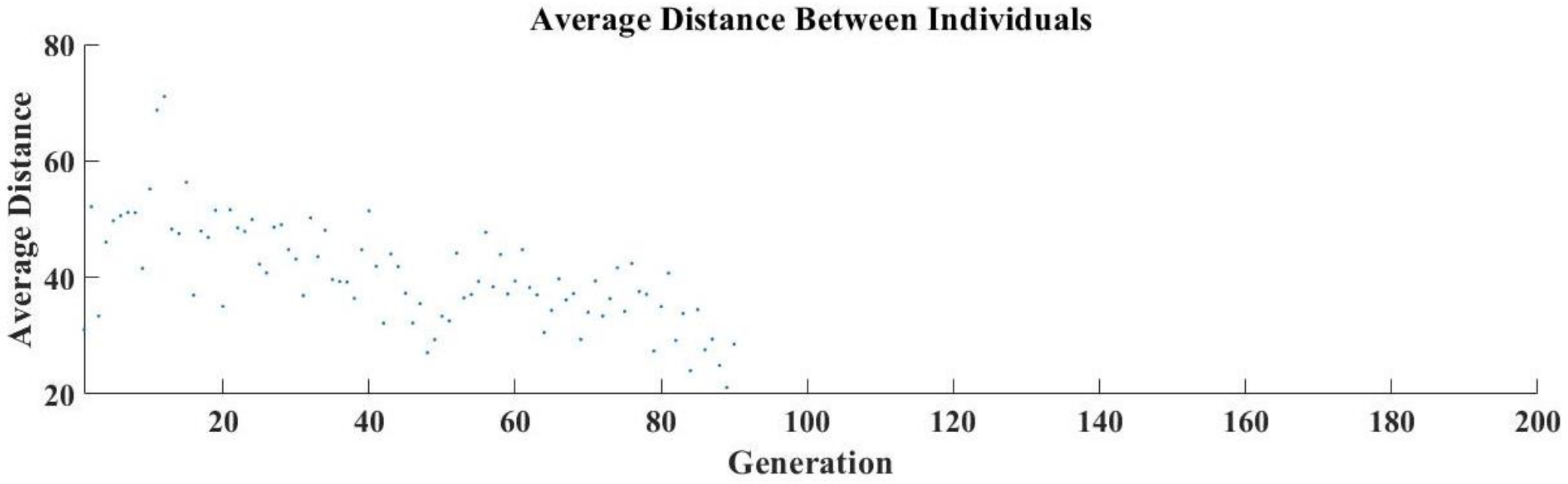




با توجه به عدم انجام عملیات جهش بر روی کروموزم ها، متوسط فاصله بین کروموزم ها با افزایش تکرارها کاهش یافته و پس از مدت کوتاهی به صفر می رسد.

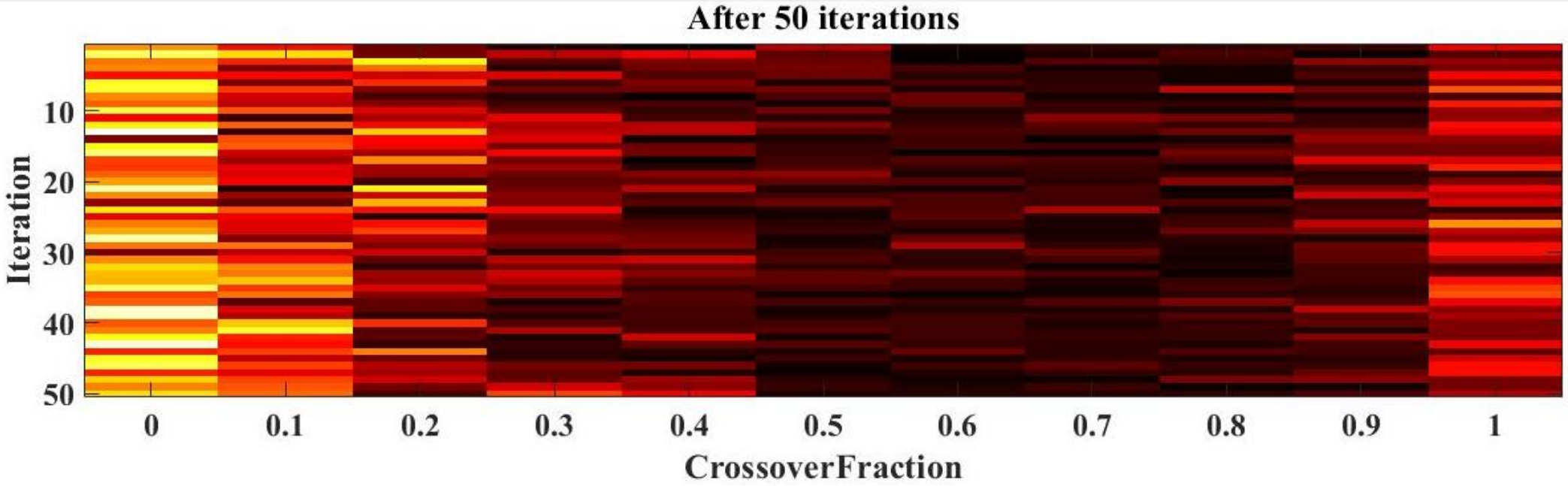
معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

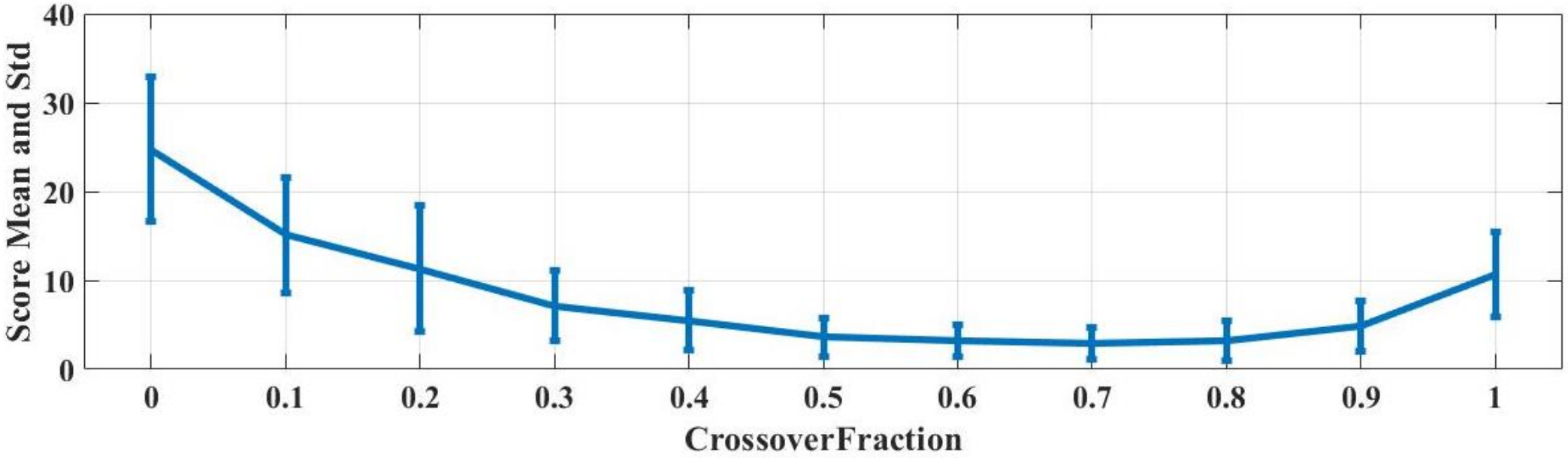




در این حالت علی رغم ایجاد تغییرات در مقادیر ژن ها، تغییری در بهترین مقدار تابع هدف به دلیل عدم ترکیب کروموزم ها، ایجاد نمی شود. با توجه به عدم تغییر مقدار تابع هدف در طی ۵۰ تکرار متوالی، اجرای برنامه متوقف می شود.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن





تحلیل حساسیت انجام شده بر روی بهترین مقدار پارامتر CrossoverFraction نشان می دهد که اعمال مقدار ۰/۷ برای این پارامتر می تواند نتایج بهتری از تابع هدف را ارائه دهد.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

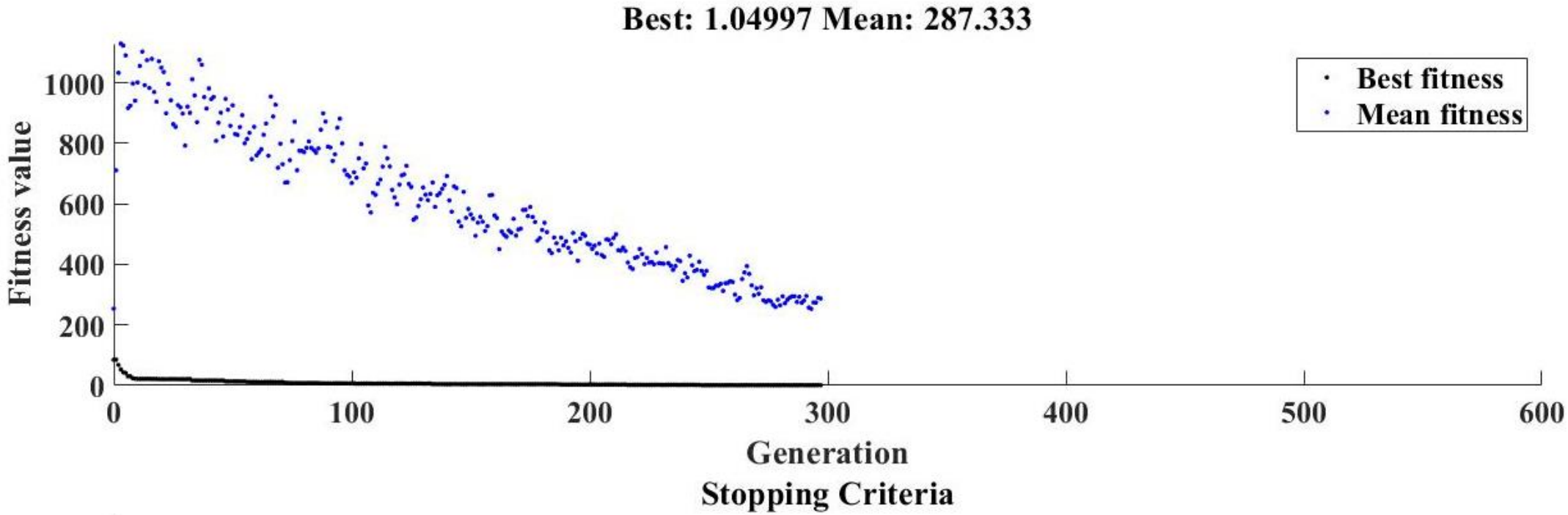
۶- استفاده از هیبرید الگوریتم ژنتیک با سایر الگوریتم های فراکاوشی و یا کلاسیک:

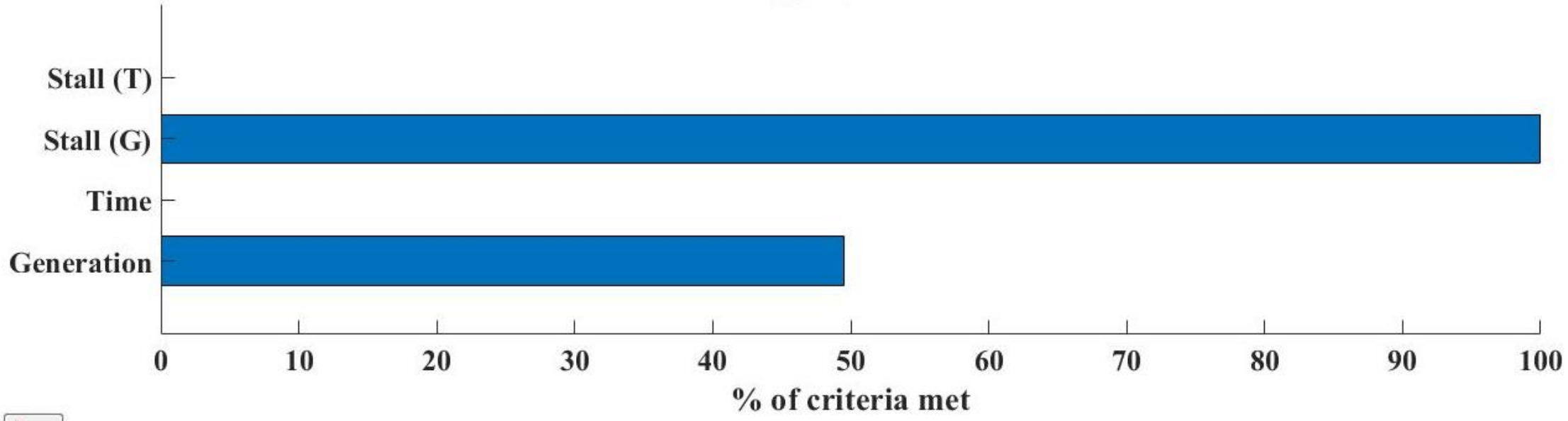
جهت اطمینان از کلی (Global) بودن جواب بهینه بدست آمده از الگوریتم ژنتیک، بهتر است مقدار بهینه جواب بدست آمده از این الگوریتم به عنوان مقدار اولیه، وارد سایر الگوریتم های فراکاوشی و یا کلاسیک شود. در صورتی که همان جواب مجدد حاصل شد، می توان جواب محلی بدست آمده از الگوریتم ژنتیک را به عنوان جواب بهینه کلی در نظر گرفت.

مثال) تابع Rastrigin را با اعمال شش متغییر تصمیم با استفاده از هیبرید الگوریتم ژنتیک و یکی از الگوریتم های کلاسیک حل نمائید.

```
clc
clear
% options = optimoptions('ga','Display','iter','PlotFcn',{@gaplotbestf,@gaplotstopping});
% x=ga(@rastriginsfcn,2,[],[],[],[],[],[],[],options);
%% With Hybrid Method
options = optimoptions('ga','Display','iter','PlotFcn',{@gaplotbestf,@gaplotstopping});
fminuncOptions = optimoptions(@fminunc,'Display','iter','Algorithm','quasi-newton');
options = optimoptions(options,'HybridFcn',{@fminunc, fminuncOptions});
x=ga(@rastriginsfcn,6,[],[],[],[],[],[],[],options);
```

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن





در صورت عدم استفاده از رویکرد هیبریدی، مقدار بهینه بدست آمده توسط الگوریتم ژنتیک برابر با ۱/۰۴۹ است که به دلیل عدم تغییر تابع هدف در طی ۵۰ تکرار متوالی، اجرای برنامه متوقف شده است.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

Generation	Func-count	Best f(x)	Mean f(x)	Stall Generations
286	57400	1.05	294.3	39
287	57600	1.05	275.9	40
288	57800	1.05	292.8	41
289	58000	1.05	273.5	42
290	58200	1.05	280.7	43
291	58400	1.05	296	44
292	58600	1.05	257.6	45
293	58800	1.05	253.4	46
294	59000	1.05	274.3	47
295	59200	1.05	273.9	48
296	59400	1.05	290.1	49
297	59600	1.05	287.3	50

Optimization terminated: average change in the fitness value less tha
Switching to the hybrid optimization algorithm (FMINUNC).

Iteration	Func-count	f(x)	Step-size	First-order optimality
0	7	1.04997		23.4
1	21	0.470893	0.00427919	15.6
2	28	2.13714e-05	1	0.129
3	35	3.58637e-09	1	0.00164
4	42	1.84741e-13	1	9.54e-06

با ادامه اجرای برنامه توسط الگوریتم کلاسیک، مقدار جواب بهینه کلی که معادل با تابع هدف برابر با صفر است، بدست آمد.

نکته: لازم است جهت اطمینان از کلی بودن جواب بهینه، مدل بهینه سازی تدوین شده را چندین بار مورد اجرا قرار داد تا در صورت وجود جواب بهتر، آن جواب به عنوان جواب برتر و بهینه کلی انتخاب شود.

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

مثال) جهت بهینه نمودن میزان آب تخصیصی به نیازهای پایین دست سد کرج از اطلاعات میزان جریان ورودی به مخزن سد، نیاز آبی ماهانه پایین دست، میزان تبخیر از سطح دریاچه سد، میزان بارش نازل شده بر سطح دریاچه سطح و اطلاعات مرتبط با حجم های حداقل و حداکثر سد استفاده می شود. با استفاده از الگوریتم بهینه سازی ژنتیک و شاخص های اطمینان پذیری، برگشت پذیری و آسیب پذیری، نتایج مقادیر بهینه را با سیاست بهره برداری استاندارد (SOP) مورد مقایسه قرار داده و کارآیی الگوریتم فراکاوشی ژنتیک را بررسی نمائید.

$$Minimize\ f = \sum_{t=1}^{T}\left((R_t - D_t)^2 + LOSS_t\right)$$

R_t: میزان خروجی آب از مخزن سد (MCM)

D_t: میزان نیاز آبی پایین‌دست دست (MCM)

$LOSS_t$: مقدار تابع جریمه ناشی از عدم رعایت محدودیت های مسأله

$$S_{t+1} = S_t + I_t - E_t \times Aave_t \times 0.001 + P_t \times Aave_t \times 0.001 - R_t$$

$$A_t = -6 \times 10^{-9} \times S_{t+1}^4 + 2 \times 10^{-6} \times S_{t+1}^3 - 0.0004 \times S_{t+1}^2 + 0.0398 \times S_{t+1} + 0.1904$$

$$Spill_t = \begin{cases} S_{t+1} - S_{max} & if \quad S_{t+1} > S_t \\ 0 & otherwise \end{cases}$$

$$LOSS_t = MAX\left(\left(1 - \frac{S_{t+1}}{S_{min}}\right), 0\right) \times 10^4$$

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

شاخص‌های ارزیابی عملکرد سیستم

جهت بررسی عملکرد مدل تلفیقی توسعه داده شده از معیارهای قابلیت اطمینان‌پذیری (Reliability)، برگشت پذیری (Resiliency) و آسیب پذیری (Vulnerability) استفاده شده است. شاخص پایداری SI (Sustainability Index) با استفاده از این سه معیار، اولین بار توسط Loucks در سال ۱۹۹۷ جهت کمی کردن پایداری سیستم‌های منابع آب و تخمین ظرفیت سیستم در کاهش آسیب‌پذیری ارائه شده‌اند. به عبارت دیگر پایدارتر بودن سیستم منابع آب، نشان‌دهنده‌ی ظرفیت بالای مخزن در کاهش آسیب‌پذیری در آینده است.

شاخص اطمینان پذیری

این معیار اولین بار توسط Hashimoto و همکاران در سال ۱۹۸۲ جهت ارزیابی تأمین نیاز پایین‌دست منابع آب برای اطلاع از دوره هایی که منبع دچار شکست یا کمبود می‌شود، در دو حالت حجمی و زمانی توسعه داده شد. اعتمادپذیری حجمی مقدار حجم آب رها شده در کل دوره زمانی نسبت به مقدار نیاز مخزن است و طبق رابطه‌ی زیر بدست می‌آید:

$$Re_v = \frac{\sum_{t=1}^{T} R_t}{\sum_{t=1}^{T} D_t}$$

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

شاخص اعتمادپذیری زمانی، همان گونه که از نام آن پیدا است، نشان‌دهنده‌ی دوره‌هایی است که سیستم به طور کامل مقدار نیاز پایین دست منابع آب را تأمین کرده است. مطابق رابطه زیر هر چقدر مقدار شاخص اعتمادپذیری بیشتر باشد، قابلیت اطمینان سیستم بیشتر است.

$$Re = \frac{k}{T}$$

k: تعداد دوره‌هایی است که کل نیاز پایین‌دست منابع آب تأمین شده است و کمبود و شکستی رخ نداده است.

$$k = Number(N = 0)$$

N: اختلاف مقادیر رهاسازی و نیاز کل است.

$$N = Demand - Release$$

شاخص آسیب‌پذیری

معیار آسیب‌پذیری بدان علت استفاده می‌گردد که اهمیت میزان کمبودها در هر دوره یکسان نیست. به طور مثال در یک سیستم با مقدار نیاز کل ۲۰ میلیون متر مکعب، ارزش میزان کمبود ۱۰ میلیون متر مکعب با ۵ میلیون متر مکعب یکسان نیست و زمانی که کمبود بیشتر است، اهمیت بیشتری دارد. معیار آسیب‌پذیری طبق روابط مختلفی محاسبه می‌شود. مطابق آخرین روابط ارائه شده، شاخص آسیب‌پذیری مطابق با رابطه زیر به صورت نسبت بین مجموع کل کمبودها به تعداد دوره‌های آنها تقسیم بر کل نیاز دوره‌ی مورد نظر، به دست می آید. مدلی مناسب‌تر است که شاخص آسیب‌پذیری آن کمتر باشد.

$$Vul = \frac{\sum_{t=1}^{T} N_t}{(T-k) \times \sum_{t=1}^{T} D_t}$$

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

شاخص برگشت‌پذیری

این شاخص نشان‌دهنده‌ی این است که در صورت مواجه شدن سیستم با شکست با چه سرعتی می تواند از آن عبور کند. به عبارت دیگر تعداد دوره‌های شکست به صورت متوالی تأثیر قابل توجهی بر سیستم دارد، نسبت به زمانی که سیستم با یک دوره شکست و پس از آن دوره‌ی عدم شکست مواجه باشد. براساس مطالعات Hashimoto و همکاران این شاخص به صورت احتمال برگشت سیستم به حالت مطلوب قبل از رسیدن به شکست است. پس از آن Moy و همکاران شاخص برگشت‌پذیری را به صورت حداکثر دوره‌های متوالی دارای کمبود در یک سیستم قبل از بازگشت به حالت مطلوب تعریف کرده‌اند. براساس روابط ارائه شده در دهه‌های اخیر، پر کاربردترین رابطه که نسبت بین تعداد دوره‌های تبدیل سیستم از حالت شکست به حالت مطلوب به تعداد کل دوره‌هایی که شکست در آن اتفاق افتاده است؛ تعریف می‌گردد و مطابق رابطه زیر محاسبه می شود. مدلی مناسب‌تر است که برگشت پذیری بیشتری داشته باشد.

$$\mathrm{Res} = \frac{Nun_{t=1}^{T}(D_{t+1} = 0 | D_t > 0)}{Nun_{t=1}^{T}(D_t > 0)} \times 100$$

شاخص پایداری (SI)

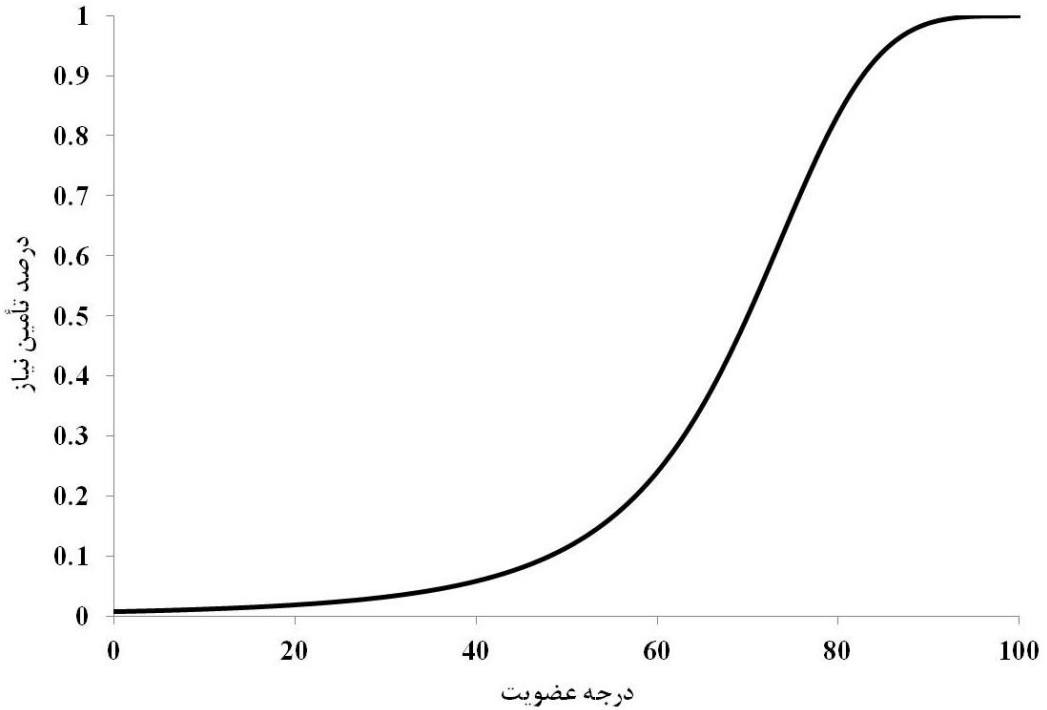
این شاخص اولین بار توسط Loucks در سال ۱۹۹۷ جهت تسهیل فرآیند ارزیابی عملکرد سیستم مخازن ارائه شد. سپس در سال ۲۰۱۱ توسط Sandoval-Solis و همکاران با استفاده از توان هندسی به صورت رابطه‌ی زیر با تلفیق معیارهای ذکر شده در بخش‌های قبل به صورت زیر توسعه داده شد:

$$SI = (Re \times (1 - \mathrm{Vul}) \times \mathrm{Res})^{\frac{1}{3}}$$

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

شاخص های فازی ارزیابی سیستم های منابع آب

جهت توسعه روابط مربوط به معیارهای عملکرد بر اساس تئوری فازی، در ابتدا باید یک تابع مطلوبیت تعریف شود. این تابع بیان کننده این مفهوم است که هرچه مقدار تأمین یا عرضه به نیاز یا تقاضا نزدیک تر شود مطلوبیت و رضایتمندی از عملکرد سیستم بالاتر خواهد رفت. بر این اساس درصدهای تأمین بیشتر دارای مطلوبیت بیشتر خواهند بود، بنابراین پارامترهای این تابع باید به صورت S شکل تعریف شوند. حالات مختلف مطلوبیت سیستم می رضایت تواند در حالت ریسک بالا، حالت رضایت بخش و حالت رضایت مندی زیاد باشد. علاوه بر ویژگی های تابع عضویت زنگوله‌ای، چنانچه این تابع به صورت S تعریف شود، می تواند در مقادیر درصدهای بالا از تأمین نیاز، در حالت رضایتمندی زیاد، برای درصدهای تأمین پایین در حالت ریسکی و برای درصدهای بین این دو، در حالت رضایت بخش عمل کند.



براساس نظر کارشناسان بهره‌برداری پارامترهای a، b و c مرتبط با این تابع مطلوبیت فازی به ترتیب برابر با مقادیر ۳۰، ۲ و ۱۰۰ در نظر گرفته می شود.

$$\mu(x) = \text{bell}(x, a, b, c) = \frac{1}{1 + \left|\frac{x - c}{a}\right|^{2b}}$$

معرفی الگوریتم فراکاوشی ژنتیک (GA) و نحوه تنظیم پارامترهای آن

بر اساس این تابع مطلوبیت، شاخص های ارزیابی اطمینان پذیری، آسیب پذیری و برگشت پذیری فازی به صورت زیر محاسبه می شود:

$$Re_{fuzzy} = \frac{\sum_{t=1}^{T} \mu(x_t)}{T} \times 100$$

$$\mathrm{Res}_{fuzzy} = \frac{\sum_{t=1}^{T} \left(\mu(x_t) - \mu(x_{t-1}) | \mu(x_t) > \mu(x_{t-1})\right)}{\sum_{t=1}^{T} \left(1 - \mu(x_t)\right)} \times 100$$

$$Vul_{fuzzy} = \sum_{t}^{\mathrm{Max}} \left(1 - \mu(x_t)\right) \times 100$$

برنامه تهیه شده مرتبط با مثال بهره برداری از مخزن سد کرج در آدرس \GA\Dam Optimization موجود است.