درس ۱۷:

الگوریتم فراابتکاری

الگوریتم ژنتیک

تهیه شده توسط گروه بهینه‌یاب

www.behinehyab.com

مقدمه

محاسبه راه حل بهینه‌یا *Optimal solution* برای اکثر مسایل بهینه‌سازی که در خیلی از زمینه‌های کاربردی و عملی مشاهده می‌گردند، کار دشوار و سختی است. در عمل، معمولا به راه حل‌های خوب که از الگوریتم‌های هیوریستیک *Heuristic* یا متاهیوریستیک (همان فراابتکاری)*Metaheuristic* بدست می‌آید، اکتفا می‌گردد. متاهیوریستیک‌ها مجموعه ای از فنون بهینه‌سازی تقریبی *Approximate optimization techniques* را که عمدتا در طول دو دهه گذشته شهرت پیدا کرده اند، در بر می‌گیرند. روش‌های فراابتکاری راه حل‌های قابل قبول در زمان معقول را برای مسایل پیچیده و سخت، در زمینه‌های مهندسی و علوم پایه ارایه می نمایند. برخلاف الگوریتم‌های بهینه‌سازی دقیق *Exact optimization algoritms*، فراابتکاری‌ها بهینه بودن جواب‌های بدست آمده را ضمانت نمی نمایند.

کلمه متاهیوریستیک از کلمه یونانی *"Heuriskein"* به معنای هنر کشف قواعد جدید برای حل مسایل گرفته شده است. پیشوند "متا" نیز از یک کلمه یونانی به معنای "متدولوژی" سطح بالا گرفته شده است. واژه " متاهیوریستیک " اولین بار توسط گلوور در سال ۱۹۸۶ ارایه گردید. روش جستجوی متاهیوریستیک را می توان به صورت متدولوژی‌های عمومی سطح بالایی که می توانند به عنوان یک استراتژی راهنما در طراحی هیورستیک‌های اختصاصی برای حل مسایل بهینه‌سازی تخصصی به کار روند، تعریف کرد.

برخلاف روش‌های دقیق، متاهیوریستیک‌ها (فراابتکاری‌ها) برای مسایل با اندازه‌های بزرگ، کاربرد دشوار دارد و راه حل‌هایی راضی کننده ای در زمان معقولی ارایه می نمایند. در این الگوریتم‌ها، هیچ گونه ضمانتی برای یافتن جواب بهینه‌سراسری یا حدودی از آن وجود ندارد. متاهیوریستیک‌ها (فراابتکاری‌ها) در طول بیست سال گذشته شهرت زیادی پیدا کرده اند. کاربرد و استفاده از آن‌ها در خیلی از مسایل، کارایی و اثر بخشی آن‌ها را برای حل مسایل پیچیده و بزرگ نشان می دهد. فراابتکاری‌ها در خیلی از زمینه‌ها از قبیل موارد زیر کاربرد دارند:

- ✓ طراحی مهندسی، بهینه‌سازی توپولوژی، بهینه‌سازی مسایل الکترونیک، آئرودینامیک، دینامیک سیالات، مخابرات، رباتیک

- ✓ یادگیری ماشین و کاوش داده‌ها
- ✓ مدل سازی سیستم‌ها، شبیه سازی و تحقیق در شیمی، فیزیک، بیولوژی، کنترل، سیگنال، و پردازش تصویر
- ✓ مسایل مسیردهی و برنامه‌ریزی، برنامه‌ریزی ربات، مسایل تولید و زمان بندی، حمل و نقل و لجستیک، مدیریت زنجیره تامین

روش‌های مختلف الگوریتم‌های فراابتکاری یا متاهیوریستیک تا به حال پیشنهاد شده است که به صورت زیر هستند:

- ✓ بهینه‌سازی کلونی مورچگان *Ant colony optimization*
- ✓ بهینه‌سازی کلونی زنبوران *Bee colony*
- ✓ الگوریتم‌های ترتیبی *cultural algorithms*
- ✓ الگوریتم‌های با هم تکاملی *Co-evolutionary algorithms*
- ✓ الگوریتم ژنتیک *Genetic algorithm*
- ✓ جستجوی محلی تکراری *Iterated local search*
- ✓ بهینه‌سازی توده ذرات *particle swarm intelligent*
- ✓ انجماد تدریجی *Simulated Annealing*
- ✓ جستجوی ممنوع *Taboo search*
- ✓ جستجوی همسایگی متغیر *Variable neighboar search*

در طراحی یک فراابتکاری، دو معیار متناقض شامل کاوش *Exploration* در فضای جستجو (گوناگونی و تنوع) و تبعیت (*Exploition*) از بهترین راه حل‌های پیدا شده، باید در نظر گرفته شوند. در کاوش در ناحیه‌های جستجو نشده بررسی صورت می‌گیرد. در تبعیت، در ناحیه‌های امید بخش که تا به حال در آن ناحیه یک جواب خوب پیدا شده است بررسی بیشتر صورت می‌گیرد. در صورتیکه به رفتار **کاوش** اهمیت بیشتری داده شود، الگوریتم رفتار تصادفی بیشتری خواهد داشت و به سمت رفتار تصادفی میل می‌کند و در

صورتی که به رفتار **تبعیت** توجه بیشتری شود، الگوریتم از رفتار تصادفی فاصله می‌گیرد و جستجو تنها در محدوده راه حل‌های خوب به جستجو می پردازد که می تواند به راه حل‌های محلی منجر شود.

معیارهای طبقه بندی زیادی ممکن است برای طبقه بندی فراابتکاری‌ها استفاده شود که در زیر به بعضی از آن‌ها اشاره می‌شود:

الهام گرفته از طبیعت در مقابل عدم الهام از طبیعت

خیلی از الگوریتم‌های فراابتکاری از فرایندهای طبیعی الهام گرفته شده اند: از قبیل الگوریتم‌های اجتماع مورچگان و زنبور عسل که از هوش توده ای از گونه‌های مختلف مورچگان و زنبوران استفاده می کنند. برخی از الگوریتم این چنین نبوده و از یک رفتار منطقی پیروی می کنند که می توان به الگوریتم جستجوی ممنوع نام برد.

نحوه استفاده از حافظه

بعضی از الگوریتم‌های فراابتکاری از قبیل انجماد تدریجی یا گرم و سرد کردن شبیه سازی شده بدون حافظه هستند و حرکت‌های قبلی را در جایی ذخیره نمی کنند. در مقابل الگوریتم جستجوی ممنوع از یک حافظه که بعضی از اطلاعات را در طول جستجو بدست می‌آورد، استفاده می‌کند.

قطعی در مقابل احتمالی

یک الگوریتم قطعی، یک مسئله بهینه‌سازی را از طریق تصمیم گیری قطعی حل می نماید(برای مثال الگوریتم جستجوی محلی و جستجوی ممنوع). در الگوریتم‌های فراابتکاری احتمالی، بعضی از قواعد احتمالی در فرایند جستجو مورد استفاده قرار می‌گیرد که می توان به الگوریتم انجماد تدریجی و الگوریتم ژنتیک اشاره کرد. در الگوریتم‌های قطعی، با داشتن یک راه حل اولیه و اجراهای متفاوت، تنها یک جواب نهایی بدست می‌آید و در حالی که در الگوریتم‌های تصادفی، با داشتن یک راه حل اولیه و اجراهای متفاوت، ممکن است که جواب‌های نهایی متفاوتی بدست آید.

تکراری در مقابل حریصانه

در الگوریتم‌های تکراری، الگوریتم با یک راه حل کامل شروع شده و در هر تکرار راه حل یا راه حل‌ها تغییر پیدا می کنند. در الگوریتم‌های حریصانه یک راه حل کامل در اختیار نبوده بلکه با یک راه حل ساخته نشده شروع شده و در هر مرحله، یک متغیر تصمیم از مسئله یک قسمت از راه حل را می سازد. اغلب الگوریتم‌های فراابتکاری، الگوریتم‌های تکراری هستند.

در ادامه **الگوریتم ژنتیک** که الگوریتمی است که از طبیعت الهام گرفته است و در رده الگوریتم‌های احتمالی، و تکراری با حافظه قرار می‌گیرد صحبت می‌شود.

الگوریتم ژنتیک

براساس تئوری انتخاب طبیعی، گیاهان و موجودات زنده ای که در حال حاضر وجود دارند، نتیجه میلیون‌ها سال تطابق با تقاضای‌های محیط می باشند. در هر زمانی، تعدادی از جانداران مختلف، ممکن است که با هم دیگر در یک اکوسیستم زندگی نمایند و برای دستیابی به منابع مشترک با هم دیگر به رقابت بپردازند. موجوداتی که توانایی بیشتری در دستیابی به منابع و تولید مثل داشته اند، در طبیعت نیز بیشتر مشاهده می شوند. موجوداتی که توانایی کمتری داشته اند، به دلایل مختلف، به مرور تعداد آن‌ها کم و کمتر شده و حتی به نابودی کشیده شده اند. در این حالت، گفته می شود که گونه‌های اول نسبت به گونه‌های بعدی شایسته تر می باشند و مشخصه هایی که موجب شده است تا بعضی گونه‌های شایسته تر باشند مشخصه‌های ارجح می نامند. در طول زمان، گفته می شود که کل جمعیت اکوسیستم دچار تکاملی شده است به طور متوسط، شامل موجوداتی شده است که شایسته تر از موجودات قبلی آن می باشند، زیرا این موجودات مشخصاتی داشته اند که منجر به بقای بیشتر آن‌ها شده است.

تکنیک **محاسبات تکاملی** یا ***Evolutionary computation*** یا *EC* قواعد تکامل را در الگوریتم هایی برای جستجوی راه حل‌های بهینه مسایل به کار می گیرد. در واقع، تکنیک محاسبات تکاملی، توسعه علوم تکامل بیولوژیکی به حوزه علوم محاسباتی و کامپیوتر بوده است. در یک الگوریتم جستجو، تعدادی از راه حل‌های ممکن از یک مسئله در اختیار بوده و هدف یافتن بهترین راه حل ممکن در زمان محدود می باشد.

تفاوت اصلی الگوریتم‌های تکاملی نسبت به الگوریتم‌های دیگر این است که این الگوریتم‌ها مبتنی بر جمعیت عمل می نمایند. در این الگوریتم ها، معمولا یک جمعیت اولیه ایجاد و سپس تکامل داده می شود.

الگوریتم ژنتیک یا ***Genetic Algorithm*** مشهورترین تکنیک در تحقیقات الگوریتم‌های تکاملی است. این الگوریتم یک تکنیک جستجو را برای یافتن راه حل‌های نزدیک به بهینه در زمان قابل قبول برای مسایل بهینه‌سازی ارایه می نماید.

الگوریتم ژنتیک با یک جمعیت اولیه از راه حل‌ها شروع می شود. هر راه حل از طریق یک کروموزوم نمایش داده می شود و تمامی راه حل‌های ممکن باید با استفاده از یک سیستم کدگذاری، تبدیل به کد شوند. سپس مجموعه ای از اپراتورهای تولید مثل، باید تعیین گردد. اپراتورهای تولید مثل، مستقیما روی کروموزوم‌ها عمل نموده و سپس کروموزوم‌ها تحت اپراتور جهش و ترکیب قرار می گیرند. طراحی ساختار کدگذاری و اپراتورها باید با حوصله و دقت فراوان انجام گردد. این طراحی، تاثیر بسیار زیادی روی عملکرد الگوریتم ژنتیک خواهد داشت.

رویه انتخاب برای رقابت افراد در داخل جمعیت به کار می رود که براساس یک تابع شایستگی عمل می‌نماید. برای هر کروموزوم، یک مقدار مرتبط با شایستگی راه حل که نمایش می دهد، وجود دارد. الگوریتم ژنتیک به دنبال حداکثر کردن مقدار تابع شایستگی است. البته در صورتی که تابع هدف به صورت حداقل سازی یک تابع هدف باشد، به هنگام کردن الگوریتم برای حداقل سازی کاری ساده ای است. هر تابع هزینه به راحتی قابل تبدیل به یک تابع شایستگی است مثلا با معکوس کردن تابع هزینه، می توانی یک تابع شایستگی متناسب آن ساخت.

بعد از این که مراحل تولید مثل و تابع برازندگی تعریف شد، یک الگوریتم ژنتیک براساس یک ساختار مشابه و پایه طراحی می‌گردد. این ساختار ساده با تولید یک جمعیت اولیه از کروموزوم‌ها شروع می‌گردد. معمولا، جمعیت اولیه به صورت تصادفی تولید می‌گردد.

مراحل الگوریتم ژنتیک

الگوریتم ژنتیک یک رویه تکراری را به منظور تکامل جمعیت انجام می دهد که در آن فعالیت‌های زیر انجام می شود.

انتخاب

اولین قدم شامل انتخاب افرادی از جمعیت برای تولید مثل است. این انتخاب به صورت تصادفی، با استفاده از یک احتمال متناسب با تابع برازندگی افراد انجام می‌گردد. بنابراین، همواره بهترین فرد شانس بیشتری برای انتخاب نسبت به فرد ضعیف تر دارد.

تولید مثل

در قدم دوم، عملگر تقاطع (ترکیب) و جهش روی افراد انتخاب شده به کار گرفته شده و کروموزوم‌های جدید تولید می‌گردند. در این مرحله، فرزندان جدید تولید می شوند.

ارزیابی

در این مرحله، میزان برازندگی فرزندان جدید تولید شده تعیین می شود.

جابه جایی

در این مرحله، افرادی از جمعیت قبلی حذف و با افراد جدیدی که به تازگی تولید شده جایگزین می شوند.

پس از بیان اصول الگوریتم ژنتیک می توان مراحل این الگوریتم را به صورت زیر بیان کرد:

شروع

در این مرحله، یک جمعیت اولیه شامل n کروموزوم تولید می‌گردد.

برازندگی

مقدار برازندگی $f(x)$ هر کروموزوم x از جمعیت ارزیابی می‌گردد.

جعیت جدید

ایجاد یک جمعیت جدید با تکرار مراحل زیر انجام گردد.

انتخاب

انتخاب دو کروموزوم از جمعیت مطابق با مقدار برازندگی آن‌ها. در این حالت افرادی که برازندگی بیشتری دارند با احتمال بیشتری انتخاب می شوند.

عملگر تقاطعی

با استفاده از عملگر تقاطعی روی والدین انتخاب شد، فرزندان جدید تولید می‌گردند. اگر عملگر تقاطع انجام نگردد، فرزندان همان والدین خود خواهند بود.

جهش

با استفاده از عملگر جهش، فرزندان جدید تحت عمل جهش قرار می گیرند. در این حالت، جهش روی هر ژن کروموزوم صورت می گیرد.

پذیرش

فرزند جدید در جمعیت قرار داده می شود.

جابه جایی

جمعیت جدید (فرزندان) با جمعیت قبلی (والدین) با همدیگر تشکیل یک نسل جدید را می دهند. در این حالت، بعضی از والدین حذف شده و بعضی از فرزندان جانشین آن‌ها می شوند و دو جمعیت تبدیل به یک جمعیت شده تا اندازه جمعیت ثابت بماند.

شرط توقف

اگر شرط توقف برآورده شده است توقف کنید و بهترین راه حل در جمعیت جاری را گزارش دهید.

تکرار

در صورت عدم برآورده شدن شرط توقف، به **قدم جمعیت جدید** است برگردید.

اجزای الگوریتم ژنتیک

در ساختار الگوریتم ژنتیک بحث شده در قسمت قبلی، اجزایی به کار گرفته شده است که در ادامه آن‌ها مورد بحث قرار می گیرد.

کروموزوم

به رشته یا دنباله ای از بیت‌ها که به عنوان شکل کد شده یک جواب ممکن از مساله مورد نظر می باشد، کروموزوم می گوییم. در حقیقت، بیت‌های یک کروموزوم، نقش ژن‌ها را در طبیعت بازی می‌کند. هر ژن، متغیری گسسته است که از یک مجموعه انتخاب می شود. چنانچه از کد گذاری باینری استفاده شود، هر ژن یکی از دو مقدار ۰ یا ۱ را می پذیرد. یک کروموزوم که دارای ۱۰ ژن در شکل زیر نمایش داده شده است.

1	0	1	0	1	1	1	0	0	1

جمعیت Population

مجموعه ای از کروموزوم‌ها را جمعیت گویند. یکی از ویژگی‌های الگوریتم ژنتیک این است که به جای تمرکز بر روی یک نقطه از فضای جستجو یا یک کروموزوم، بر روی جمعیتی از کروموزوم‌ها تمرکز می‌کند. بدین ترتیب در هر مرحله، الگوریتم دارای جمعیتی از کروموزوم‌ها بوده که خواص مورد نظر را بیشتر از جمعیت مرحله قبل دارا می باشند. هر جمعیت یا یک نسل از کروموزوم ها، دارای یک اندازه می باشد که به اندازه جمعیت *population size* معروف است. اندازه جمعیت معرف تعداد کروموزوم‌های موجود در جمعیت یا یک نسل است. اگر تعداد کروموزوم‌ها خیلی کم باشد، امکان شکل گیری عملیات جابه جایی توسط الگوریتم ژنتیک بسیار کم خواهد بود و تنها قسمت کمی از فضای جستجو مورد کاوش قرار خواهد گرفت. از طرف دیگر، اگر تعداد کروموزم‌ها خیلی زیاد باشد، الگوریتم بسیار کند خواهد شد. براساس تحقیقات، جمعیت‌های با اندازه ۲۰ تا ۳۰ کروموزوم، مناسب تر است. در شکل زیر یک جمعیت با ۴ کروموزوم نمایش داده شده است.

جمعیت	کروموزوم ۱	۱	۰	۱	۰	۱	۱	۱	۰	۰	۱
	کروموزوم ۲	۰	۱	۱	۱	۰	۰	۰	۱	۱	۱
	کروموزوم ۳	۱	۰	۱	۱	۱	۰	۰	۱	۰	۱
	کروموزوم ۴	۱	۱	۰	۱	۱	۰	۱	۱	۱	۱

مقدار برازندگی

مناسب بودن یا نبود جواب با استفاده از مقدار تابع هدف سنجیده می شود. جواب هر چه مناسب تر باشد، به همان اندازه مقدار برازندگی بزرگتری دارد. برای آنکه شانس بقای چنین جوابی بیشتر شود، احتمال بقای آن جواب متناسب با مقدار برازندگی آن در نظر گرفته می شود. بنابراین، کروموزومی که برازنده تر است به احتمال بیشتری در تولید فرزندان شرکت می‌کند. به عنوان مثال، چنانچه هدف مسئله بیشینه کردن یک تابع باشد، مقدار برازندگی، یک تابع صعودی از تابع هدف در نظر گرفته می شود و اگر هدف یافتن مقدار کمینه یک تابع باشد، عدد برازندگی، یک تابع نزولی از آن در نظر گرفته می شود. معمولا در مواردی که امکان دارد توصیه می شود تابع برازندگی در فاصله صفر و یک نرمال شود.

کد گذاری

کدگذاری، فرایندی است که به واسطه آن، راه حل‌ها در فضای فیزیکی مساله تبدیل به راه حل هایی با کدهایی می‌گردد که قابل درک برای کامپیوتر است. کدگذاری تاثیر زیادی روی کیفیت الگوریتم خواهد داشت و طراحی مناسب یک ساختار کدگذاری از اهمیت بالایی در الگوریتم ژنتیک برخوردار است. نمایش اعداد در مبنای ۲ نوعی از کد گذاری‌های رایج است.

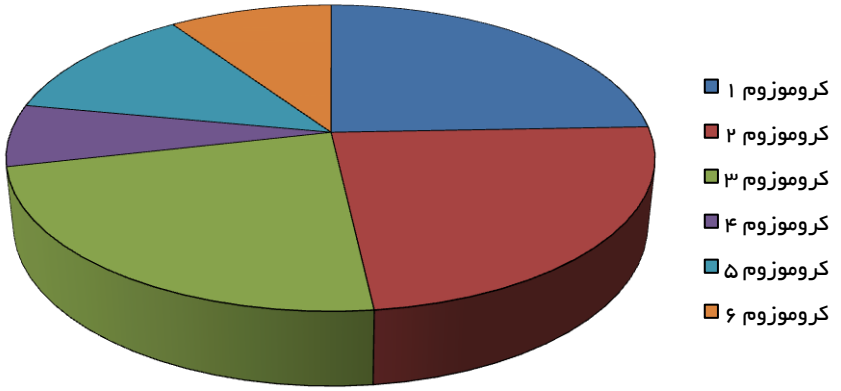
انتخاب

انتخاب، فرایند انتخاب دو والد از جمعیت برای عمل **تقاطع** یا *Crossover* است. در این مرحله، در خصوص نحوه انتخاب والدین برای عمل تقاطع، نحوه تولید فرزندان و تعدد فرزندان تصمیم گیری گردد. هدف انتخاب این است که والدین شایسته تر انتخاب شده که منجر به تولید فرزندانی با برازندگی بالا گردد. کروموزوم هایی که از جمعیت اولیه برای تولید مثل انتخاب می‌گردند، والدین نام دارند. انتخاب، نحوه تعیین این والدین را مشخص می نماید. براساس نظریه تکامل داروین، بهترین‌ها باید شانس بیشتری برای تولید مثل و بقا داشته باشند.

به طور کلی، دو رویه انتخاب وجود دارد که عبارتند از رویه مبتنی بر تناسب و رویه مبتنی بر ترتیب. رویه مبتنی بر تناسب، والدین را براساس مقدار برازندگی آن‌ها نسبت به مقدار برازندگی سایر والدین جمعیت انتخاب می‌کند. برنامه انتخاب مبتنی بر ترتیب، والدین را نه بر اساس مقدار خام برازندگی آنها، بلکه براساس رتبه آن‌ها در جمعیت انتخاب می‌کند. چند روش برای رویه انتخاب در ادامه بیان می شود.

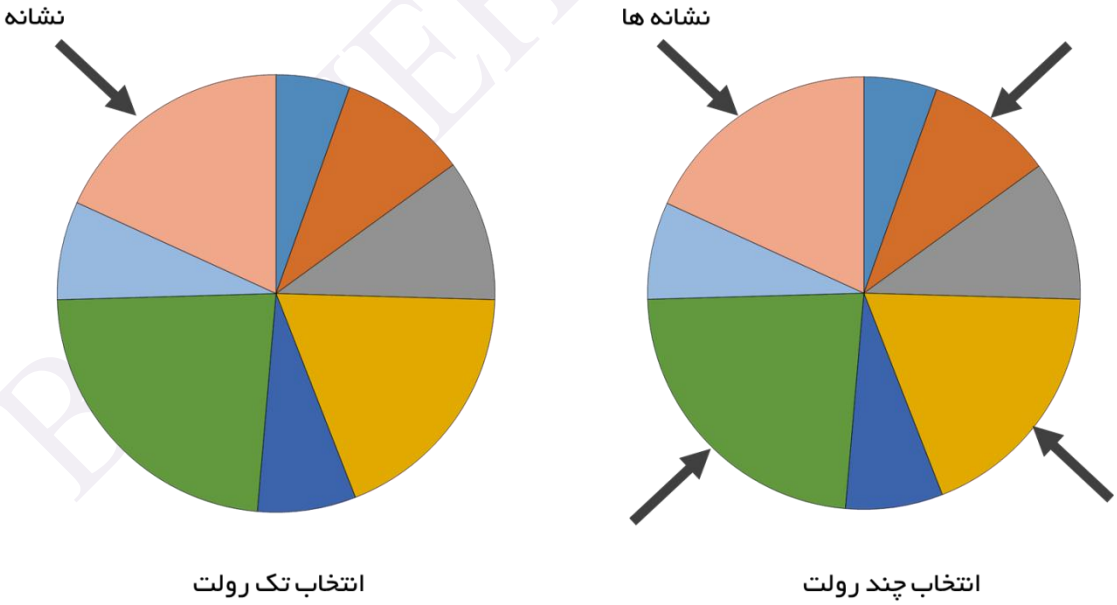
روش چرخ رولت Roulette wheel

روش چرخ رولت، یکی از روش‌های سنتی انتخاب در الگوریتم ژنتیک است. در این حالت یک کرموزموم از جمعیت براساس احتمالی متناسب با مقدار شایستگی اش انتخاب می‌گردد. در این روش، سطح چرخ به بخش هایی تقسیم می شود که تعداد آن‌ها برابر با تعداد اعضای جمعیت و سطح هر بخش متناسب با مقدار برازندگی هر کروموزوم است. سپس چرخ به گردش در می آید تا در نقطه ای به تصادف متوقف گردد. این نقطه ، کروموزوم انتخاب شده را مشخص می سازد. شکل زیر نمایی از چرخ رولت را نشان می دهد.



در این شکل، کروموزوم‌های ۱، ۲، و ۳ دارای برازندگی بیشتری نسبت به کروموزوم ۴، ۵ و ۶ هستند. این شیوه با گذشت زمان، تعداد کروموزوم‌های مطلوب در جمعیت افزایش می یابد.

دو نوع چرخ رولت وجود دارد. در نوع اول با هر بار چرخش، یک کروموزوم انتخاب می‌گردد. در نوع دوم، با هر بار چرخیدن، بیش از دو کروموزوم انتخاب می شود. در شکل زیر دو نوع چرخ رولت را نشان می دهد.



روش تصادفی

در این روش، والدین براساس یک روش کاملا تصادفی از جمعیت انتخاب می‌گردند. این روش نسبت به سایر روش‌ها کمتر مورد استفاده قرار گرفته است. با توجه به عدم اهمیت به شانس بیشتر بهترین‌ها برای تولید مثل، این روش از کیفیت پایینی نیز برخوردار است.

روش رتبه بندی

در روش چرخ رولت، مشکلی که ممکن است به وجود آید این است که اگر برازندگی یک کروموزوم خیلی متفاوت از بقیه باشد، ممکن است شانس بسیار زیاد یا شانس بسیار کمی برای انتخاب آن باشد. در صورتی که یک کروموزوم، مقدارش خیلی بیشتر از بقیه باشد، شانس انتخاب او خیلی زیاد شده و در نتیجه شانس انتخاب بقیه خیلی کم خواهد شد. روش دیگری که مشکل فوق را حل کرده است، روش رتبه بندی است. در این روش انتخاب، کرموزوم ها براساس مقدار برازندگی آن‌ها رتبه بندی شده و رتبه آن‌های جاری برازندگی آن‌ها در چرخ رولت مورد استفاده قرار می گیرد. برای این منظور، ابتدا کرموزوم‌های را به ترتیب از بدترین به بهترین مرتب می‌گردند، سپس کرموزوم ها رتبه بندی می شوند. در این حالت، بدترین کروموزم رتبه ۱،کروموزوم ماقبل بدترین رتبه ۲، و الی اخر تا اینکه به بهترین کروموزم رتبه n اختصاص داده می شود(n تعداد کرموزوم‌های موجود در جمعیت است). احتمال انتخاب کروموزم i –ام به صورت زیر محاسبه می شود.

$$p_i = \frac{Rank_i}{\sum_{j=1}^{n} Rank_j}$$

عملگر تقاطعی crossower

عملگر تقاطعی، فرایندی است که دو والد را در نظر گرفته و براساس آن‌ها یک فرزند جدید تولید می نماید. عملگر تقاطعی در جمعیت به این امید فعالیت می‌کند که فرزند بهتری تولید گردد. این عملگر یک اپراتور ترکیب بندی است که در سه مرحله صورت می گیرد.

۱- اپراتور تولید مثل یک زوج از والدین را از جمعیت انتخاب می‌کند.

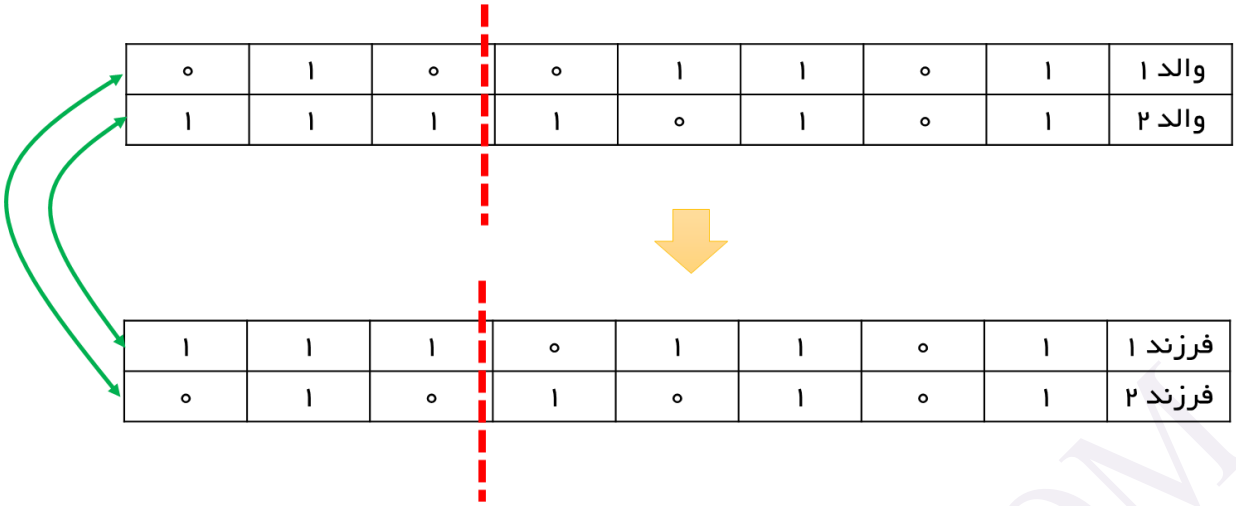
۲- یک نقطه تقاطع به صورت تصادفی در طول رشته انتخاب می شود.

۳- در نهایت، مقادیر رشته‌ها با توجه به نقطه تقاطع تعویض می‌گردند.

عملگر تقاطعی با یک احتمال از قبل تعیین شده، بر روی کروموزم‌های والد عمل می‌کند. اگر هیچ تقاطعی صورت نگیرد، فرزندان دقیقا مشابه والدین خواهند بود. در صورتی که عمل تقاطع صورت گیرد، فرزندان از قسمت‌های مختلف کرموزوم‌های والد ساخته می‌شوند. اگر احتمال تقاطع ۱ باشد، تمامی فرزندان از طریق عمل تقاطعی ایجاد می شوند. عملیات تقاطع با این هدف انجام می شود که کرموزوم‌های جدید دربردارنده ویژگی های خوب کروموزوم‌های قبلی خواهد بود و شاید کروموزوم‌های جدید عملکرد بهتری داشته باشند. اما بهتر است همیشه بهترین کرمومزوم‌های نسل قبلی بدون هیچ تغییر به نسل جدید منتقل شوند. روش‌های مختلفی برای عملگر تقاطع وجود دارد که در ادامه برخی از روش‌ها متداول شرح داده می شود.

تقاطع تک نقطه ای

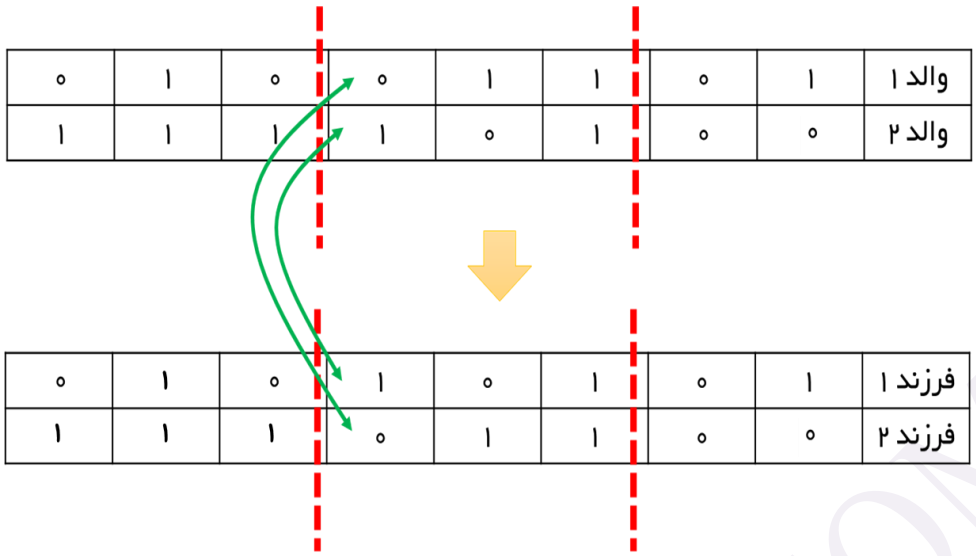
عملگر تقاطعی تک نقطه ای، دو کروموزوم را به طور تصادفی از یک نقطه شکسته و بخش های شکسته شده دو کروموزوم را جابه جا می‌کند. بدین ترتیب، دو کروموزوم جدید به دست می آید. به کروموزوم های اولیه، کروموزم‌های والد و به کروموزوم‌های حاصل شده از عمل جابه جایی، کروموزوم فرزند می گویند. شکل زیر یک مثال از عملگر تقاطع تک نقطه ای را نشان می دهد.



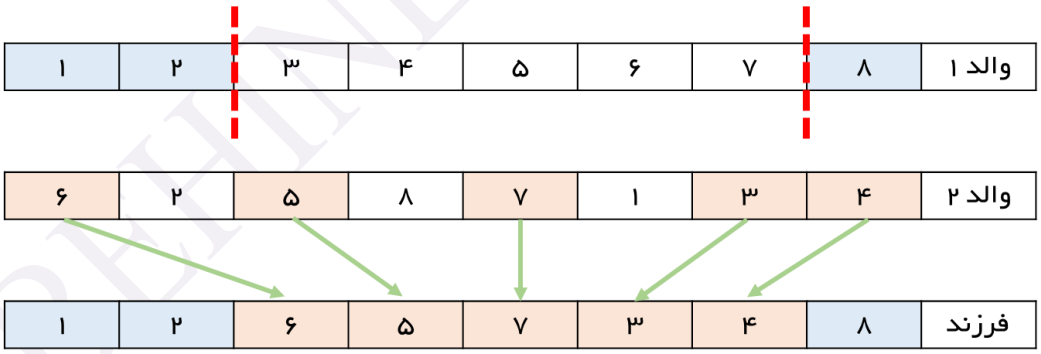
تقاطع دو نقطه ای

در عملگر تقاطع دو نقطه ای، دو نقطه شکست تصادفی در طول رشته جواب در نظر گرفته می شـود و سپس به صورت یک در میان قسمت‌های والدها به فرزندان منتقل می شود. باید توجه داشت در صورتی که تعداد نقاط شکست افزایش یابد عملکرد الگوریتم ژنتیک کاهش می یابد. از طرفی، در صورت افزایش تعـداد نقاط شکست، فضای مساله به صورت کاملتری جستجو می‌گردد.

در الگوریتم ژنتیک، معمولا از عملگر تقاطع تک نقطه ای استفاده می‌گردد. اما در تقاطع تک نقطـه ای این مشکل وجود دارد که هیچ گاه ابتدا و انتهای رشته جواب یک والد نمی توانـد بـه یـک فرزنـد بـه طـور همزمان منتقل گردد، زیرا همواره ابتدای رشته یا انتهای رشته به یک فرزند منتقل می شـود. اگـر ژن‌هـای ابتدایی و انتهایی الگوریتم به طور همزمان خصوصیات خوبی را به وجود آورند، در صورت استفاده از تقـاطع تک نقطه ای، این خصوص به هیچکدام از فرزندان منتقل نمی‌گردد. استفاده از تقاطع دو نقطـه ای، از ایـن مشکل جلوگیری نموده و به طور کلی عملکرد بهتری نسبت به تک نقطه ای دارد.



عملگر تقاطع دو نقطه ای برای یک کروموزوم ترتیبی به سادگی شکل فوق نیست. در این حالت، ابتدا قسمت های کناری از والد اول به فرزند منتقل شده و سپس بقیه ژن‌ها به ترتیب مشخص شده در والد دوم به فرزند منتقل می‌گردد. در صورتی که قسمت دوم کروموزوم فرزند عینا از والد دوم انتخاب گردد، می تواند یک کروموزوم غیرقانونی تولید می‌گردد. به همین دلیل، از والد دوم، مقادیر باقی مانده فرزند، به ترتیب ظهور در والد دوم، به فرزند منتقل می‌گردد. در شکل زیر نحوه عملیات عملگر تقاطع دو نقطه ای بیان شده است.



تقاطع یکنواخت

براساس این عملگر، یک ژن از هردو والد به طور مستقل از سایر ژن ها، شانس برابر برای حضور در کروموزوم یک فرزند را دارند. در این حالت، براساس یک توزیع تصادفی باینری مشخص می‌گردد که یک ژن از کدام والد انتخاب گردد. مثلا اگر مقدار توزیع باینری ۱ باشد ژن از والد اول و در صورتی که صفر

باشد، از والد دوم انتخاب می‌گردد. برای فرزند دوم، عکس فرزند اول در نظر گرفته شده است. تابع توزیع به صورت زیر می شود.

$$p_i = \begin{cases} 1 & \text{ژن از والد اول گرفته شود} \\ 0 & \text{ژن از والد دوم گرفته شود} \end{cases}$$

براساس تابع توزیع فوق، عملگر تقاطع یکنواخت برای یک مثال به صورت زیر می شود.

والد ۱	۱	۰	۱	۱	۰	۰	۱	۰
والد ۲	۱	۰	۱	۰	۱	۱	۱	۱
مقدار تصادفی	۱	۱	۰	۱	۰	۱	۱	۰
فرزند ۱	۱	۰	۱	۱	۰	۱	۱	۱
فرزند ۲	۱	۰	۱	۰	۱	۱	۱	۰

تقاطع از سه والد یا Three Parent Crossover

از آن جایی که تقاطع در الگوریتم‌های تکاملی محدود به خصوص بیولوژیکی نیست، می توان بیش از دو والد را برای تولید یک فرزند در این الگوریتم‌ها استفاده کرد. در تکنیک تقاطع از سه والد، سه والد به طور تصادفی انتخاب می‌گردند. هر ژن از والد اول با همان ژن از والد دوم مقایسه می‌گردد. اگر مشابه باشند، همان ژن به فرزند منتقل می‌گردد. در صورتی که مشابه نباشد، ژن والد سوم در نظر گرفته می شود.

شماره ژن	۱	۲	۳	۴	۵	۶	۷	۸
والد ۱	۱	۱	۰	۱	۰	۰	۰	۱
والد ۲	۰	۱	۱	۰	۱	۰	۰	۱
والد ۳	۰	۱	۱	۰	۱	۱	۰	۰
فرزند	۰	۱	۱	۰	۱	۰	۰	۱

تقاطع PPC یا Precedence Preservation Crossover

تقاطع *PPC* برای مساله مسیریابی وسایل حمل و نقل و برای مسایل زمان بندی توسعه داده شد. این عملگر به نحوی طراحی شده است که برای مسایل مزبور مناسب باشد. در زیر عملگر *PPC* برای یک مسئله توالی شرح داده شده است:

۱- در ابتدا یک بردار به صورت کروموزوم در نظر بگیرید. بردار را با استفاده از اعداد ۱ و ۲ به طور تصادفی پر کنید.

۲- یک فرزند یا کروموزوم خالی را در نظر بگیرید. ژن‌های این کرموزوم به صورت زیر پر می شوند.

a. بردار تصادفی را بخوانید. در صورتی که یک باشد، ژن مربوطه از والد اول گرفته می شود و در صورتی که ۲ باشد، ژن مربوطه از والد دوم گرفته می شود.

b. در صورتیکه این ژن قبلا به فرزند منتقل شده باشد، ژن قبلی از آن والد در نظر گرفته می شود.

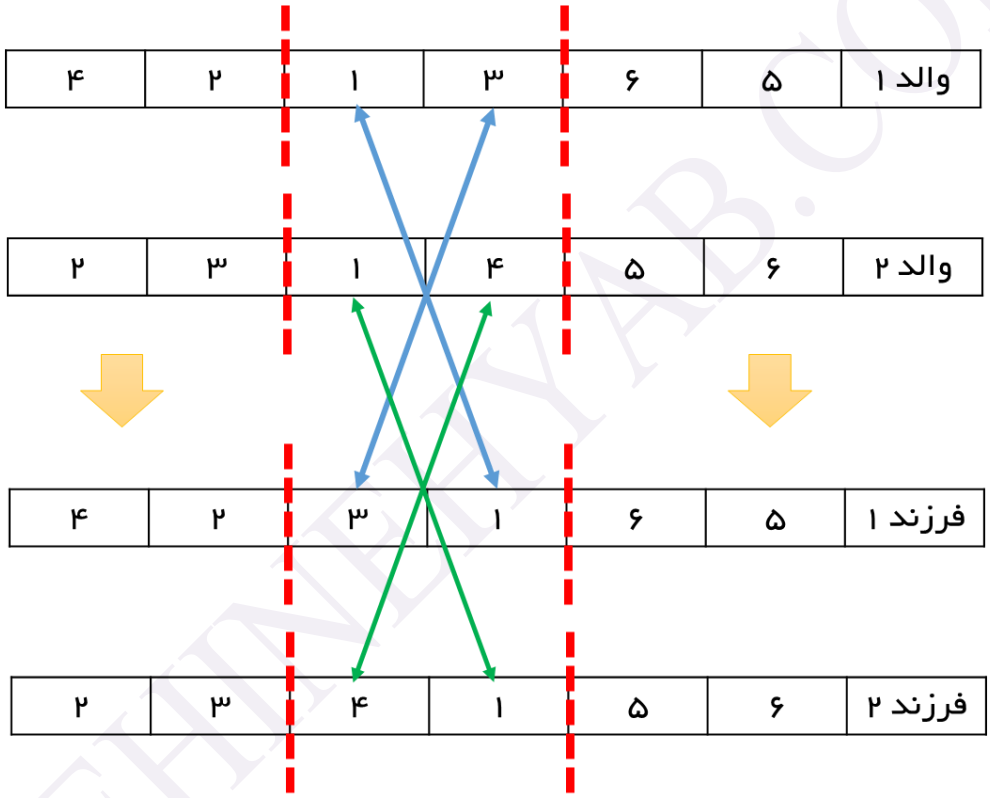
c. برای تمامی ژن‌ها فرزند، عمل فوق انجام می‌گردد و کروموزوم فرزند تکمیل می‌گردد.

مثالی از عملگر *PPC* در شکل زیر نشان داده شده است.

والد ۱	A	B	C	D
والد ۲	C	A	B	F
مقدار تصادفی	۱	2	1	1
فرزند ۱	A	C	B	D
توضیحات	چون مقدار تصادفی ۱ است لذا والد ۱ انتخاب می شود و ژن A می شود.	چون مقدار تصادفی ۲ است، A می تواند انتخاب شود ولی چون ژن قبلی A بوده است C که ژن ما قبلی والد ۲ است Cانتخاب می شود.	چون مقدار تصادفی 1 است، C می تواند انتخاب شود ولی چون ژن قبلی C بوده است C که ژن ما قبلی والد 1 است Bانتخاب می شود.	چون مقدار تصادفی ۱ است لذا والد ۱ انتخاب می شود و ژن D می شود.

تقاطع مرتب ordered crossover

از تقاطع دو نقطه ای مرتب زمانی استفاده می شود که مساله مبتنی بر ترتیب باشد. با دو والد داده شده، دو نقطه تقاطع تصادفی انتخاب می‌گردد که آن‌ها را به سه قسمت چپ، وسط و راست تقسیم می‌کند. عملگر به این ترتیب عمل می نماید. فرزند ۱ قسمت چپ و راست را از والد ۱ و قسمت وسط آن براساس ژن‌های قسمت وسط والد ۱ به نحوی که ترتیب آن براساس والد ۲ باشد تعیین می‌گردد. فرایند مشابه برای فرزند دوم صورت می گیرد. یک مثال از تقاطع مرتب در شکل زیر نمایش داده می شود.



جهش

بعد از تقاطع، کروموزوم‌های تحت عملگر جهش قرار می گیرند. عملگر جهش از افتادن الگوریتم در بهینه محلی جلوگیری می نماید. اگر عملگر تقاطعی برای کاوش روی راه حل‌های اخیر در جهت یافتن راه حل بهتر به کار گرفته شده است. جهش موجب می‌گردد که گوناگونی جمعیت حفظ شده و ساختار ژنتیکی جدیدی در جمعیت با تغییرات تصادفی بعضی از ژن‌ها به وجود آید. عمگر جهش با حفظ گوناگونی جمعیت موجب می‌گردد که از افتادن الگوریتم در بهینه محلی جلوگیری گردد.

معکوس کردن

معکوس کردن یک ژن شامل تغییر صفر به یک و یک به صفر براساس یک کروموزم جهش در یک بردار تصادفی تولید شده است. این کروموزوم جهش، نشان می دهد که چه ژن هایی از والد یـا کرومـوزوم اصـلی باید جهش پیدا کند. برای هر عدد ۱ در کروموزوم جهش، ژن‌های مرتبط با آن باید معکوس گردد(از صفر به یک و از یک به صفر) و کروموزم فرزند تولید گردد. در این شکل، در کروموزم جهش، ۳ عدد ۱ وجود دارد و ژن مرتب از والد معکوس شده اند و کروموزم فرزند تولید شده است.

والد	۱	۰	۱	۰	۱	۱	۰	۱
کروموزوم تصادفی	۱	۰	۰	۱	۰	۰	۰	۱
فرزند	۰	۰	۱	۱	۱	۱	۰	۰

تعویض

در این نوع جهش، دو موقعیت تصادفی از یک رشته انتخاب و ارزش‌های مرتبط آن‌ها با همدیگر تعویض می‌گردد. شکل زیر این نوع جهش را نشان می دهد.

والد	۱	۰	۱	۰	۱	۱	۰	۱
فرزند	۱	۰	۰	۰	۱	۱	۱	۱

احتمال جهش

احتمال جهش یک پارامتر مهم در تکنیک جهش است. احتمال جهش، احتمـال جهـش ژن‌هـای یـک کروموزوم را مشخص می سازد. در صورتی که جهشی رخ ندهد، فرزندان بعد از عمل تقاطع بدون هیچ گونه تغییری تولید می شود. اگر جهش اتفاق بیافتد، یک یا چنـد قسـمت از کرومـوزم تولیـد شـده تغییـر پیـدا می‌کند. اگر احتمال جهش، ۱۰۰ درصد باشد تمامی قسمت‌های کروموزم تغییر پیدا می‌کند و اگـر احتمـال جهش صفر درصد باشد، هیچ تغییری به وجود نمی اید. جهش از افتادن الگوریتم در بهینه محلی جلوگیری می نماید و طراحی مناسب یک نرخ جهش از اهمیت بالایی برخوردار است.

جابه جایی

جابه جایی آخرین مرحله از چرخه تولید مثل است. دو والد از یک جمعیت با اندازه ثابت، انتخاب شده و دو فرزند را به وجود آورده اند. نمی توان کل این چهار کروموزوم راه به جمعیت برگرداند و در نتیجه باید دو کروموزوم حذف گردد. به عبارت دیگر زمانی که فرزندان تولید شدند، باید روشی تعریف گردد که براساس آن مشخص گردد که کدام یک از اعضا فعلی جمعیت باید حذف شده و چه فرزندانی باید جانشین آن‌ها شود. این روش بر همگرایی الگوریتم ژنتیک تاثیر زیادی خواهد داشت. روش‌های مختلفی برای انتخاب جمعیت جدید وجود دارد که به طور مثال می توان از دو روش زیر نام برد.

- ✓ تمامی اعضای جمعیت جدید از میان کروموزوم‌های فرزندان انتخاب شوند.
- ✓ تعدادی از افراد جمعیت مرحله بعد، همان افراد جمعیت مرحله قبل بوده و مابقی از میان فرزندان جدید انتخاب گردند. البته در هر مورد، شایسته ترین کروموزوم‌ها انتخاب می شود.

تحقیقات نشان داده است که حذف همه کروموزوم‌های جمعیت مرحله قبل و انتخاب جمعیت جدید از میان فرزندان، ممکن است بسیاری از جواب‌های مناسب را که در میان جمعیت مرحله قبل وجود دارد، حذف نماید.

قاعده توقف

قواعد توقف متعددی برای الگوریتم ژنتیک وجود دارد که در زیر به طور خلاصه به بعضی از آن اشاره می نماییم.

- ✓ حداکثر تولید نسل. با استفاده از این قاعده، الگوریتم ژنتیک زمانی متوقف می‌گردد که تعداد مشخصی از تولید نسل اتفاق افتاده باشد. مثلا شمارنده تولید نسل به عدد خاصی مثل ۱۰۰ برسد.
- ✓ زمان سپری شده. زمانی که فرایند الگوریتم ژنتیک زمان خاصی را سپری کرد، الگوریتم متوقف می‌گردد.

- ✓ عدم بهبود در برازندگی. در این حالت، در صورتی که هیچ تغییری در بهترین برازندگی جمعیت بعد از تعداد مشخصی تولید نسل به وجود نیاید، الگوریتم ژنتیک متوقف می‌گردد.

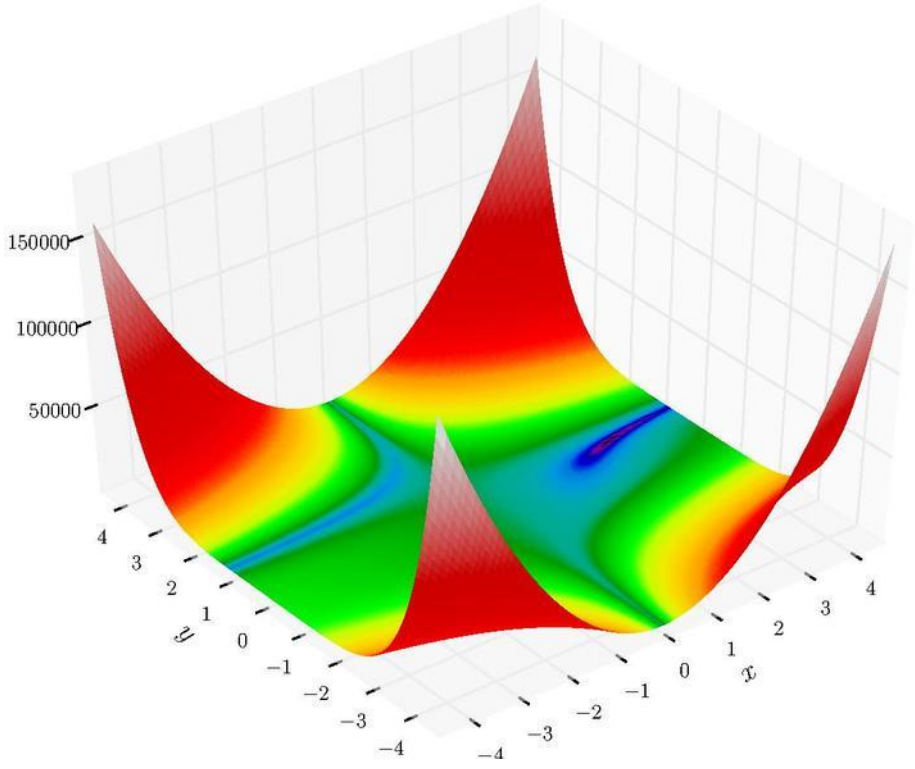
با توجه به موارد گفته شده، در ادامه یک مسئله بهینه سازی با استفاده از الگوریتم فراابتکاری ژنتیک حل می شود.

مثال:

در این قسمت می خواهیم یک مسئله بهینه سازی بدون محدودیت را با استفاده از روش الگوریتم ژنتیک حل نماییم. در این مثال به دنبال کمینه کردن تابع هدف زیر هستیم.

$$f(x,y)=(1.5-x+xy)^2+(2.25-x+xy^2)^2+(2.625-x+xy^3)^2$$

مقدار بهینه تابع هدف این مدل برابر با صفر و مقدار متغیرهای برابر با $(x^*,y^*)=(3,0.5)$ هستند. نمایش سه بعدی این تابع به صورت زیر است.



برای یافتن جواب بهینه این تابع مراحل زیر باید طی شود.

تنظیمات الگوریتم

در این قدم تنظیمات اولیه برای اجرای الگوریتم آورده می شود. نرخ جهش برابر ۰.۱ و تعداد جمعیت برابر ۲۰ در نظر گرفته می شود. بازه ی تغییرات دو متغیر $x \in [0,10]$ و $y \in [0,10]$ است.

تشکیل جمعیت اولیه

برای نمایش متغیرهای x و y به صورت ژن و از رویکرد مبنای دو استفاده شد. هر عدد صحیح در مبنای ده را می توان به صورت یک عدد در مبنای ۲ نمایش داد.

$$(x')_{10} = (a_0, a_1, a_2, ..., a_{n-1}, a_n) \rightarrow x' = \sum_{i=0}^{n} a_i \, 2^i$$

در این تمرین، در صورت استفاده از نمایش دودویی هر جواب، دقت جواب بدون رقم اعشار در نظر گرفته می شود و این برای محاسبه جواب بهینه کافی نیست. برای افزایش دقت و تولید اعداد بین صفر تا ۱۰، اعدادی بر مبنای دو تولید می شود که بین صفر تا ۱۰۲۳ است و سپس عدد تولید شده بر عدد ۱۰۰ تقسیم می شود. برای نمونه مثال زیر را در نظر بگیرید.

$$(x'')_2 = (0,0,1,1,0,0,1,1,0,0)$$
$$\rightarrow (x')_{10} = 0\times 2^0 + 0\times 2^1 + 1\times 2^2 + 1\times 2^3 + 0\times 2^4 + 0\times 2^5 + 1\times 2^6 + 1\times 2^7 + 0\times 2^8 + 0\times 2^9$$
$$= 4 + 8 + 64 + 128 = 204$$

برای تولید یک مقدار برای متغیر x تنها کافی است مقدار تولید شده در بالا را بر ۱۰۰ تقسیم نماییم. در این صورت مقدار تولید شده برای x برابر با 2.04 می شود.

نمایش فوق برای یکی از متغیرها استفاده می شد. برای ایجاد یک عضو از جمعیت، کافی است که دو بردار دودویی متغیرهای x و y را در کنار هم قرار داد و لذا یک بردار ۲۰ عضو(یا ژن) خواهیم داشت. با

توجه به این که تعداد اعضای جمعیت مسئله برابر ۳۰ است، لذا یک بردار ۳۰ در ۲۰ خواهیم داشت. در شکل زیر نمایش جمعیت و ژن های این مدل آورده شده است.

ژن های مربوط به متغیر y

ژن های مربوط به متغیر X

	شماره ژن																				
شماره جمعیت	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵	۱۶	۱۷	۱۸	۱۹
۱	0	0	1	0	1	0	0	1	0	0	0	0	0	1	1	1	0	1	0	1
۲	0	0	0	1	0	0	0	1	0	1	0	0	0	0	1	0	0	1	1	1
۳	1	1	0	1	0	1	0	0	1	1	1	1	0	1	0	1	0	0	0	1
										⋮										
۱۷	1	0	1	1	0	1	0	0	1	0	0	0	1	1	1	0	0	1	1	0
۱۸	0	0	1	1	0	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0
۱۹	0	0	1	0	0	0	1	1	1	0	0	1	0	1	1	0	0	0	0	1
۲۰	0	0	0	0	1	1	0	0	1	0	1	0	0	1	1	1	0	0	1	1

در انتهای این بخش، مقدار تابع هدف براساس دو متغیر x و y برای کلیه اعضای جمعیت برای تابع هدف زیر محاسبه می شود.

$$f(x,y)=(1.5-x+xy)^2+(2.25-x+xy^2)^2+(2.625-x+xy^3)^2$$

روش چرخه رولت

در این قسمت از میان جمعیت دو عضو برای انجام عملیات تقاطع به عنوان والد انتخاب می شود. با توجه به این که در این مسئله به دنبال کمینه کردن مقدار تابع هدف هستیم، لذا در روش چرخه رولت، معکوس تابع هدف را به عنوان تابع برازندگی این چرخه در نظر می گیریم. هر چه مقدار تابع هدف بیشتر باشد برازندگی آن کمتر است و در این صورت در روش چرخه رولت با احتمال کمتری انتخاب می شود.

عملگر تقاطع

در این قسمت از عملگر ساده تقاطع استفاده می شود به این صورت که دو عضو از جمعیت با استفاده از چرخه رولت به صورت تصادفی انتخاب می شود. سپس یک عدد بین ۰ تا ۱۹ انتخاب می شود و از آن محل تقاطع، عملگر تقاطع بین این دو والد انجام می شود و نتیجه دو فرزند می شود. مقدار تابع هدف این دو فرزند محاسبه می شود. در صورتیکه مقدار تابع هدف این دو فرزند بهتر از اعضای جمعیت باشد در جمعیت جایگزین می شود.

عملگر جهش

عملگر جهش در این مسئله برابر با ۰.۱ در نظر گرفته شد و به این صورت عمل می شود که یک عدد تصادفی بین صفر و یک تولید می شود. اگر عدد تولید شده کمتر از ۰.۱ باشد، این عملگر اجرایی می شود و در این صورت یک عدد تصادفی بین صفر و ۱۹ تولید می شود. ژن مربوط به عدد تولید شده در صورتی که صفر باشد به یک و در صورتیکه یک باشد به صفر تغییر می کند.

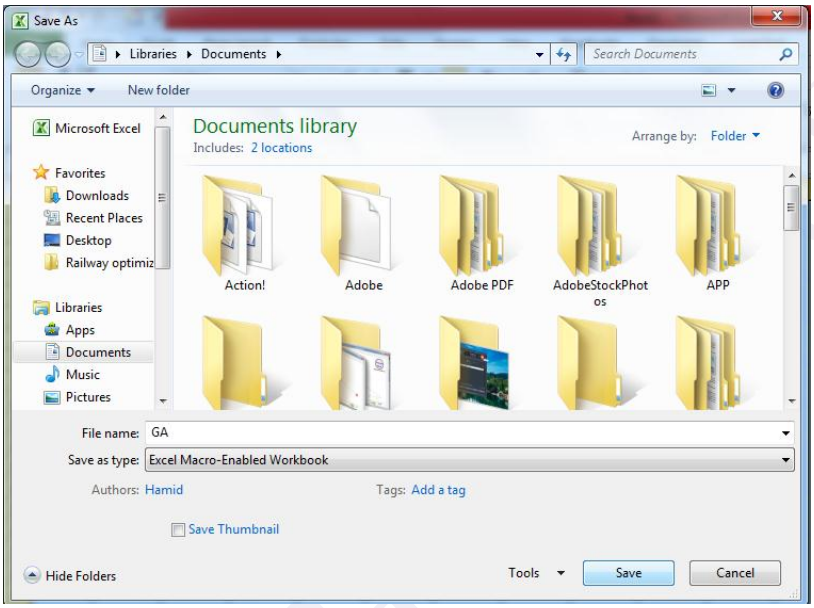
شرط توقف

شرط توقف زمانی برقرار می شود که ۱۰۰۰ تکرار انجام شود.

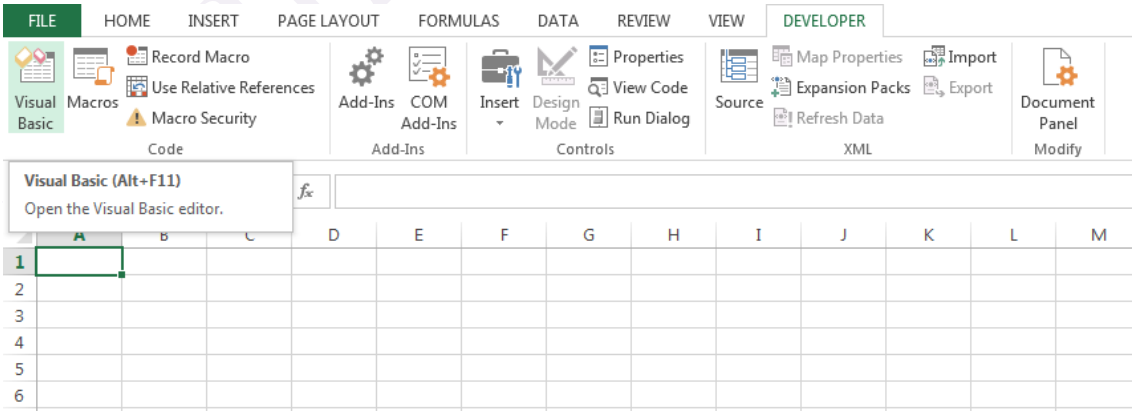
با توجه به موارد ذکر شده اکنون می توان جواب بهینه تابع *f(x,y)* را محاسبه کرد. برای پیاده سازی الگوریتم گرم و سرد کردن شبیه سازی شده از زبان *Visual Basic for Application* که زبان برنامه نویسی در نرم افزارهای *Office* است استفاده می کنیم. این زبان همان زبان برنامه نویسی *Visual Basic* است ولی این امکان در آن فراهم شده است که از قابلیت های *Office* در این زبان بهره برد. در کنار این زبان برنامه نویسی از قابلیت های *Excel* برای نمایش جواب ها استفاده می شود.

در ادامه به صورت جزیی تمامی مراحل حل تابع *f(x,y)* با زبان *Visual Basic for Application* توضیح داده می شود.

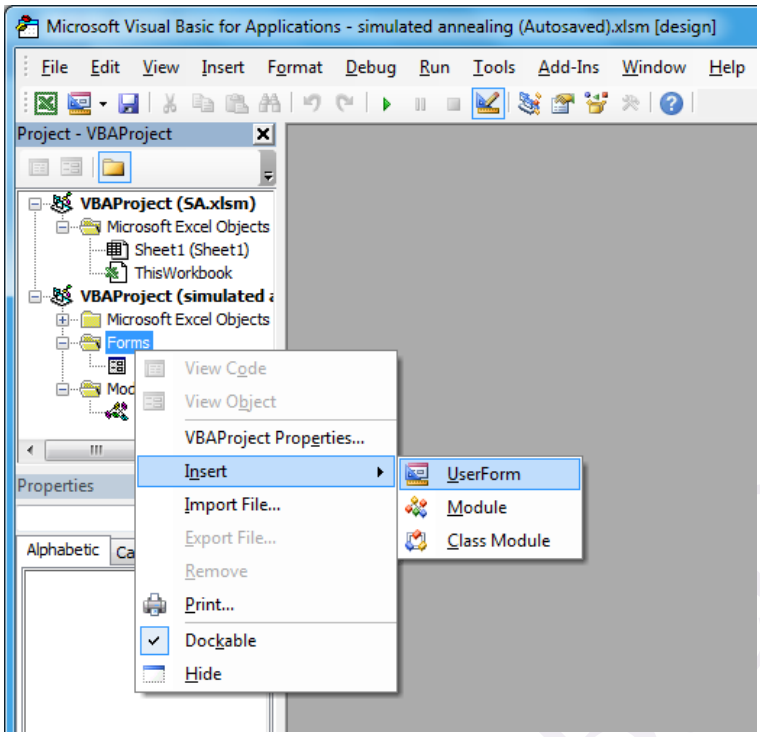
در ابتدا برنامه اکسل را باز می کنید و یک با نام *GA* یک فایل اکسل به فرمت *Excel Macro-Enabled Workbook* ایجاد می کنیم. این فرمت به ما امکان می دهد که در محیط اکسل به توان به زبان *Visual Basic for Application* برنامه نویسی کنیم.



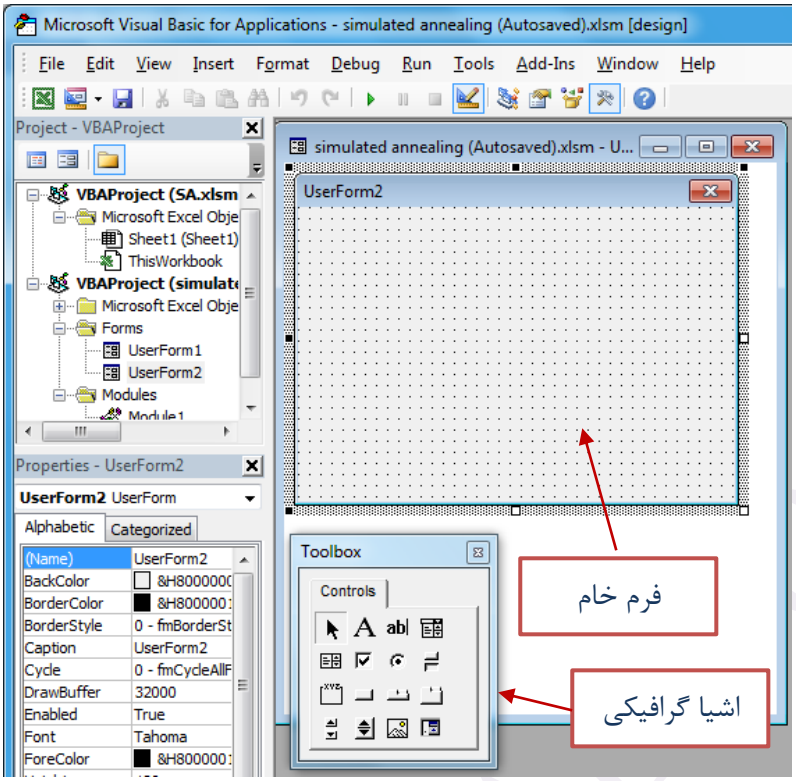
از تب *developer* بر روی دکمه *Visual Basic* کلیک می کنیم تا برنامه *Visual Basic for Application* باز شود.



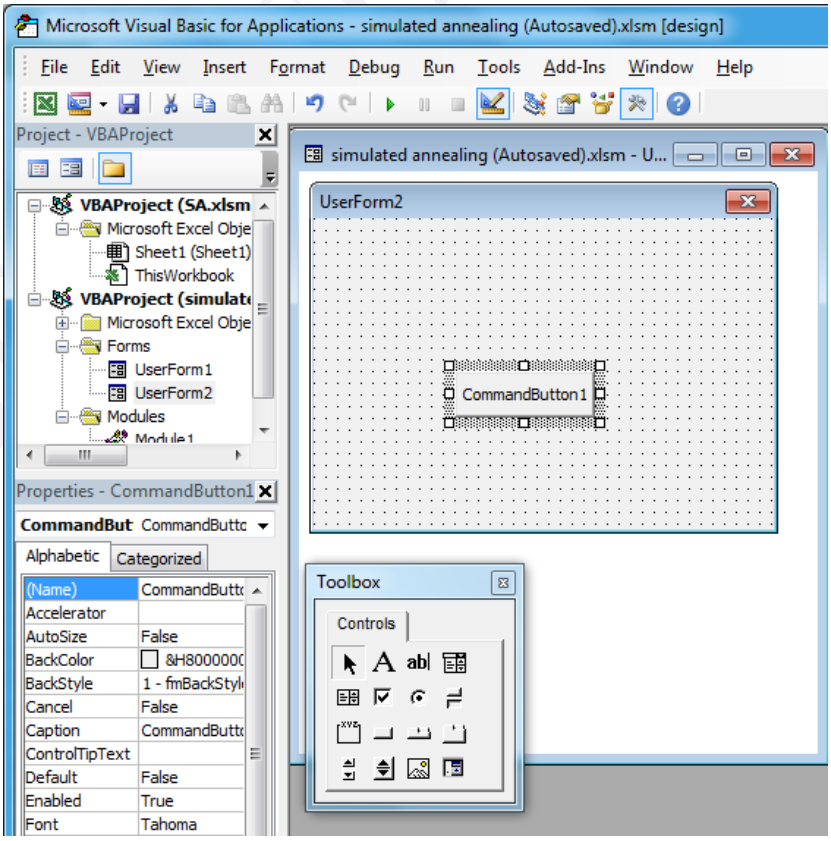
پس از باز کردن این برنامه وارد محیط برنامه نویسی برنامه اکسل می شوید. برای ایجاد یک پروژه می توانید از پنجره *VBA Project* استفاده کنید. در این تمرین با ایجاد یک فرم ساده کار را آغاز می کنیم.



فرم ایجاد شده این امکان را به شما می دهد تا شی های گرافیکی مختلفی را در فرم قرار دهید. البته باید به این نکته اشاره کرد که قابلیت های شما در این زبان برنامه نویسی در اکسل محدود است و در صورت نیاز به اشیا گرافیکی پیشرفته تر باید از سایر نرم افزارها مانند *Visual Studio* استفاده کنید. پس از باز کردن یک فرم جدید، نام آن را *Userform*۱ می نامیم و فرمی به صورت زیر باز می شود.

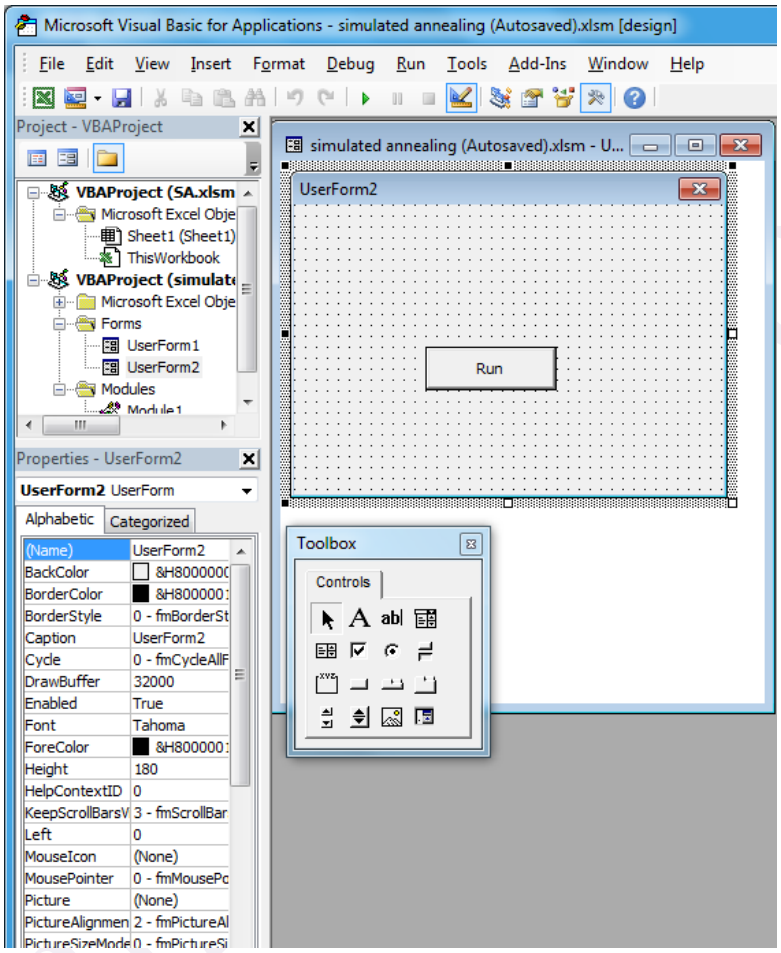


برای ایجاد برنامه تنها به یک دکمه یا *button* نیاز است که با درگ از *Toolbox* و گذاشت در فرم خام ایجاد می شود. که در این صورت فرم شما به صورت زیر می شود.

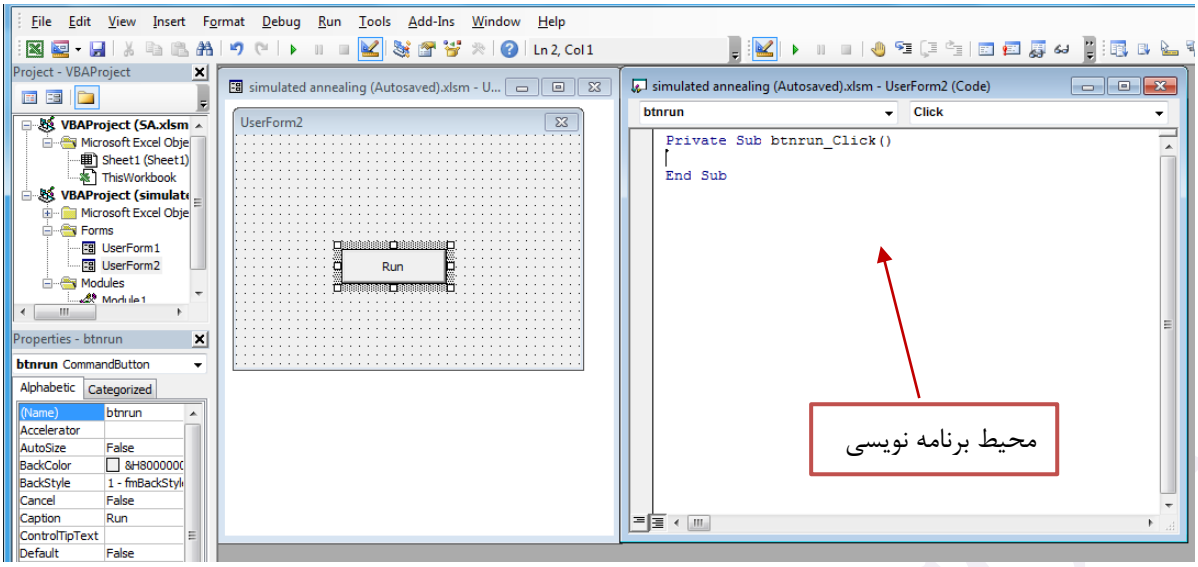


نام دکمه یا همان *Caption* را به *Run* و نام مشخصه دکمه یا *Name* را به *Btnrun* تغییر مـی دهـیم.

نام گذاری اشیا دارای استانداردی است که در صورت علاقه می توانید به کتاب های آموزش برنامـه نویسـی مراجعه کنید.پس از این تغییرات به فرم زیر می رسیم.



برای اضافه کردن برنامه باید روی دکمه *Run* دوبار کلیک کرد و به محیط برنامه نویسی وارد می شویم.



در محیط برنامه نویسی کد زیر را وارد می کنیم.

توجه: در این برنامه، اندازه جمعیت برابر ۳۰ در نظر گرفته شده است.

```
Dim pop(۳۰, ۲۰) As Integer ' size of population is ۳۰. Each Koromozon has  ۲۰gens.
Dim popvalue(۳۰) As Double ' size of population is .۳۰
Dim barpop(۳۰) As Double ' size of population is .۳۰
Dim cumaldistrupop(۳۰) As Double ' size of population is .۳۰
Dim rmutation As Double
Dim rrollet As Double
Dim selectedpop ۱As Integer
Dim selectedpop ۲As Integer
Dim temppop(۲۰) As Integer
Dim averagefucntion(۱۰۰۲) As Double
'setting paratmeter
rmutation = ۰.۱
Randomize
For i =  ۱To ۳۰
For j =  ۰To ۱۹
pop(i, j) = Math.Rnd(۱
Next
Next
Dim xp As Double
xp = ۰
Dim yp As Double
yp = ۰
```

```
Dim Base(۱۰) As Integer
Base(۰) = ۱
For i = ۱To ۹
Base(i) = Base(i - ۱) * ۲
Next
For j = ۱To ۳۰
xp = ۰
yp = ۰
For i = ۰To ۹
xp = pop(j, i) * Base(i) + xp
Next
For i = ۱۰To ۱۹
yp = pop(j, i) * Base(i - ۱۰) + yp
Next
x = ۰
y = ۰
x = xp / ۱۰۰
y = yp / ۱۰۰
popvalue(j) = ( ۱.۵- x + x * y) * ( ۱.۵- x + x * y) + ( ۲.۲۵- x + x * y * y) * ( ۲.۲۵- x + x * y * y)
+ ( ۲.۶۲۵- x + x * y * y * y) * ( ۲.۶۲۵- x + x * y * y * y(
Next
'reputation
For geneticalgorithm = ۱To ۱۰۰۰
'calculating Barazandegi function
Dim sumpopvalue As Double
sumpopvalue = ۰
For i = ۱To ۳۰
sumpopvalue = ۱/ popvalue(i) + sumpopvalue
Next
For i = ۱To ۳۰
barpop(i) = ۱/ (popvalue(i((
barpop(i) = barpop(i) / sumpopvalue
Next
cumaldistrupop(۰) = ۰
For i = ۱To ۳۰
cumaldistrupop(i) = cumaldistrupop(i - ۱) + barpop(i(
```

```
Next
Randomize
rrollet = Math.Rnd
Dim counter As Integer
counter = ۱
While rrollet > cumaldistrupop(counter(
counter = counter + ۱
Wend
selectedpop ۱= counter
counter = ۱
Randomize
rrollet = Math.Rnd
counter = ۱
While rrollet > cumaldistrupop(counter(
counter = counter + ۱
Wend
selectedpop ۲= counter
Dim crossover As Integer
crossover = Int(Math.Rnd() * ۱۹) + ۱'generating a number between ۱and ۹
Dim newpop۱(۲۰) As Integer
Dim newpop۲(۲۰) As Integer
Dim newpop۱value As Double
Dim newpop۲value As Double
For i = ۰To ۲۰
newpop۱(i) = pop(selectedpop۱, i(
newpop۲(i) = pop(selectedpop۲, i(
Next
For i = ۰To crossover
temppop(i) = newpop۱(i(
newpop۱(i) = newpop۲(i(
newpop۲(i) = temppop(i(
Next
xp = ۰
yp = ۰
For i = ۰To ۹
xp = newpop۱(i) * Base(i) + xp
Next
```

```
For i = ۱۰To ۱۹
yp = newpop۱(i) * Base(i - ۱۰) + yp
Next
x = xp / ۱۰۰
y = yp / ۱۰۰
newpop۱value = ( ۱.۵- x + x * y) * ( ۱.۵- x + x * y) + ( ۲.۲۵- x + x * y * y) * ( ۲.۲۵- x + x * y * y) + ( ۲.۶۲۵- x + x * y * y * y) * ( ۲.۶۲۵- x + x * y * y * y(
xp = ۰
yp = ۰
For i = ۰To ۹
xp = newpop۲(i) * Base(i) + xp
Next
For i = ۱۰To ۱۹
yp = newpop۲(i) * Base(i - ۱۰) + yp
Next
x = xp / ۱۰۰
y = yp / ۱۰۰
newpop۲value = ( ۱.۵- x + x * y) * ( ۱.۵- x + x * y) + ( ۲.۲۵- x + x * y * y) * ( ۲.۲۵- x + x * y * y) + ( ۲.۶۲۵- x + x * y * y * y) * ( ۲.۶۲۵- x + x * y * y * y(
'sorting population
Dim arraytemp As Double
arraytemp = ۰
For i = ۱To ۳۰
For j = ۱To ۲۹
If popvalue(j) > popvalue(j + ۱) Then
  arraytemp = popvalue(j(
  popvalue(j) = popvalue(j + (۱
  popvalue(j + ۱) = arraytemp
  For t = ۰To ۲۰
   arraytemp = pop(j, t(
   pop(j, t) = pop(j + ۱, t(
   pop(j + ۱, t) = arraytemp
  Next ' t
 end If
```

```
Next ' j
Next  ' i
If popvalue(۲۹) > newpop۱value And popvalue(۲۹) > newpop۲value Then
If newpop۱value < newpop۲value Then
For t =  ۰To ۲۰
pop(۲۹, t) = newpop۱(t(
pop(۳۰, t) = newpop۲(t(
Next
popvalue(۲۹) = newpop۱value
popvalue(۳۰) = newpop۲value
End If
End If
If popvalue(۲۹) > newpop۱value And popvalue(۲۹) < newpop۲value Then
For t =  ۰To ۲۰
pop(۲۹, t) = newpop۱(t(
Next
popvalue(۲۹) = newpop۱value
End If
If popvalue(۲۹) < newpop۱value And popvalue(۲۹) > newpop۲value Then
For t =  ۰To ۲۰
pop(۲۹, t) = newpop۲(t(
Next
popvalue(۲۹) = newpop۲value
End If
If popvalue(۲۹) < newpop۱value And popvalue(۲۹) < newpop۲value Then
If popvalue(۳۰) > newpop۱value And popvalue(۳۰) > newpop۲value Then
If newpop۱value < newpop۲value Then
For t =  ۰To ۲۰
pop(۳۰, t) = newpop۱(t(
Next
popvalue(۳۰) = newpop۱value
Else
For t =  ۰To ۲۰
pop(۳۰, t) = newpop۲(t(
Next
popvalue(۳۰) = newpop۲value
End If
```

```
End If
End If
'sorting before mutation
'Dim arraytemp As Double
arraytemp = ۰
For i =  ۱To ۳۰
For j =  ۱To ۲۹
If popvalue(j) > popvalue(j + ۱) Then
    arraytemp = popvalue(j(
    popvalue(j) = popvalue(j + (۱
    popvalue(j + ۱) = arraytemp
    For t =  ۰To ۲۰
     arraytemp = pop(j, t(
     pop(j, t) = pop(j + ۱, t(
     pop(j + ۱, t) = arraytemp
    Next ' t

End If
Next ' j
Next  ' i
'mutation
Dim mut As Double
If mut < rmutation Then
Dim muran As Double
Dim mutrangen As Double
muran = Int(Math.Rnd() * ۳۰) + ۱
mutrangen = Int(Math.Rnd * ۲۰) + ۱
For i =  ۰To ۲۰
newpop۱(i) = pop(muran, i(
'newpop۲(i) = pop(selectedpop۲, i(
Next
xp = ۰
yp = ۰
For i =  ۰To ۹
xp = newpop۱(i) * Base(i) + xp
Next
For i =  ۱۰To ۱۹
```

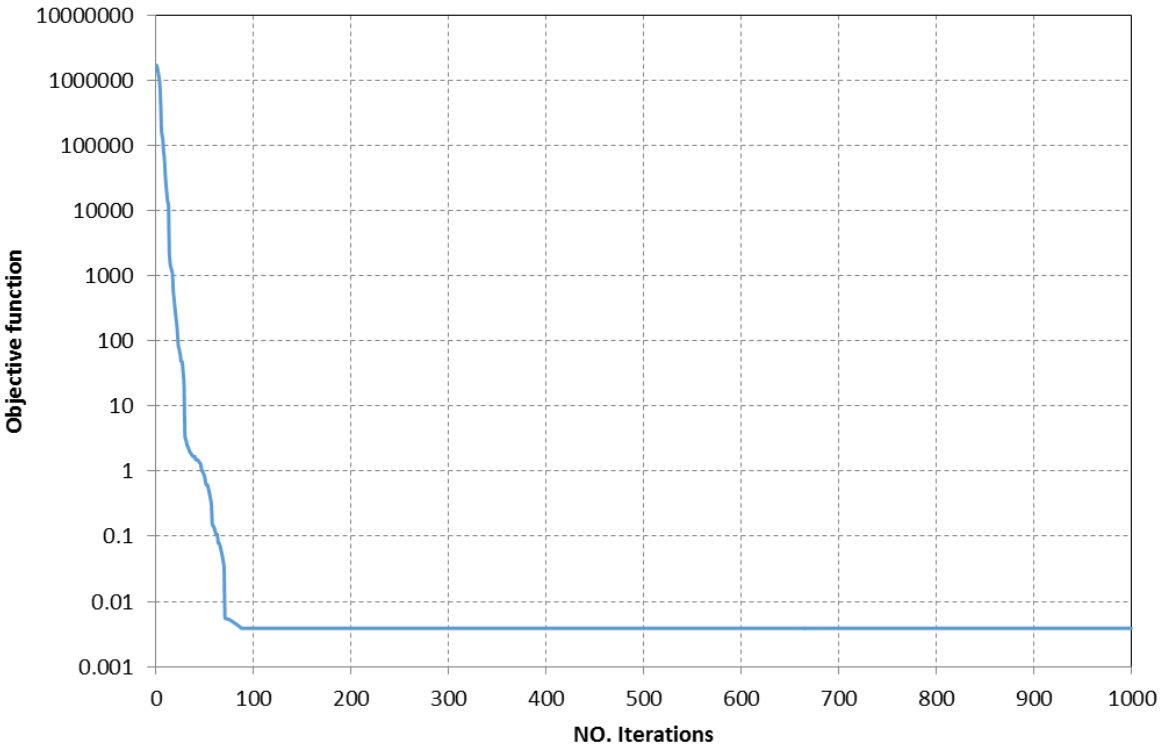
```
yp = newpop\(i) * Base(i - ۱۰) + yp
Next
x = xp / ۱۰۰
y = yp / ۱۰۰
newpop\value = ( ۱.۵- x + x * y) * ( ۱.۵- x + x * y) + ( ۲.۲۵- x + x * y * y) * ( ۲.۲۵- x + x * y
* y) + ( ۲.۶۲۵- x + x * y * y * y) * ( ۲.۶۲۵- x + x * y * y * y(
If popvalue(۳۰) > newpop\value Then
For t =  ۰To ۲۰
pop(۳۰, t) = newpop\(t(
Next
popvalue(۳۰) = newpop\value
End If
End If
'sorting after mutation
'Dim arraytemp As Double
arraytemp = ۰
For i =  ۱To ۳۰
For j =  ۱To ۲۹
If popvalue(j) > popvalue(j + ۱) Then
    arraytemp = popvalue(j(
    popvalue(j) = popvalue(j + (۱
    popvalue(j + ۱) = arraytemp
    For t =  ۰To ۲۰
     arraytemp = pop(j, t(
     pop(j, t) = pop(j + ۱, t(
     pop(j + ۱, t) = arraytemp
    Next ' t

End If
Next ' j
Next  ' i
Dim avera As Double
avera = ۰
For t =  ۱To ۳۰
avera = avera + popvalue(t(
Next
```

```
avera = avera / ۳۰
averagefucntion(geneticalgorithm) = avera
Next
'reporting x and y variables
Dim xvariable(۳۰) As Double
Dim yvariable(۳۰) As Double
Dim objectivefu(۳۰) As Double
For t = ۱To ۳۰
xp = ۰
yp = ۰
For i = ۰To ۹
xp = pop(t, i) * Base(i) + xp
Next
For i = ۱۰To ۱۹
yp = pop(t, i) * Base(i - ۱۰) + yp
Next
x = xp / ۱۰۰
y = yp / ۱۰۰

newpop۱value = ( ۱.۵- x + x * y) * ( ۱.۵- x + x * y) + ( ۲.۲۵- x + x * y * y) * ( ۲.۲۵- x + x * y
* y) + ( ۲.۶۲۵- x + x * y * y * y) * ( ۲.۶۲۵- x + x * y * y * y(
xvariable(t) = x
yvariable(t) = y
objectivefu(t) = newpop۱value
Next
For t = ۱To ۱۰۰۰
Cells(t, ۱۷).Value = averagefucntion(t(
Next
```

پس از اجرای برنامه فوق، الگوریتم ژنتیک توانست به جواب بهینه تابع هدف با دقت مناسبی نزدیک شود. در شکل زیر تغییرات مقدار میانگین تابع هدف جمعیت برای تکرارهای مختلف آورده شده است.



همان طور که ملاحظه می شود که در کمتر از ۱۰۰ تکرار به جواب بهینه با دقت مناسب رسیده است. شایان یاد است که مقدار بهینه تابع هدف برابر صفر است.

برای استفاده دانشجویان و محققان محترم فایل برنامه فوق به صورت فایل متنی به پیوست این جزوه در اختیار شما قرار می گیرد که می توانید از آن در پروژه های خود استفاده کنید.

برای دریافت بسته‌های آموزشی گروه **بهینه یاب** به وب سایت ما به نشانی www.behinehyab.com مراجعه کنید.

در صورت هر گونه سوال از طریق ایمیل به نشانی behinehyab@gmail.com و یا بخش تماس با ما وب سایت گروه **بهینه یاب** با ما در تماس باشید.

با تشکر از توجه شما

گروه آموزشی **بهینه یاب**