بسم الله الرحمن الرحیم

الگوریتم های فرا اکتشافی جستجو

الگوریتم های ژنتیک

گردآورنده : مصطفی عباسی کیا

این کتاب یک کتاب رایگان است و می توان مطالب موجود در آن را با /بدون اجازه گردآورنده توزیع کرده یا تغییر داد. این کتاب به این امید گردآوری شده است که بتواند در ارتقای سطح علمی دوستان موثر واقع شود. از خواننده محترم تقاضا می شود با ارسال نظرات و پیشنهادات خود در مورد این کتاب و یا اشکالات موجود در محتوی یا متن آن به آدرس m.abbasi.kia@gmail.com در بهبود این کتاب سهیم شود

با تشکر

مصطفی عباسی کیا

بهار 1388

فهرست

درباره علم ژنتیک 7
مقدمه ای بر بهینه سازی 8
الگوریتم های مینیمم یابنده 8
هیوریستیک ها 9
هیوریستیک 10
انواع الگوریتم‌های هیوریستیک 11
معرفی کلی الگوریتم ژنتیک 12
مکانیزم الگوریتم ژنتیک 15
عملگرهای الگوریتم ژنتیک 18
چارت و کد الگوریتم 19
شبه کد وتوضیح 20
چارت الگوریتم 21
کُدینگ و ساختمان داده 22
کدینگ باینری 23
کدینگ جایگشتی 24
کدگذاری مقدار 25
کدینگ درخت 25
نمایش رشته ها 26
متغیر های پیوسته 26
متغیر های گسسته 27
انواع روش های تشکیل رشته(کدینگ باینری) 27
روش سری 28
روش محاطی 28
باز گرداندن رشته ها به مجموعه متغیرها 28

تعداد بیتهای متناظر با هر متغیر 29
جمعیت 30
ایجاد جمعیت اولیه 30
اندازه جمعیت 31
محاسبه برازندگی(تابع ارزش) 32
انتخاب (selection) 33
انتخاب چرخ رولت 34
انتخاب ترتیبی 35
انتخاب بولتزمن 36
انتخاب حالت پایدار 36
نخبه سالاری 37
انتخاب رقابتی 37
انتخاب قطع سر 38
انتخاب قطعی بریندل 38
انتخاب جایگزینی نسلی اصلاح شده 38
انتخاب مسابقه 38
انتخاب مسابقه تصادفی 39
ترکیب(crossover) 39
جا به جائی دودوئی 39
جا بجائی حقیقی 42
روشهای پیاده سازی عملگر ترکیب 43
ترکیب تک نقطه ای 43
ترکیب دو نقطه ای 44
ترکیب n نقطهای 45
ترکیب یکنواخت 45

ترکیب حسابی 45
ترتیب 46
چرخه 46
محدب 47
بخش- نگاشته 48
احتمال ترکیب 49
تحلیل مکانیزم جا بجائی 49
جهش 50
تقسیم بندی روش های جهش 51
جهش باینری 51
جهش حقیقی 52
چند روش برای پیاده سازی عملگر جهش 52
وارونه سازی بیت 52
تغییر ترتیب قرارگیری 53
وارون سازی 53
تغییر مقدار 53
احتمال جهش 54
محک اختتام اجرای الگوریتم ژنتیک 54
استدلال همگرایی الگوریتم ژنتیک 55
طرحواره 55
تاثیر انتخاب 55
تاثیر ترکیب 57
تاثیر جهش 59
قضیه طرحواره 60
پردازش مفید 60

انواع الگوریتم های ژنتیکی..... 62
الگوریتم ژنتیکی سری..... 62
الگوریتم ژنتیک موازی..... 63
برنامه ریزی ژنتیک 63
مقایسهٔ الگوریتم ژنتیک با سیستم‌های طبیعی. 65
تقاط قوت وضعف الگوریتم های ژنتیک..... 66

NP-hard Problems 68

محدودیتهای GA..... 69
استراتژی برخورد با محدودیت‌ها 69
استراتژی اصلاح عملگرهای ژنتیک 69
استراتژی ردی..... 70
استراتژی اصلاحی..... 70
استراتژی جریمه‌ای..... 70
بهبود الگوریتم ژنتیک..... 71
چند نمونه از کاربرد های الگوریتم های ژنتیک..... 71
یک مثال ساده..... 72

مسائل حل شده..... 75
الگوریتم ژنتیک و حل مساله TSP 75
مقدمه..... 75
حل مسئله TSP به وسیله الگوریتم ژنتیک..... 76
مقایسه روشهای مختلف الگوریتم ژنتیک برای TSP..... 84
نتیجه گیری..... 85
الگوریتم ژنتیک و حل معمای سودوکو(Sodocu) 86
صورت مسئله 86

حل مسئله 87
تعیین کروموزم..... 87
ساخت جمعیت آغازین یا نسل اول..... 88
ساخت تابع از ارزش..... 89
ترکیب نمونه‌ها و ساختن جواب جدید..... 90
ارزشیابی مجموعه جواب..... 92
ساختن نسل بعد..... 92
بهینه سازی چینش حروف فارسی بر روی صفحه کلید با استفاده از الگوریتم های ژنتیکی 94
چکیده..... 94
مقدمه..... 94
مساله 96
حل مسئله با الگوریتم ژنتیک..... 96
جمعیت..... 97
تابع تناسب..... 98
عملگر های ژنتیکی..... 99
کارایی..... 100
حل مسئله ضرب ماتریس ها به وسیله الگوریتم ژنتیک(کد) 102
لغت نامه (Glossary) 103
منابع و ماخذ..... 108

درباره علم ژنتیک

مقوله ژنتیک با انتشار کتاب مهم و جنجالی چارلز داروین انگلیسی که در آن فرضیه تکاملی خود را مطرح کرده بود درتاریخ 24 نوامبر سال 1859 میلادی به طور جدی مطرح شد.او در کتاب خود که در فارسی با نام <بنیاد انواع> شهرت پیدا کرده است,مبانی فکری و فرضیه خود دال برتکامل سیر پیش رونده و روبه جلوی <خلقت تدریجی> را ارائه کرد.

ناگفته پیداست که این مساله در تعارض شدید با نظریات کلیسای آن زمان که به آفرینش آنی و <خلقت دفعی > انسان معتقد بود، قرار داشت.

در سال 1865 میلادی تحقیقات گریگوری مندل کشیش اتریشی درباره وراثت و تکامل و اصولی که به طور تجربی به دست آورده بود چند سال پس از مرگش انتشار یافت. این تحقیقات توجه بسیاری را معطوف به این موضوعات نمود. در سال 1903 <کروموزوم > به عنوان واحد وراثت معرفی گردید. درسال 1905 برای اولین بار واژه <ژنتیک>توسط یک زیست شناس انگلیسی به نام ویلیام بیتسون وضع گردید و مورد استفاده قرار گرفت.

در سال 1927 واژه <جهش> برای بیان تغییرات فیزیکی در ژنها وضع شد. در سال 1931 واژه <برش> یا <همبُری> وضع گردید. در سال 1953 میلادی ساختار DNA بطور کامل به شکل مارپیچی توسط جیمز واتسون وفرانسیس کریک توضیح داده شد که برای آنان جایزه نوبل را نیز به ارمغان آورد.در سال 1977 میلادی اولین تلاشها برای دستیابی به مدل کامل ژنتیکی یعنی ژنوم انسانی به بار نشست.

مشکلات بسیاری همچون تحقیقات بسیار پر هزینه و زمان طولانی نسلها برای مطالعات ژنتیکی و نیز عدم امکان ایجاد ازدواجهای با برنامه مانند موجودات ازمایشگاهی همچنان خودنمایی می کرد.

پیش بینی می‌شود پروژه <ژنومیک> یعنی نقشه برداری کامل ژنتیکی انسان تا پایان سال 2005میلادی به انجام برسد. در صورت وقوع چنین امری، اهمیت این مساله همردیف با کشف آتش یا اختراع خط ارزیابی می‌شود.

هم اکنون شاخه های فراوانی در علم ژنتیک مشغول بسط بیش از پیش دامنه خود هستند:

بررسی کروموزوم ها (سیتو ژنتیک)

مطالعه ساختمان ژنها (ژنتیک مولکولی)

تشخیص بیماریها (ژنتیک بالینی)

ژنتیک اپیدمیولوژی

ژنتیک توسعه

ژنتیک جمعیت

مقدمه ای بر بهینه سازی

بهینه سازی به فرآیند بهتر کردن هر چیزی اطلاق می شود .یک مهندس و یا یک محقق ایده جدیدی را خلق میکند و بهینه سازی به این ایده خلق شده کیفیت می بخشد . در فرایند بهینه سازی تغییراتی بر روی ایده اولیه انجام می شود و با نتایج حاصل از این تغییرات، ایده اولیه بهبود می یابد. تا مادامی که بتوان ایده مورد نظر را در غالب الکترونیکی نوشت، کامپیوتر وسیله ای مناسب برای بهینه سازی خواهد بود .بدلیل محدودیت های الگوریتم موضعی ، افراد به سمت استفاده کردن از روشهای جامع تر که بر پایه فرایندهای بیولوژیکی می باشند، تغییر جهت داده اند واژه" جواب بهتر" دال بر این میباشدکه بیش از چند جواب با مقادیر نایکسان برای مسئله مورد بررسی وجود دارد . بهتر درنظر گرفتن یک جواب به مسئله مورد بررسی، روش بررسی مسئله و محدوده تغییرات بستگی دارد. بعضی از مسائل دارای جواب مشخص میباشند . در این بخش با اشاره ای کوتاه به الگوریتم های بهینه سازی قدیمی و ذکر معایب آنها به لزوم استفاده از *GA* می پردازیم

الگوریتم های مینیمم یابنده

جستجوی سطح هزینه (تمامی مقادیر ممکن تابع) جهت یافتن هزینه مینیمم در بطن تمامی الگوریتم های بهینه سازی قرار دارد. الگوریتم بهینه سازی قدیمی را میتوان در 4 دسته تقسیم بندی کرد

* جستجوی جامع
* بهینه سازی تحلیلی
* روش شیب دار نلدر- مید
* بهینه سازی مبتنی بر مینیمم سازی خطی

عیب الگوریتم های قدیمی ذکر شده در میزان سرعت همگرایی و مهمتر از آن پیدا نکردن اکسترمم سراسری است. از این رو الگوریتمهای کارا وپر قدرتی همچون روش های بهینه سازی طبیعی، پا به عرصه ظهور گذاشتند. تعدادی از این الگوریتم ها عبارتند از : الگوریتم ژنتیک (هولند 1975)، حرارت دهی تشبیهی (کیریک

پاتریک و همکاران 1983)، بهینه سازی گروه ذرات (پاراساپولوس و وارهاتیس 2002)، بهینه سازی مستعمری (دوریگو وماریا 1997) ونهایتاً الگوریتم های تکاملی (اسونل 1995). تمامی این الگوریتم های جدید با اعمال عملگرهائی روی نقاط اولیه، نقاط جدیدی را در فضای جستجوی تابع هزینه تولید میکنند و تدریجاً به سوی مکانهای بهینه این فضا پیش میروند. این روشها مبتنی بر نوعی جستجوی هوشمند در فضایی بزرگ اما محدود هستند .این الگوریتم ها برخلاف الگوریتم های قدیم نیازی به محاسبه مشتقات توابع ندارند و از رو محدودیتی برای توابع هزینه ناپیوسته و همچنین متغییر های گسسته پیش نمی آید .

در بسیاری از روش های بهینه سازی ، ما با احتیاط از یک نقطه منفرد در فضای تصمیم گیری با استفاده از قوانین انتقال ، به نقطه بعدی می رویم. این روش نقطه به نقطه ممکن است منجر به بروز اشتباه و خطا شود.

زیرا شامل ترشیح کامل از موقعیت قله های اشتباه در فضای مودی (چندین قله) می باشد. در مقابل ، الگوریتم های ژنتیک بطور هم زمان با مجموعه ای از نقاط (جمعی از رشته ها) کار می کنند و قله های زیادی را به طور موازی با یکدیگر جستجو می کنند بنابراین ، احتمال پیدا کردن قله های اشتباه نسبت به روش های نقطه به نقطه ، کاهش می یابد.

در واقع این رهیافت ها بیانگر فرآیند های موجود در طبیعت میباشند که بطور چشمگیری در بهینه سازی مسائل طبیعی موفق هستند. در ادامه کار به توضیح الگوریتم ژنتیک میپردازیم که بحث اصلی این تحقیق می باشد

هیوریستیک ها

سیستم‌های پیچیده اجتماعی تعداد زیادی از مسائل دارای طبیعت ترکیباتی را پیش روی ما قرار می‌دهند. مسیر کامیونهای حمل و نقل باید تعیین شود، انبارها یا نقاط فروش محصولات باید جایابی شوند، شبکه‌های ارتباطی باید طراحی شوند، کانتینرها باید بارگیری شوند، رابط‌های رادیویی می‌بایست دارای فرکانس مناسب باشند، مواد اولیه چوب، فلز، شیشه و چرم باید به اندازه‌های لازم بریده شوند؛ از این دست مسائل بی‌شمارند. تئوری پیچیدگی به ما می گوید که مسائل ترکیباتی اغلب پلی‌نومیال نیستند .این مسائل در اندازه‌های کاربردی و عملی خود به قدری بزرگ هستند که نمی‌توان جواب بهینه آنها را در مدت زمان قابل پذیرش به دست آورد. با این وجود، این مسائل باید حل شوند و بنابراین چاره‌ای نیست که به جوابهای زیر بهینه بسنده نمود به گونه‌ای که دارای کیفیت قابل پذیرش بوده و در مدت زمان قابل پذیرش به دست آیند.

چندین رویکرد برای طراحی جوابهای با کیفیت قابل پذیرش تحت محدودیت زمانی قابل پذیرش پیشنهاد شده است. الگوریتم‌هایی هستند که می‌توانند یافتن جوابهای خوب در فاصله مشخصی از جواب بهینه را تضمین کنند که به آنها الگوریتم‌های تقریبی می‌گویند. الگوریتم‌های دیگری هستند که تضمین می‌دهند با احتمال بالا جواب نزدیک بهینه تولید کنند که به آنها الگوریتم‌های احتمالی گفته می‌شود. جدای از این دو دسته، می‌توان الگوریتم‌هایی را پذیرفت که هیچ تضمینی در ارائه جواب ندارند اما بر اساس شواهد و سوابق نتایج آنها، به طور متوسط بهترین تقابل کیفیت و زمان حل برای مسئله مورد بررسی را به همراه داشته‌اند. به این الگوریتم‌ها، الگوریتم‌های هیوریستیک گفته می‌شود.

هیوریستیک

هیوریستیک‌ها عبارتند از معیارها، روشها یا اصولی برای تصمیم‌گیری بین چند گزینه خط‌مشی و انتخاب اثربخش‌ترین برای دستیابی به اهداف مورد نظر .هیوریستیک‌ها نتیجه برقراری اعتدال بین دو نیاز هستند: نیاز به ساخت معیارهای ساده و در همان زمان توانایی تمایز درست بین انتخاب‌های خوب و بد.

یک هیوریستیک می‌تواند حسابی سرانگشتی باشد که برای هدایت یک دسته از اقدامات به کار می‌رود. برای مثال، یک روش مشهور برای انتخاب طالبی رسیده عبارتست از فشار دادن محل اتصال به ریشه از یک طالبی نامزد انتخاب و سپس بو کردن آن محل. اگر بوی آن محل مانند بوی داخل طالبی باشد آن طالبی به احتمال زیاد رسیده است. این قاعده سرانگشتی نه تضمین می‌کند که تنها طالبی‌های رسیده به عنوان نامزد انتخاب شوند و نه تضمین می‌کند که طالبی‌های رسیده آزمایش شده، رسیده تشخیص داده شوند اما به هر حال این روش، اثربخش‌ترین روش شناخته شده است.

به عنوان مثالی دیگر از استفاده هیوریستیک‌ها، یک استاد بزرگ شطرنج را در نظر بگیرید که با انتخاب بین چندین حرکت ممکن روبرو شده است. وی ممکن است تصمیم بگیرد که یک حرکت خاص، اثربخش‌ترین حرکت خواهد بود زیرا موقعیتی فراهم می‌آورد که «به نظر می‌رسد» بهتر از موقعیت‌های حاصل از حرکت‌های دیگر باشد. به کارگیری معیار «به نظر می‌رسد» خیلی ساده‌تر از تعیین دقیق حرکت یا حرکاتی خواهد بود که حریف را مجبور به مات کند. این واقعیت که اساتید بزرگ شطرنج همواره پیروز بازی نخواهند بود نشان دهنده این است که هیوریستیک‌های آنها انتخاب اثربخش‌ترین حرکت را تضمین نمی‌کنند. نهایتا وقتی از آنها خواسته

می‌شود که هیوریستیک خود را تشریح نمایند آنها فقط توصیفی ناقص از قواعدی ارائه می‌دهند و به نظر خود آنها، انجام آن قواعد از توصیف آنان ساده‌تر است.

خاصیت هیوریستیک‌های خوب این است که ابزار ساده‌ای برای تشخیص خط‌مشی‌های بهتر ارائه دهند و در حالی که به صورت شرطی لازم، تشخیص خط‌مشی‌های اثربخش را تضمین نمی‌کنند اما اغلب به صورت شرط کافی این تضمین را فراهم آورند. بیشتر مسائل پیچیده نیازمند ارزیابی تعداد انبوهی از حالت‌های ممکن برای تعیین یک جواب دقیق می‌باشند. زمان لازم برای یافتن یک جواب دقیق اغلب بیشتر از یک طول عمر است. هیوریستیک‌ها با استفاده از روش‌های نیازمند ارزیابی‌های کمتر و ارائه جوابهایی در محدودیت‌های زمانی قابل قبول دارای نقشی اثربخش در حل چنین مسائل خواهند بود (پیرل1984).

انواع الگوریتم‌های هیوریستیک

درحالت کلی سه دسته از الگوریتم‌های هیوریستیک قابل تشخیص است: الگوریتم‌هایی که برویژگی‌های ساختاری مسئله و ساختار جواب متمرکز می‌شوند و با استفاده ازآنها الگوریتم‌های سازنده یا جستجوی محلی تعریف می‌کنند.

1. الگوریتم‌هایی که بر هدایت هیوریستیک یک الگوریتم سازنده یا جستجوی محلی متمرکز می‌شوند به گونه‌ای که آن الگوریتم بتواند بر شرایط حساس (مانند فرار از بهینه محلی) غلبه کند. به این الگوریتم‌ها، متاهیوریستیک گفته می‌شود.
2. الگوریتم‌هایی که بر ترکیب یک چارچوب یا مفهوم هیوریستیک با گونه‌هایی از برنامه‌ریزی ریاضی (معمولاً روشهای دقیق) متمرکز می‌شوند.

هیوریستیک‌های نوع اول می‌توانند خیلی خوب عمل کنند (گاهی اوقات تا حد بهینگی) اما می‌توانند در جواب‌های دارای کیفیت پایین گیر کنند. همان طور که اشاره شد یکی از مشکلات مهم این الگوریتم‌ها با آن روبرو می‌شوند افتادن در بهینه‌های محلی است بدون اینکه هیچ شانسی برای فرار از آنها داشته باشند. برای بهبود این الگوریتم‌ها از اواسط دهه هفتاد، موج تازه‌ای از رویکردها آغاز گردید. این رویکردها شامل الگوریتم‌هایی است که صریحاً یا به صورت ضمنی تقابل بین ایجاد تنوع جستجو (وقتی علائمی وجود دارد که جستجو به سمت مناطق بد فضای جستجو می‌رود) و تشدید جستجو (با این هدف که بهترین جواب در منطقه مورد بررسی

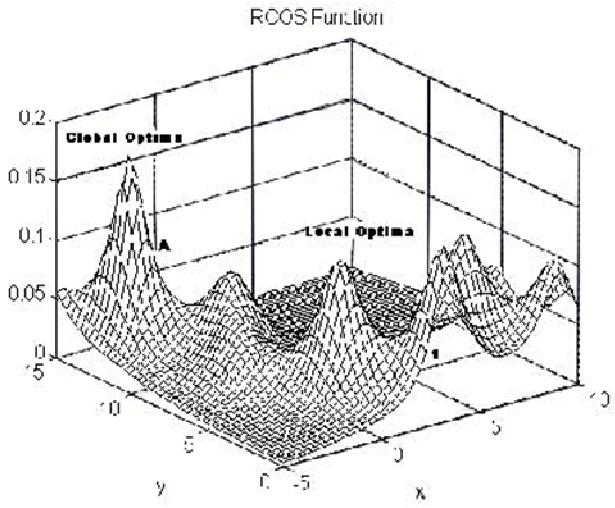
را پیدا کند) را مدیریت می‌کنند. این الگوریتم‌ها متاهیوریستیک نامیده می‌شوند. از بین این الگوریتم‌ها می‌توان به موارد زیر اشاره کرد:

- بازپخت شبیه‌سازی شده
- جستجوی ممنوع
- الگوریتم‌های ژنتیک
- شبکه‌های عصبی مصنوعی
- بهینه‌سازی مورچه‌ای یا الگوریتم‌های مورچگان

معرفی کلی الگوریتم ژنتیک

در بسیاری از مسایل مهندسی و علوم معمولا با تابع هدفی روبرو هستیم که می خواهیم آن را بهینه نماییم. مسایل مهندسی با روشهای متفاوتی مورد تحلیل قرار می گیرند. روش های شیوه تحلیلی نظیر روش مضارب لاگرانژ(LAGRANGE MULTIPLIERS METHOD) حساب تغییرات (Calculus Of Variations) و شیوه های عددی (Numerical Methods) مانند روش های مبتنی بر گرادیان (Gradient based Methods) و روش های تابع جریمه (Penalty Function) را شامل می شوند.در حالت کلی مسایل مهندسی را می توان در چارچوب مسایل برنامه ریزی ریاضی به روش های مبتنی بر گرادیان و روشهای جستجوی مستقیم تقسیم نمود، که در روش اول مشتقات تابع هدف و قید ها به همراه مقادیر این تابع برای یافتن طرح بهینه به کار گرفته می شود. در برخی از مسایل مهندسی استفاده از روش های مبتنی بر گرادیان تابع هدف امکان پذیر است ولی در تعدادی از مسایل یا نمی توان از این روش استفاده کرد و یا به کار گیری آنها به سادگی امکان پذیر نخواهد بود. از دیدگاه دیگر می توان این روش ها را در گروه روش های قطعی (Deterministic) و یا غیر تصادفی، و روش های تصادفی(Stochastic) جای داد. منظور از روش های تصادفی روش هایی است که از نمونه برداری تصادفی در فضای جستجو یا مد لهای تصادفی تابع هدف استفاده می کنند. که در سالهای اخیر توجه زیادی را به خود جلب کرده اند و این به دلیل ارایه روش های موثری در حل مسایل بهینه سازی مشکل و امکان دستیابی به نقاط بهینه کلی می باشد . از طرف دیگرذ بیشتر روش های

غیر تصادفی دارای این اشکال هستند که به محض رسیدن به اولین نقطه بهینه محلی متوقف شده و توانایی خروج از این نقطه و حرکت به سوی نقاط بهینه دیگر و در نهایت نقطه بهینه مطلق را ندارند



الگوریتم ژنتیک

الگوریتم ژنتیک به روش بهینه سازی الهام گرفته از طبیعت جاندار است که می توان در طبقه بندی ها ، از آن به عنوان یک روش عددی ، جستجوی مستقیم و تصادفی معرفی کرد. این الگوریتم ، الگوریتمی مبتنی بر تکرار است و اصول اولیه آن از علم ژنتیک اقتباس گردیده است و با تقلید از تعدادی از فرایندهای مشاهده شده در تکامل طبیعی اختراع شده است و به طور موثری از معرفت قدیمی موجود در یک جمعیت استفاده می کند، تا حل های جدید و بهبود بافته را ایجاد کند. این الگوریتم در مسایل متنوعی نظیر بهینه سازی، شناسایی و کنترل سیستم ، پردازش تصویر و مسایل ترکیبی، تععین توپولوژی و آموزش شبکه های عصبی مصنوعی و سیستم های مبتنی بر تصمیم و قاعده به کار می رود.

چنانکه میدانیم علم ژنتیک،علمی است که درباره چگونگی توارث و انتقال صفحات بیولوژیکی از نسلی به نسل بعد صحبت می کند. عامل اصلی انتقال صفحات بیولوژیکی در موجودات زنده کروموزوم(chromosome)ها و ژن ها(Gene) می باشد و نحوه عملکرد ان ها به گونه ای است که در نهایت ژن ها و کروموزوم ها ی برتر و قوی تر باقی مانده و ژن های ضعیف تر از بین می روند. به عبارت دیگر نتیجه عملیات متقابل ژن ها و کروموزوم ها باقی ماندن موجودات اصلح و برتر می باشد. این الگوریتم برای بهینه سازی، جستجو و یاد گیری ماشین مورد استفاده قرار میگیرد. اساس این الگوریتم قانون تکامل داروین"بقا بهترین" است که میگوید موجودات ضعیف تر از بین میروند و موجودات قوی تر باقی میمانند. در واقع تکامل فرایندی است که روی رشته ها صورت میگیرد،نه روی موجودات زندهای که معرف موجودات رشته است. در واقع، قانون انتخاب طبیعی(Natural Selection) برای بقا می گوید که هر چه امکان تطبیق موجود بیشتر باشد بقای موجود امکان پذیر تر است و احتمال تولید مثل بیشتری، برایش وجود دارد. این قانون بر اساس پیوند بین رشته ها و عملکرد ساختمان های رمز گشایی شده ان ها می باشد.

الگوریتم ژنتیک به دلیل تقلید نمودن از طبیعت دارای چند اختلاف اساسی با روش های جستجوی مرسوم می باشد که در زیر به تعدادی از ان ها اشاره می کنیم.

الگوریتم ژنتیک با رشته های بیتی کار می کند که هر کدام از این رشته ها کل مجموعه متغیر ها را نشان میدهد، حال انکه بیشتر روشها به طور مستقل با متغیر های ویژه بر خورد می کنند.

الگوریتم ژنتیک برای راهنمایی جهت جستجو، انتخاب تصادفی انجام می دهد، که به این ترتیب به اطلاعات مشتق نیاز ندارد.

در الگوریتم ژنتیک، روش های جستجو بر اساس مکانیزم انتخاب و ژنتیک طبیعی ذکر شده در بالا، عمل می نمایند. این الگوریتم ها مناسب ترین رشته ها را از میان اطلاعات تصادفی سازمان دهی شده انتخاب میکنند. در هر نسل، یک گروه جدید رشته ها با استفاده از بهترین قسمت های دنباله های قبلی و بخش جدید اتفاقی برای رسیدن به یک جواب مناسب به وجود می ایند. با وجود اینکه الگوریتم ها تصادفی هستند، ولی در زمره الگوریتم های تصادفی ساده نیستند. ان ها به طور کار امدی به اکتشاف اطلاعات گذشته در فضای جستجو می پردازند تا در یک نقطه جستجوی جدیدی با پاسخ های بهتر، به سمت بهترین جواب پیش روند. هنگام پیش امد سازی(Randomization) الگوریتم های ژنتیک عمل پیش امد سازی ساده را نمی پیمایند بلکه ان ها داده های پیشین را با تفکر انتخاب جستجوی جدید برای رسیدن پیشرفت مورد نظر، توام میکنند.

الگوریتم ژنتیک در هر تکرار چند نقطه از فضای جستجو را در نظر میگیرد بنابر این شانس اینکه به یک ماکزیمم محلی همگرا شود کاهش می یابد. در بیشتر روش های جستجوی مرسوم (روش گرادیان) ، قاعده تصمیم حاکم به این صورت عمل می کند که از یک نقطه به نقطه ی دیگر حرکت می کند. این روش ها متوانند در فضاهای جستجو دارای چند بیشینه ی خطر ناک باشند.

زیرا ممکن است انها به یک مازیمم محلی همگرا شوند. لیکن الگوریتم ژنتیک جمعیت های کاملی از رشته ها(نقاط) را تولید می کند سپس هر نقطه را به صورت انفرادی امتحان می کند و با ترکیب محتویات انها ، یک جمعیت جدید را که شامل نقاط بهبود یافته است ، تشکیل می دهد. صرف نظر از انجام یک جستجو، ملاحظه هم زمان تعدادی نقطه بهبود یافته است، تشکیل می دهد. صرف نظر از انجام یک جستجو، ملاحظه همزمان تعدادی نقطه در الگوریتم ژنتیک ان ها را با ماشین های موازی قابل طبیق می سازد، زیرا در اینجا تکامل هر نقطه یک فرایند مستقل است. لذا الگوریتم ژنتیک فقط نیاز به اطلاعاتی در مورد کیفیت حل های ایجاد شده به وسیله هر مجموعه از متغیر ها دارد.

در صورتی که بعضی از روش های بهینه سازی نیاز به اطلاعات یا حتی نیاز به شناخت کامل از ساختمان مسئله و متغیر ها دارند.

چون الگوریتم ژنتیک نیاز به چنین اطلاعات مشخصی از مسئله ندارد بنابراین قابل انعطاف تر از بیشتر روس های جستجو است. همچنین الگوریتم ژنتیک از روشه های جستجوی نوعی، که برای راهنمایی جهت روش های جستجویشان از انتخاب تصادفی استفاده می کنند، متفاوت است، زیرا اگر چه برای تعریف روش های تصمیم گیری از تصادف و شانس استفاده می کند، ولی در فضای جستجو به صورت تصادفی قدم نمی زند.

مکانیزم الگوریتم ژنتیک

الگوریتم ژنتیک، به عنوان یک الگوریتم محاسباتی بهینه سازی، با در نظر گرفتن مجموعه ای از نقاط فضای جواب در هر تکرار محاسباتی، بنحو مؤثری نواحی مختلف فضای جواب را جستجو می کند. در مکانیزم جستجو گر چه مقدار تابع هدف تمام فضای جواب محاسبه نمی شود ولی مقدار محاسبه شده تابع هدف برای هر نقطه، در متوسط گیری اماری تابع هدف برای هر نقطه، در متوسط گیری اماری تابع هدف در کلیه زیر فضاها یی که ان نقطه بدان ها وابسطه بوده، دخالت داده می شود و این زیر فضاها بطور موازی از نظر تابع هدف متوسط گیری اماری می شوند. این مکانیزم را توازی ضمنی(implicit parallelism) می گویند. این روند باعث می شود که جستجوی فضا به نواحی از ان که متوسط اماری تابع هدف در ان ها زیاد بوده و امکان وجود نقطه بهینه مطلق در ان ها بیشتر است، سوق پیدا کند. چون در این روش بر خلاف روش های تک مسیری، فضای جواب به طور همه جانبه جستجو می شود، امکان کمتری برای همگرایی به یک نقطه بهینه محلی وجود خواهد داشت. امتیز دیگر این الگوریتم آن است که هیچ محدودیتی برای تابع بهینه شونده، مثل مشتق پذیری یا پیوستگی لازم ندارد و در روند جستجو خود تنها به تعیین مقدار تابع هدف در نقاط مختلف نیاز دارد و هیچ اطلاعات کمکی دیگری، مثل مشتق تابع را استفاده نمی کند. لذا می توان در مسائل مختلف اعم از خطی، پیوسته یا گسسته استفاده می شود و به سهولت با مسائل مختلف قابل تطبیق است.

در هر تکرار هر یک از رشته های موجود در جمعیت رشته ها، رمز گشایی شده و مقدار تابع هدف برای ان بدست می اید. بر اساس مقادیر بدست امده تابع هدف در جمعیت رشته ها، به هر رشته یک عدد برازندگی نسبت داده می شود. این عدد برازندگی احتمال انتخاب را برای هر رشته تعیین خواهد کرد. بر اساس این احتمال

انتخاب، مجموعه ای از رشته ها انتخاب شده و با اعمال عملکرد های ژنتیکی روی ان ها رشته های جدید جای گزین رشته هایی از جمعیت اولیه می شوند تا تعداد جمعیت رشته ها در تکرار های محاسباتی مختلف ثابت باشد.

مکانیزم های تصادفی که روی انتخاب و حذف رشته ها عمل می کنند به گونه ای هستند که رشته هایی که عدد برازندگی بیشتری دارند، احتمال بیشتری برای ترکیب و تولید رشته های جدید داشته و در مر حله جایگزینی نسبت به دیگر رشته ها مقاوم تر هستند. بدین لحاظ جمعیت دنباله ها در یک رقابت بر اساس تابع هدف در طی نسل های مختلف، کامل شده و متوسط مقدار تابع هدف در جمعیت رشته ها افزایش می یابد.

بطور کلی در این الگوریتم ضمن انکه در هر تکرار محاسباتی، توسط عملگر های ژنتیکی نقاطی جدیدید از فضای جواب مورد جستجو قرار می گیرند توسط مکانیزم انتخاب، روند جستجو نواحی از فضا را که متوسط اماری تابع هدف در ان ها بیشتر است، کنکاش می کند. بر اساس سیکل اجرایی فوق، در هر تکرار محاسباتی، توسط عملگر های ژنتیکی نقاط جدیدی از فضای جواب مورد جستجو قرار می گیرند توسط مکانیزم انتخاب، روند جستجو نواحی از فضا را که توسط آماری تابع هدف در ان ها بیشتر است، کنکاش می کند.

بر اساس سیکل اجرایی فوق، در هر تکرار محاسباتی، سه عملگر اصلی روی رشته ها عمل می کند. این سه عمل گر عبارتند از دو عملگر ژنتیکی و عملکرد انتخاب تصادفی.

گلد برگ (Gold berg) الگوریتم ژنتیکی هالند را با عنوان الگوریتم ژنتیک ساده(SGA) معرفی می کند. الگوریتم ژنتیک را از الگوریتم ژنتیک طبیعی اقتباس کردند:

بدن همه موجودات زنده از سلول ها تشکیل شده است. و در هر سلولی دسته کروموزوم های یکسانی وجود دارد. کروموزوم ها رشته هایی از DNA هستند که در واقع الگویی برای تمام بدن هستند. هر کروموزومی محتوی دسته هایی DNA است که ژن نامیده میشوند. و هر ژنی پروتئن خاصی را رمز گذاری می کند. اساسا میتوان گفت که هر ژن، ویژگی خاصی (مثلا رنگ چشم) را رمز گذاری می کند. حالت های مختلف یک خصیصه (ابی،قهوه ای)

آلل (Allele)نامیده می شود. هر ژنی موقعیت خاص خود را بر روی کروموزوم دارد. که این موقعیت لوکاس(Locus) نامیده میشود. مجموعه کاملی از مواد ژنتیکی (همه کروموزوم ها) ژنو نامیده میشود. دسته خاصی از ژن های موجود در ژنوم، ژنو تیپ نامیده می شود.ژنوتیپ به همراه تغییرات پس از تولد، پایه و اساس فنو تیپ موجود زنده(ارگانیسم) ، ویژگی های فیزیکی و ذهنی از قبیل رنگ چشم و هوش و غیره است.

در تولید مثل، ابتداترکیب یا تغییر (crossover) اتفاق می افتد. ژن های والدین برای ایجاد کروموزوم های جدید ترکیب می شوند. سپس جنین تشکیل شده دچار تغییر می شود. جهش به این معناست که عناصر DNAکمی تغییر پیدا می کنند. و این تغییرات اغلب نتیجه نسخه برداری غلط از ژن های والدین است. میزان شایستگی (Fitness) موجود زنده (جنین) بواسطه بقای ان اندازه گیری می شود.

در الگوریتم ژنتیک، مجمو عه ای از متغیر های طراحی را توسط رشته هایی با طول ثابت (Fixed length) یا متغیر(variable) کد گذاری می کنند که در سیستم های بیولوژیکی آن ها را کروموزوم یا فرد (Individual) می نامند. هر متغیر طراحی از چند حرف که ژن را الل نامیده اند. هر رشته یک نقطه پاسخ در فضای جستجو را نشان می دهد. به ساختمان رشته ها یعنی مجموعه ای از پارامتر ها که توسط یک کروموزوم خاص نمایش داده می شود ژنو تیپ (genotype) و به مقدار رمز گشایی ان فنو تیپ (phenotype)می گویند. الگوریتم های وراثتی فرایند های تکراری هستند، که هر مرحله تکراری را نسل و مجموعهای از پاسخ ها در هر نسل را جمعیت نامیده اند.

الگوریتم های ژنتیک، جستجوی اصلی را در فضای پاسخ به اجرا می گذارند. این الگوریتم ها با تولید نسل (seeding) اغاز می شوند که وظیفه ایجاد مجموعه نقاط جستجوی اولیه به نام جمعیت اولیه

(Initial Population) را بر عهده دارند و به طور انتخابی یا تصادفی تعیین می شوند. از انجایی که الگوریتم های ژنتیک برای هدایت عملیات جستجو به طرف نقطه بهینه از روش های اماری استفاده می کنند، در فرایندی که به انتخاب طبیعی وابسته است، جمعیت موجود به تناسب برازندگی(Fitness) افراد ان برای نسل بعد انتخاب می شود.

سپس عملگر های ژنتیکی شامل انتخاب(Selection) ،پیوند، جهش و دیگر عملگر های احتمالی اعمال شده و جمعیت جدید به وجود می آید. پس از ان، جمعیت جدیدی جایگزین جمعیت پیشین می شود و این چرخه ادامه می یابد.

معمولا جمعیت جدید برازندگی بیشتری دارد. این بدان معناست که از نسلی به نسل دیگر جمعیت بهبود می ابد. هنگامی جستجو نتیجه بخش خواهد بود که به حد اکثر نسل ممکن رسیده باشیم یا همگرایی حاصل شده باشد و یا معیار های توقف براورده شده باشد.

عملگرهای الگوریتم ژنتیک

به طور خلاصه الگوریتم ژنتیک از عملگرهای زیر تشکیل شده است:

Encoding

این مرحله شاید مشکلترین مرحله حل مساله به روش الگوریتم ژنتیک باشد. الگوریتم ژنتیک به جای این که بر روی پارامترها یا متغیرهای مساله کار کند، با شکل کد شده آنها سروکار دارد. یکی از روش‌های کد کردن ،کدکردن دودویی می باشد که در آن هدف تبدیل جواب مساله به رشته‌ای از اعدادباینری (در مبنای 2) است.

Evaluation

تابع برازندگی را از اعمال تبدیل مناسب بر روی تابع هدف یعنی تابعی که قرار است بهینه شود به دست می آید. این تابع هر رشته را با یک مقدار عددی ارز یابی می کند که کیفیت آن را مشخص می نماید. هر چه کیفیت رشته جواب بالاتر باشد مقدار برازندگی جواب بیشتر است و احتمال مشارکت برای تولید نسل بعدی نیز افزایش خواهد یافت.

Crossover

مهمترین عملگر در الگوریتم ژنتیک، عملگر ترکیب است. ترکیب، فرایندی است که در آن نسل قدیمی کروموزومها با یکدیگر مخلوط و ترکیب میشوند تا نسل تازهای از کروموزومها بوجود بیاید.

جفت هایی که در قسمت انتخاب، به عنوان والد در نظر گرفته شدند، در این قسمت ژنهایشان را با هم مبادله میکنند و اعضایی جدید بوجود میآورند . .ترکیب در الگوریتم ژنتیک باعث از بین رفتن پراکندگی یا تنوع ژنتیکی جمعیت میشود .زیرا اجازه میدهد ژنهای خوب یکدیگر را بیابند.

Mutation(جهش)

جهش نیز عملگر دیگری هست که جوابهای ممکن دیگری را متولد می کند . در الگوریتم ژنتیک بعد از اینکه یک عضو در جمعیت جدید بوجود آمد، هر ژن آن با احتمال جهش، جهش مییابد .در جهش ممکن است ژنی از مجموعه ژن های جمعیت حذف شود یا ژنی که تا حال در جمعیت وجود نداشته است به آن اضافه شود .جهش یک ژن به معنای تغییر آن ژن است و وابسته به نوع کدگذاری، روشهای متفاوت جهش استفاده می شود.

Decoding (رمز گشایی)

عکس عمل encoding است. در این مرحله بعد از اینکه الگوریتم بهترین جواب را برای مساله ارائه کرد لازم است عکس عمل رمزگذاری روی جوابها یا همان عمل decoding اعمال شود تا بتوانیم نسخه واقعی جواب را به وضوح در دست داشته باشیم

چارت الگوریتم به‌همراه شبه‌کد آن

در حالت کلی، وقتی یک الگوریتم ژنتیکی اعمال می‌شود، چرخه زیر را طی می‌کند :

ابتدا یک جمعیت اولیه از افراد به‌طور اتفاقی و بدون در نظر گرفتن معیار خاصی انتخاب می‌شود.برای تمامی کروموزوم های (افراد) نسل صفر، مقدار برازش با توجه به تابع برازش که ممکن است بسیار ساده یا پیچیده باشد تعیین می‌شود . سپس با مکانیزمهای مختلف تعریف شده برای عملگر انتخاب،زیرمجموعه‌ای از جمعیت اولیه انتخاب خواهد شد. سپس روی این افراد انتخاب شده عملیات بُرش وجهش درصورت لزوم با توجه به صورت مساله اعمال خواهد شد.

حال باید این افراد که مکانیزم الگوریتمهای ژنتیک در موردشان اعمال شده است، با افراد جمعیت اولیه (نسل صفر) از لحاظ مقدار برازش مقایسه شوند. (قطعاً توقع داریم که افراد نسل اول با توجه به یکبار اعمال الگوریتمهای ژنتیک روی آنان, از شایستگی بیشتری برخوردار باشند. اما الزاماً چنین نخواهد بود). به‌هر حال افرادی باقی خواهند ماند که بیشترین مقدار برازش را داشته باشند. چنین افرادی در مقام یک مجموعه به عنوان جمعیت اولیه برای مرحله بعدی الگوریتم عمل خواهد کرد.

هر مرحله تکرار الگوریتم یک نسل جدید را ایجاد می‌کند، که با توجه به اصلاحاتی که در آن صورت پذیرفته است، رو به سوی تکامل خواهد داشت.تذکر این نکته خالی از لطف نیست که هرچند الگوریتمهای ژنتیک دارای پایه ریاضی متقن ومشخصی نیستند، اما به‌عنوان یک مدل اجرایی و مطمئن که به‌خوبی نیز پیاده سازی می‌شود،کارآیی خود را نشان داده‌اند.

شبه کد و توضیح آن

در اینجا الگوریتم ژنتیک به‌صورت شبه کد بیان شده است.

```
PSEUDO CODE of  GA

t:=0;                    // start with an initial time
nitpopulation P(t);       // initialize a usually random population of individuals
evaluate P(t);      // evaluate fitness of all initial individuals of population
while (not done) do       // test for termination criterion (time, fitness, etc.)
t:=t+1;                  // increase the time counter
P':=selectparents P(t);  // select a sub-population for offspring production
recombine P' (t);         // recombine the "genes" of selected parents
mutate P' (t);           // perturb the mated population stochastically
evaluate P'(t);            // evaluate it's new fitness
P:=survive P,P'(t);       // select the survivors from actual fitness

end GA.
```

طرح کلی یک الگوریتم به شرح کلی زیر می باشد.

1-{اغاز}: جمعیت n کروموزومی، به صورت تصادفی ایجاد کنید(راه حل های مناسب مساله).

2-{ارزش گذاری (fitness)}: برازندگی f(x) هر کروموزوم X در جمعیت را ارز یابی کنید.

3-{جمعیت جدید}: جمعیت جدیدی را تشکیل دهید. مراحل زیر را تکرار کنید تا جمعیت جدید کامل شود:

{انتخاب}: دو کروموزوم(والدین) را با توجه به برازندگی ان ها از میان جمعیت انتخاب کنید(هر چه برازندگی بهتر باشد، شانس انتخاب بیشتر است).

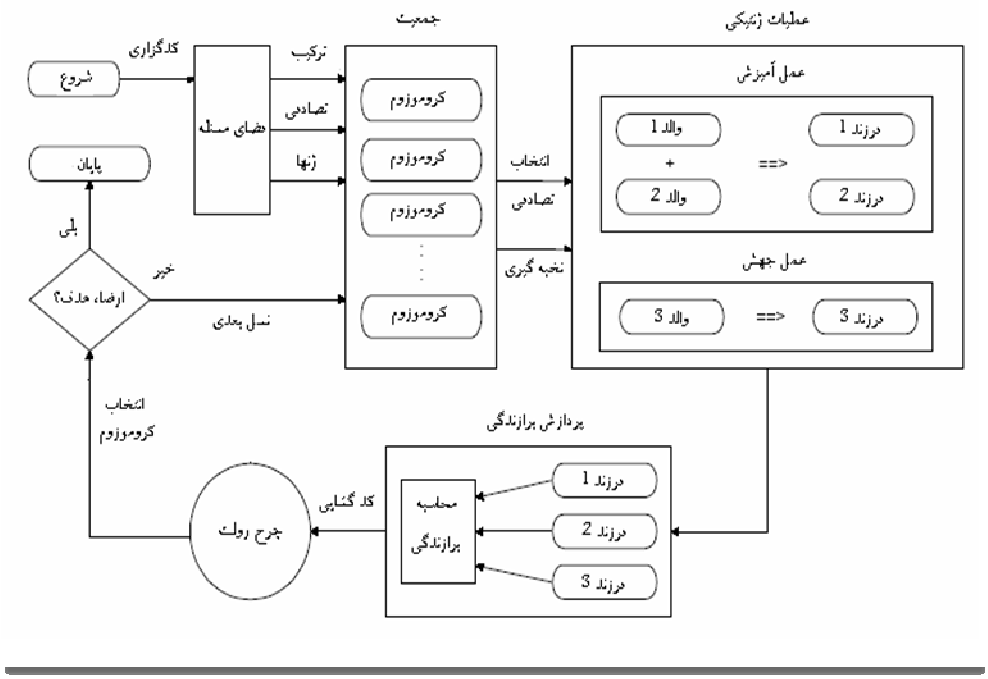
{ترکیب}:با توجه به احتمال ترکیب شدن (pc) والدین را برای تشکیل فرزندان (offspring) جدید با هم ترکیب کنید.

{جهش}: با توجه به احتمال جهش(pm) فرزندان را در هر locus (موقعیت در کروموزوم) مورد جهش قرار دهید.

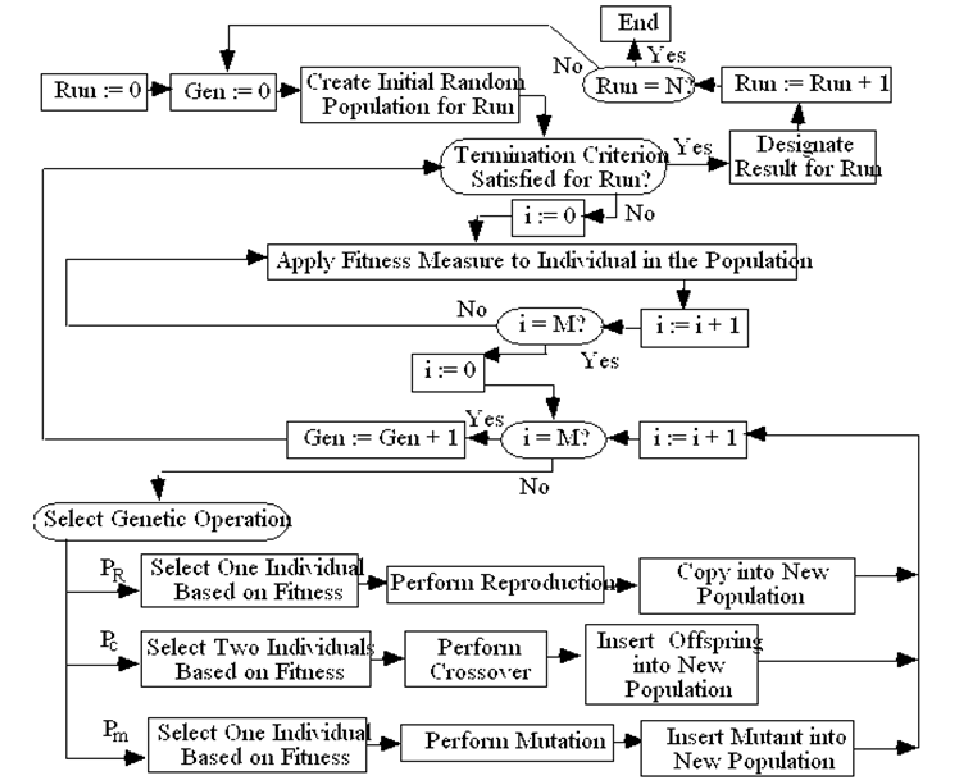
{پذیرفتن}: فرزندان جدید را در جمعیت جدید بگنجانید.

{جایگزینی}: جمعیت جدید ایجاد شده را برای روند الگوریتم بکار ببرید.

چارت الگوریتم



فلوچارت *GA* 1



فلوچارت *GA* 2

کُدکردن

در این قسمت به بررسی کامل انواع کدینگ خواهیم پرداخت.هرچند همان طور که قبلاً هم گفته شده بود معمولا از کد کردن دودویی استفاده میشود ،اما در بسیاری از موارد کدینگ های دیگری بدلیل ماهیت مساله مورد نیازاست.

در این فصل این روشها و استفاده عملگر ها در انان توضیح داده میشود.

Coding :

- Binary coding
- Permutation coding
- Value coding
- Tree coding

الگوریتم ژنتیک به جای این که بر روی پارامترها یا متغیرهای مساله کار کند، با شکل کد شده آنها سروکار دارد. یکی از روش‌های کد کردن،کدکردن دودویی می باشد که در آن هدف تبدیل جواب مساله به رشته‌ای از اعدادباینری (در مبنای 2) است.

تعداد بیت‌هایی که برای کدگذاری متغیرها استفاده می‌شود، به دقت مورد نظر برای جواب‌ها، محدوده تغییر پارامترها و رابطه بین متغیرها وابسته است. رشته یا دنباله‌ای از بیت‌ها که به عنوان شکل کد شده یک جواب از مساله مورد نظر میباشد،کروموزوم نامیده می‌شود. در حقیقت بیت‌های یک کروموزوم،نقش ژنها در طبیعت را بازی می‌کنند. یکی از ویژگی‌های اصلی الگوریتم‌های ژنتیک آن است که متناوباً بر روی فضای کدینگ و فضای جواب کار می‌کنند. اعمال ژنتیک بر روی فضای کدینگ یا کروموزوم‌ها اعمال شده، در حالی که انتخاب و ارزیابی بر روی فضای جواب عمل می‌نماید.

درطبیعت نیز به همین شکل است یعنی افراد (کروموزوم ها) در یک فضای حقیقی غیر کد شده در حالت فنوتیپ حضور دارند. در صورت کدشدن با هر مکانیزمی حالت ژنوتیپ خود را بروزمی‌دهند.

ذکر این نکته ضروری است که هر زمان که از کدینگ صحبت به میان می‌آید بطور پیش فرض منظور کد کردن از نوع باینری می‌باشد(رشته های دودویی). همچنین ساختمان داده مورد استفاده بطور پیش فرض، رشته می‌باشد

کدینگ باینری:

این تبدیل ، تبدیل استاندارد در الگوریتم های ژنتیک می باشد . کدگذاری باینری ساده ترین کد گذاری و بهترین تبدیل برای عملگرهای ژنتیک است اما در مسائل پیچیده این نوع تبدیل چندان مناسب نیست چون معمولاً باعث می شود طول کروموزوم ها برای نگهداری اطلاعات پاسخ ، بسیار بزرگ شود. در تبدیل باینری ، اعضای جمعیت به رشتهایی از 0 ها و 1 ها تبدیل می شوند . به عنوان مثال فرض کنید الگوریتم میخواهد ماکزیمم تابع F(x,y,z) را پیدا کند .در نظر بگیرید جستجو باید در اعداد صحیح مثبت و در محدوده 0 تا 255انجام شود هر پاسخ ممکن شامل سه عدد x و y و z میباشد. طول هر عدد در محدوده موردنظر مساله در تبدیل باینری حداکثر 8 بیت میباشد اگر هر کروموزوم را به صورت xyz در نظر بگیریم بنابراین برای پوشش دادن به تمام پاسخهای ممکن لازم است طول کروموزوم 24= 8 × 3 بیت باشد .برای این مساله کروموزوم I میتواند به شکل زیر باشد.

C = 1101010 11100011 00110111

در همین مساله اگر لازم باشد اعداد منفی نیز جستجو شود میتوان یک بیت به ابتدای هر رشته اضافه کرد که مثلا اگر 0 باشد، عدد، مثبت و اگر 1 باشد عدد، منفی در نظرگرفته شود.

000000001 =1

100000001 = -1

تبدیل اعداد اعشاری نیز میتواند با استفاده از چنین تمهیداتی انجام شود

البته برای این نوع کدینگ دو روش در قسمت نمایش رشته ها به طور کامل بیان شده است.

کدینگ جایگشتی

در بسیاری از مسایل مانند مساله <فروشنده دوره گرد> با جایگشتهای مختلفی از مجموعه راهحلها روبرو هستیم.

در این مساله تعدادی شهر داریم که فاصله میان آنها معلوم است و با شروع از یک شهر و ختم به همان شهر : 1-از تمام شهرها فقط و فقط یکبار عبور نمائیم.

2-کمترین مسافت ممکنه را طی نماییم.

نکتهای که در اینجا مهم است و باعث شده تا کدینگ باینری روش مناسبی برای این مساله نباشد، این نکته است که حتما باید برش میان دو والد بنحوی صورت بگیرد که هیچ عنصر تکراری وجود نداشته باشد.

روش تک نقطه به این شکل اصلاح میشود که تمام قسمت قبل از نقطه برش در والد اول عینا در فرزند کپی میگردد. بقیه ژنهای والد اول که مطمئناً هنوز در فرزند تکرار نشده اند,مطابق با ترتیب قرار گرفتنشان در والد دوم در فرزند کپی می‌شوند.

(1 2 3 4 5 6 7 8 9) + (4 5 3 6 8 9 7 2 1) = (1 2 3 4 5 6 8 9 7)

Single point crossover

البته درحالت جایگشتی میتوان ازPMX هم استفاده نمود.

after crossover before crossover

I H D|B C J |A G E F I H D|E F G |A C B J

H C A|E F G|I B D J H G A|B C J |I E D F

Partially Mixed Crossover

برای جهش نیز از مکانیزم زیر استفاده میشود:

دو موقعیت کاملا دلخواه انتخاب شده و جای آنها با یکدیگر تعویض میشود.

(1 2 3 4 5 6 8 9 7) => (1 8 3 4 5 6 2 9 7)

کدگذاری مقدار

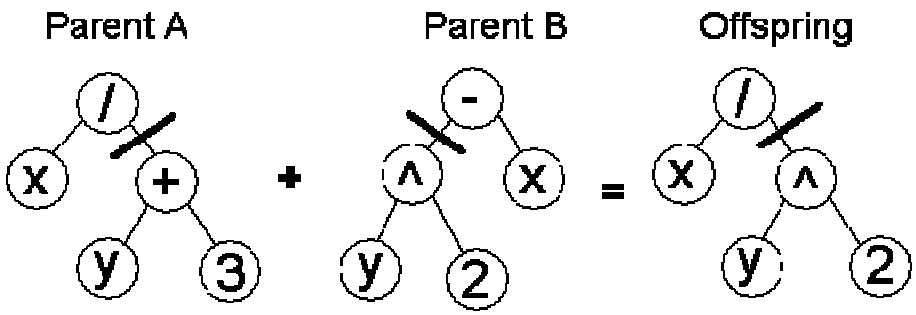
تمامی مکانیزمهای عملگر برش مانند حالت باینری ،استفاده میشود.برای عملگر جهش نیز مانند زیر عمل میشود:

(1.29 5.68 2.86 4.11 5.55) => (1.29 5.68 2.73 4.22 5.55)

یعنی به بعضی ژنها بطور تصادفی عددی اضافه شده یا از آنها کم میگردد

کدینگ درخت

در این نوع کد گذاری یک درخت دودویی برای عبارت (کروموزم) تشکیل می دهیم که معادل درخت پارس است و تمام اعمال مربوط به درخت پارس بر روی درخت قابل انجام است .



برای جهش هم یک نود دلخواه تغییر میکند.

نمایش رشته ها

نمایش مناسب رشته ها به ویژگی های فضای جستجو بستگی دارد ولی معمولا به صورت رشته های دودوئی نمایش داده می شوند. در حل با الگوریتم ژنتیک متغیر ها عموما به صورت دودوئی و با طول رشته ثابت کد گذاری می شوند. برای حل یک مساله بهینه سازی به کمک الگوریتم ژنتیک ابتدا متغیر های مستقل مساله را تشخیص داده میشود و دامنه تغییرات معین می شود.

بنابر اینکه هر متغیر پیوسته یا گسسته باشد یکی از روش های زیر را انتخاب می شود:

الف) متغیر های پیوسته:

با فرض اینکه متغیر مورد نظر از Xmin تا Xmax در حال تغییر باشد و برای نشان دادن ان از یک رشته بیتی استفاده شده باشد.

ب) متغیر های گسسته:

معمولا در مسائل طراحی استفاده از متغیر های گسسته اجتناب ناپذیر است. برای مثال می توان از انتخاب جنس مواد و یا استفاده از جداول استاندارد نام برد. در هر یک از موارد با فرض این که متغیر مورد نظر دارای n مقدار قابل قبول باشد. طول زیر رشته معادل ان از رابطه زیر بدست می اید.

برای مثال اگر متغیری دارای 8 مقدار باشد، برای تمام حالات به یک زیر رشته با سه خانه نیاز است. حال اگر متغیر مورد نظر دارای 10 مقدار متفاوت باشد به یک زیر رشته حد اقل با 4 خانه نیاز است. ولی زیر رشته ای با 4 خانه توانایی 24 مقدار متفاوت را دارد.

برای حل این مشکل می توان به تعداد مورد نظر (در اینجا 14 مورد) از مقادیری که در اختیار است به طور تصادفی انتخاب کرده و به صورت تکراری جایگزین کرد. برای مثال فرض کنید در یک مساله طراحی جنس قطعه مورد بررسی به عنوان متغیر بهینه سازی ا زمیان جنس های آهن، چدن و برنج قابل انتخاب باشد برای بیان این متغیر Li=2 خواهد بود که به صورت زیر کد گذاری می شود:

آهن 00 برنج 01 آهن 11 چدن 10

همان گونه که نشان داده شده است برای فضای چهارم به صورت تصادفی یکی از سه جنس قابل قبول قرار داده شده است پس از اینکه تمام n متغیر طراحی ب ورت زیر دسته هایی در نظر گرفته شدند با کنار هم قرار دادن این زیر رشته ها به یک رشته دودوئی از اعداد به طو ل L_C خواهیم رسید. این رشته از اعداد که معرف یک طرح خواهد بود همان کروموزوم است.

$$L_C = \sum L \qquad (1)$$

حال با داشتن مکان ابتدا و انتهای هر ژن می توان از روی کروموزوم مقدار هر ژن را محاسبه کرد.

هر کروموزوم باید اطلاعاتی درباره راه حلی که نشان می دهد داشته باشد. یکی از روش های رمز گذاری که بیشتر مورد استفاده قرار می گیرد، رشته دو تایی (binary string) است.

نمونه ای از کروموزوم ها در این حالت ، در شکل 2 نشان داده شده است.

Chromosome 1	1101100100110110
Chromosome 2	1101111000011110

شکل 2: نمونه کروموزوم الگوریتم ژنتیکی

هر کروموزوم به صورت رشته ای باینری نشان داده می شود. هر بخش موجود در رشته ، ویژگی هایی از راه حل را نشان می دهد. احتمال دیگر آهن است که تمام رشته نمایان گر یک عدد باشد. البته راه های دیگر بسیاری برای رمز گذاری وجود دارد. رمز گذاری ، عمدتا به مساله حل شده بستگی دارد. بطور مثال می توان مستقیما اعداد صحیح و یا حقیقی را رمز گذاری کرد.

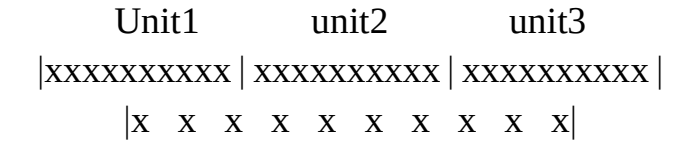
روش معمول برای رمز گشائی متغیر های گسسته به این صورت است که جدولی از تمام حالات زیر رشته مربوط تشکیل داده می شود و هنگام رمز گشائی ، معادل هر مقدار زیر رشته از جدول انتخاب می شود.

انواع روش های تشکیل رشته

برای تشکیل رشته دو روش وجود دارد

الف – روش سری

در روش سری بیت های عددی متناظر هر واحد مطابق شکل (1) کنار هم قرار می گیرند . لازم به توضیح است که تا آخر برنامه هم محل هر واحد ثابت است و هیچ تغیری نمی کند، البته در ابتدای تشکیل رشته هیچ محدودیتی در انتخاب جایگاه وجود ندارد ولی بعد از تشکیل رشته تحت هیچ شرایطی نباید جا به جا شوند.

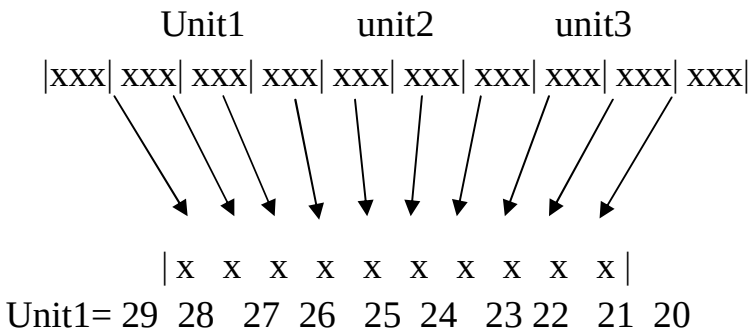


شکل 1 روش سری

ب -روش محاطی

در روش محاطی بیت های عددی م تناظر واحدها مطابق شکل (2) طوری کنار هم قرار می گیرند که در NP (تعداد واحد ها)بیتی در کنار هم قرار می گیرند و هر بیت متعلق به یک واحد می باشد .

به بیان ساده تر اینکه، جایگا ه های بیتی در هر NP بیت به طور متناوب متعلق به یک واحد خاص می باشد.



شکل 2 روش محاطی

باز گرداندن رشته ها به مجموعه متغیرها

در روند اجرای الگوریتم ژنتیک لازم است رشته ها به متغیر ها تبدیل شوند و به عبارت دیگر نمادهای مربوط به هر رشته (نماد ژنی) به دست آیند، تا از آنجا با قرار دادن متغیر ها در تابع هدف، مقدار تابع هدف و در نتیجه ارزیابی آن رشته به دست آید . بنابراین یکی از قسمت های مهم الگوریتم ژنتیک قسمت رمز گشایی است که در جزء ارزیابی انجام می شود.

برای باز گرداندن هر رشته به فضای جستجوی واقعی (فضای متغییرها) باید تعداد بیت مربوط به تک تک متغیرها، نوع متغیر ها (پیوسته یا گسسته) و محل هر متغیر در رشته مشخص باشند.

به هر حال با دانستن اطلاعات مربوط به هر متغیر، زیر رشته متناظر با این متغیر را استخراج و با توجه به محتویات آن مقدار واقعی متغیر را به دست می آوریم . در صورتی که متغیر X پیوسته باشد، این فرایند به صورت زیر انجام می شود.

ابتدا زیر رشته مربوط به این متغیر مش خص می شود .سپس مقدار واقعی آن در مبنای 10 به دست می آید . اکنون برای به دست آوردن مقدار متغیر X باید توجه داشته باشیم که در صورتی که تعداد بیت مربوط به این متغیر برابر n باشد، در واقع ما متغیر x را در یک فاصله $[0,2^n-1]$ نگاشت کرده ایم که عدد به دست آمده در مبنای 10 در واقع عدد مربوط به این فاصله می باشد .حال مقدار متغییر x با توجه به رابطه (4-1) محاسبه می شود.

$$X = \left[\frac{X\max - X\min}{2^N - 1} \right] m + XMIN$$

در صورتی که متغیر مورد نظر نا پیوسته باشد، مقدار اعشاری حاصل از زیر رشته متناظر با متغیر ناپیوسته y در واقع نمایانگر اندیس درایه از یک بردار است که مقدار آن درایه همان مقدار متغیر y می باشد .در واقع در حالت هایی که متغیرها ناپیوسته باشند، بجای رمز نمودن خود متغیرها اندیس های مربوط به آن متغیر ها رمز می شوند و فرایندهای ژنی نیز روی اندی س ها صورت می گیرد نه روی متغیرها.

تعداد بیتهای متناظر با هر متغیر

تعداد بیت های متناظر با هر متغیر در صورتی که از رشته های بیتی استفاده شود، به صورت زیر به دست می آید.

$$2^n = \frac{X\max - X\min}{d}$$

که در رابطه فوق Xmax بیشترین مقدار مجاز متغیر و Xmin کمترین مقدار مجاز متغیر X است و d نیز دقت مورد نظر برای این متغیر می باشد .اکنون با استفاده از رابطه (2-1) مقدار n محاسبه می کنیم .اگر در محاسبه مقدار n اعشاری شود، کوچک ترین عدد صحیح بزرگ تر یا مساوی آن را در نظر می گیریم.

$$n = \log_2 \left[\frac{X\max - X\min}{d} \right]$$

همانطور که گفته شد، تعداد بیت مورد نیاز برای تک تک متغیرهای مسأله به دست می آید . در نهایت در صورتی که K متغیر داشته باشیم طول هر رشته برابر خواهد بود با:

NS=N1+N2+…+NK

جمعیت

مفهوم جمعیت در الگوریتم ژنتیک شبیه به چیزی است که در زندگی طبیعی وجود دارد. برای مساله گزاره هایی وجود دارند که می توانند به عنوان پاسخ، چه درست، چه غلط در نظرگرفته شوند. به این گزاره ها پاسخهای ممکن یا شدنی می گوییم. مثلا اگر مساله یافتن ماکزیمم یک تابع در مجموعه اعداد صحیح باشد، تمام اعداد صحیح می توانند به عنوان پاسخ شدنی مساله در نظر گرفته شوند.

در الگوریتم ژنتیک به عنوان اولین مرحله لازم است مجموعه ای از جوابهای شدنی به عنوان جمعیت اولیه ایجاد شود. اعضای این مجموعه معمولا به صورت تصادفی انتخاب می شوند اما در الگوریتمهای بهینه، از قیدهایی استفاده می شود تا جمعیت پراکندگی بیش از حد نداشته باشد. تعداد اعضای جمعیت به نوع مساله بستگی دارد. در واقع تعداد اعضا، پارامتری است که با تغییر آن می توان دقت جوابها و سرعت همگرایی جستجو را بهبود بخشید. در برخی مسایل یک جمعیت 8 عضوی کاملا مناسب است در حالی که در برخی یک جمعیت 100 عضوی نیز کافی نیست. بر اساس تجربه بهتر است تعداد اعضای جمعیت عددی بین 10 تا 160 باشد.

ایجاد جمعیت اولیه

پس از تعیین سیستم کدینگ و مشخص شدن روش تبدیل هر جواب به کروموزوم، باید یک جمعیت اولیه از

کروموزوم‌ها تولید نمود. در اکثر موارد، جمعیت اولیه به صورت تصادفی تولید می‌شود. اما گاهی اوقات برای بالا بردن سرعت و کیفیت الگوریتم از روش‌های ابتکاری نیز برای تولید جمعیت اولیه استفاده می‌گردد. در هر صورت عمومی‌ترین و راحت‌ترین روش، استفاده از یک رویکرد تصادفی می باشد .

اندازه جمعیت اولیه معمولاً به سایز رشته کدشده وابسته است. به عنوان مثال اگر کروموزوم ها دریک مساله 32 بیتی هستند قطعاً باید جمعیت انتخابی اولیه بیشتر از حالتی باشد که کروموزوم ها به عنوان مثال 16 بیتی هستند.

معمولا احتمال برش بین 80 تا 95 درصد ،احتمال جهش بین نیم تا 1 درصد واندازه جمعیت بین 20تا 30 در نظر گرفته می‌شود.

آنگاه به کروموزوم های انتخاب شده با توجه به یک تابع برازش ، مقداری حقیقی که نشاندهنده ارزش آنها است تخصیص داده می‌شود ومراحل الگوریتمهای ژنتیک ادامه می‌یابد.

اندازه جمعیت

گلد برگ (Gold berg) برای محاسبه بهترین اندازه جمعیت برای کد های دودوئی متغیرهای پیوسته تا طول حداکثر 60 رشته مقدار زیر را پیشنهاد می کند.

$$N_{pop} = 1.65 \times 2^{(0.21*Lc)} \qquad (۶)$$

اگر تعداد کروموزوم ها بسیار کم باشد، GA امکان انجام عمل ترکیب کمتری خواهد داشت و تنها بخشی کوچک از فضای جستجو کشف خواهد شد.از طرفی دیگر ،اگر تعداد کروموزوم ها بسیار زیاد باشد ، روند GA کند خواهد بود. بررسی نشان داده است که در نتیجه برخی محدودیت ها (که عمدتا به کد گذاری و خود مساله بستگی دارد.)استفاده از جمعیت زیاد ، ثمر بخش نخواهد بود. زیرا این کار ، مساله را سریع تر از حالتی که جمعیتی متوسط استفاده شود ، حل نخواهد کرد.

در صورتی که تعداد اعضای جمعیت بسیار زیاد باشد، اگرچه وضعیت جستجو ممکن است به صورت بهتری نمایش داده شود زیرا با افزایش تعداد رشته ها، انتخاب رشته بهتر امکان پذیر می شود .ولی از طرفی نیازمندی های حافظه کامپیوتر و زمان اجرای زیاد می شود .اگر تعداد اعضای جمعیت نیز کوچک تر از حد مشخصی باشد، جمعیت مورد نظر فقط قسمت کوچکی از فضای جستجو را نشان می دهد و ممکن است

جستجو برای رسیدن به حل بهینه در این جمعیت موفقیت آمیز نباشد یا مستلزم صرف زمان زیادی باشد . در عمل تعداد اعضای جم عیت مقداری است که به صورت تجربی به دست آمده و نشان داده شده است . با این تعداد رشته در جمعیت، می توان به حل های مناسبی دست یافت این تعداد 2 الی 2/5 برابر طول هر رشته می باشد.

محاسبه برازندگی(تابع ارزش)

تابع برازندگی را از اعمال تبدیل مناسب بر روی تابع هدف یعنی تابعی که قرار است بهینه شود به دست می آید. این تابع هر رشته را با یک مقدار عددی ارز یابی می کند که کیفیت آن را مشخص می نماید. هر چه کیفیت رشته جواب بالاتر باشد مقدار برازندگی جواب بیشتر است و احتمال مشارکت برای تولید نسل بعدی نیز افزایش خواهد یافت. بسته به اینکه مساله مورد نظر بیشینه یا کمینه سازی باشد برازندگی بیشتر مترادف با بیشینه یا کمینه بودن تابع هدف خواهد بود. از انجائی که الگوریتم ژنتیک طبیعتا به دنبال بیشینه تابع است باید مسائل کمینه سازی به بیشینه سازی تبدیل شود.

ندین روش برای تبدیل تابع هدف به تابع برازندگی وجود دارد. ساده ترین حالت مساوی قرار دادن تابع برازندگی با تابع هدف است. این روش در مسائلی که تابع هدف بایستی بیشینه شود مناسب است. برای تبدیل مسائل کمینه سازی به بیشینه سازی روش های مختلفی وجود دارد. با فرض اینکه مقدار ϕ_i تابع هدف معادل فرد i ام باشد ساده ترین راه،کم کردن ϕ_i از یک مقدار ثابت C است بطوریکه به ازای تمام نسل ها $\phi_i < C$ باشد.

$$f_i = C - \phi_i \quad (۲)$$

در صورتی که نتوانیم بزرگترین مقدار تابع هدف را حدس بزنیم می توانیم در هر نسل ϕ_{max}و ϕ_{min} یافته و برازندگی را به صورت زیر محاسبه می کنیم.

$$f_i = (\phi_{max} + \phi_{min}) - \phi_i \quad (۳)$$

روش دیگر استفاده از تابع نمائی زیر است.

$$f_{i=}e^{-\phi_i} \quad (۴)$$

معمولا مسائل بهینه سازی دارای قید هایی هستند که هنگام حل مساله باید ارضاء شوند در صورتیکه تعداد قید ها کم یا ساده باشند و احتمال ارضاء شدن آن ها به خودی خود زیاد باشد می توان در هر نسل افرادی را که قید ها را ارضاء نمی کنند با افراد جدید به طور تصادفی جایگزین کرد ولی در شرایط پیچیده قید ها به صورت تابع جریمه در تابع هدف منظور می شوند.

$$\phi_{inew} = P_i + \phi_i \qquad (۵)$$

علامت و مقدار P_i باید به گونه ای باشد که موجب شود افرادی از هر نسل که کمتر قید ها را ارضاء می کنند در نهایت از برازندگی کمتری بر خوردار باشند.

انتخاب

در مرحله انتخاب، یک جفت از کروموزوم ها برگزیده می شوند تا با هم ترکیب شوند .عملگر انتخاب رابط بین دو نسل است و بعضی از اعضای نسل کنونی را به نسل آینده منتقل می کند .بعد از انتخاب، عملگرهای ژنتیک روی دو عضو برگزیده اعمال میشوند .معیار در انتخاب اعضاء ارزش تطابق آنها میباشد اما روند انتخاب حالتی تصادفی دارد.

شاید انتخاب مستقیم و ترتیبی به این شکل که بهترین اعضا دو به دو انتخاب شوند در نگاه اول روش مناسبی به نظر برسد اما باید به نکته ای توجه داشت .در الگوریتم ژنتیک ما با ژنها روبرو هستیم .یک عضو با تطابق پایین اگرچه در نسل خودش عضو مناسبی نمی باشد اما ممکن است شامل ژنهایی خوب باشد و اگر شانس انتخاب شدنش 0 باشد، این ژنهای خوب نمی توانند به نسلهای بعد منتقل شوند .پس روش انتخاب باید به گونه ای باشد که به این عضو نیز شانس انتخاب شدن بدهد. راه حل مناسب، طراحی روش انتخاب به گونهای است که احتمال انتخاب شدن اعضای با تطابق بالاتر بیشتر باشد .انتخاب باید به گونه ای صورت بگیرد که تا جایی که ممکن است هر نسل جدید نسبت به نسل قبلی اش تطابق میانگین بهتری داشته باشد. روشهای متداول انتخاب عبارتند از:

- انتخاب چرخ رولت
- انتخاب ترتیبی
- انتخاب بولتزمن

- انتخاب حالت پایدار
- نخبه سالاری
- انتخاب رقابتی
- انتخاب قطع سر
- انتخاب قطعی برینذل
- انتخاب جایگزینی نسلی اصلاح شده
- انتخاب مسابقه
- انتخاب مسابقه تصادفی

انتخاب چرخ رولت

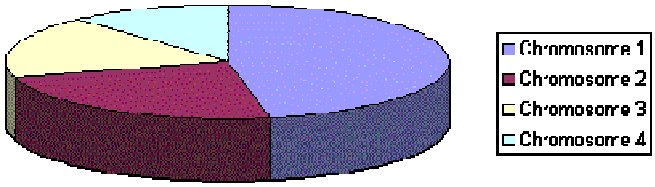
انتخاب چرخ رولت که اولین بار توسط هالند پیشنهاد شد یکی از مناسب‌ترین انتخاب‌های تصادفی بوده که ایدهٔ‌آن، احتمال انتخاب می‌باشد. احتمال انتخاب متناظر با هر کروموزوم، بر اساس برازندگی آن محاسبه شده که اگر f_k مقدار برازندگی کروموزوم k ام باشد، احتمال بقای متناظر با آن کروموزوم عبارت است از:

$$P_K = f_k / \sum_{i=1}^{n} f_i$$

حال کروموزوم‌ها را بر اساس P_K مرتب کرده و q_k که همان مقادیر تجمعی P_K می باشد به صورت زیر به دست می‌آید:

$$q_k = \sum_{i}^{k} P_i$$

چرخ رولت به این صورت عمل می‌کند که برای انتخاب هر کروموزوم یک عدد تصادفی بین یک و صفر تولید کرده و عدد مذکور در هر بازه‌ای که قرار گرفت،کروموزوم متناظر با آن انتخاب می‌شود. البته روش پیاده کردن چرخ رولت به این شکل است که ما یک دایره در نظر گرفته و آن را به تعداد کروموزوم‌ها طوری تقسیم می‌کنیم که هر بخش متناظر با مقدار برازندگی کروموزوم مربوط باشد. حال چرخ را چرخانده و هرکجا که چرخ متوقف شد به شاخص چرخ نگاه کرده،کروموزوم مربوط به آن بخش انتخاب می‌گردد .



انتخاب چرخ رولت ، روشی است که به نسبت مقدار تطابق، اعضا را انتخاب می کند . این روش یک چرخ رولت را شبیه سازی می کند تا تعیین کند کدام اعضا ء شانس بازتولید را دارند . هر عضو به نسبت تطابقش، تعدادی از بخشهای چرخ رولت را به خود اختصاص می دهد .سپس در هر مرحله انتخاب یک عضو و برگزیده می شود و روند آنقدر تکرار می شود تا به اندازه کافی، جفت، برای تشکیل نسل بعد انتخاب گردد . این روش انتخاب را میتوان به صورت زیر بیان کرد:

برداری مانند v در نظر میگیریم:

$$v=[1,...,M]$$

M تعداد عناصر بردار است و اگر تعداد اعضای مجموعه N باشد، هر عضو i 1,...,N دارای تطابقی مانند fi می باشد .هر عضو i به نسبت Pi ، fi بار در v تکرار میشود .هر چه fi بیشتر باشد، عضو مکانهای بیشتری را به خود اختصاص میدهد. عدد از تشکیل بردار v یک مقدار تصادفی $1 \le r \le M$ انتخاب می شود .این مقدار به مکانی در بردار اشاره میکند که آن مکان خود معرف عضوی از اعضای جمعیت است .به عنوان مثال اگر جمعیتی با N=4 داشته باشیم و تطابق اعضا عبارت باشد از:

$$f1= 10 , f2= 10 , f3= 15 , f4= 25$$

مقدار مجموع تطابقها عبارتست از:

$$\sum_{i=1}^{N} f_i = ۶۰$$

بردار v را برداری با 60 عنصر در نظر میگیریم .این بردار به صورت زیر پر می شود . به عضو 1 و 10 مکان ، به عضو 2 و 10 مکان ، به عضو 3 و 15 مکان و به عضو 4 و 25 مکان اختصاص می یابد.

$$V= \{1,1,..., 1,2,2,...,2,3,3,...,3,4,4,...,4\}$$

حالا r بین 1 تا 60 به تصادف انتخاب می شود .فرض کنید r=32 نتیجه می شود :

$$v = [32] = 3$$

پس عضو 3 انتخاب می شود .

انتخاب ترتیبی

در این انتخاب، اعضای جمعیت براساس تطابقشان مرتب می شوند .ارزش منتظره هر عضو به جای تطابقش، در این روش، به رتبه اش بستگی دارد . روش ترتیبی خطی که در سال 1985 توسط بیکر ارائه شده است، به صورت زیر می باشد اعضا در جمعیت طبق تطابق شان به صورت صعودی از 1 تا M مرتب می شوند. M تعداد اعضای جمعیت است . کاربر، ارزش منتظره $0 \leq Max$ را برای عضوی که رتبه M را داراست، درنظر می گیرد .ارزش منتظره هر عضو i در جمعیت در زمان t عبارتست از:

$$ExpVal(i,t)=min(max-min)\ rank(i,t)/M-1$$

که min ارزش منتظره عضو با رتبه 1 است .با اعمال قیدهای $ExpVal(i, t) = Max \quad Max \geq 0$ لازم است تا $1 \leq Max \leq 2$ و $min=2-Max$ باشد . ≤≤

در هر نسل، اعضا مرتب می شوند و طبق رابطه بالا، ارزش منتظره آنها تعیین می شود . مقدار پیشنهادی بیکر برای Max 1و1 است . این مقدار یعنی، به طور میانگین، انتظار می رود بهترین عضو 1,1 بار به عنوان والد انتخاب شود .

انتخاب بولتزمن

در چرخ رولت امکان انتخاب یک عضو به طور مستقیم به مقدار بستگی داشت که برای آن عضو از تابع تطابقش به دست می آمد .مشکلی که انتخاب مستقیم مقدار تطابق به عنوان تنها ملاک انتخاب بوجود می آورد این است که اگر در جمعیتی اختلاف بین مقدارهای تطابق در اعضا زیاد باشد، شانس انتخاب شدن اعضای بد بیشتر و بیشتر به صفر نزدیک می شود .به هم ین دلیل در روش قبل، روش ترتیبی و همچنین روش بولتزمن و برخی روشهای دیگر، از پارامترهای دیگری علاوه بر مقدار تطابق، در انتخاب استفاده میشود .

در روش بولتزمن، به جای مقدار تطابق یک عضو از مقداری به نام Φi برای هر عضو i استفاده می شود که این مقدار از رابطه بولتزمن به صورت زیر محاسبه میشود

$$\varphi_i = \frac{e^{BY_i}}{Z}$$

$$Z = \sum_{i=1}^{N} -BY_i$$

که در رابطه بالا N تعداد اعضای جمعیت و Yi مقدار تطابق عضو i میباشد.

انتخاب حالت پایدار

در اکثر الگوریتمهای GA که در مقالات ارائه شده اند، جمعیت جدید به طور کامل توسط فرزندان بوجود می آید و این فرزندان جایگزین والدین خود میشوند .در بعضی روشها، به برخی از اعضای والد یا اعضای جمعیت قدیمی، اجازه حضور در جمعیت جدید، داده میشود .انتخاب حالت پایدار یکی از این روشها است.

در این روش، فقط تعداد اندکی از اعضای جمعیت کنونی، با اعضای جدید جایگزین می شوند .به عبارت دیگر، بدترین اعضا با فرزندانی که از بهترین اعضا بوجود آمده اند تعویض می شوند اما بافت کلی جمعیت، چندان تغییر نمیکند.

نخبه سالاری

ایده نخبه سالاری، ویژگی تازهای به پروسه انتخاب اضافه میکند .در نخبه سالاری، بهترین عضو هر جمعیت، زنده می ماند و در جمعیت بعد حضور دارد .به عبارت دیگر، عضوی که بالاترین تطابق را دارد به طور خودکار به جمعیت جدید منتقل میشود .این روش ابتدا در سال 1975 توسط کند دی جونز معرفی شد .اعمال نخبه سالاری در الگوریتم ژنتیک، معمولا باعث بهبود کارایی آن می شود.

انتخاب رقابتی

این روش تعدادی از اعضای جمعیت را به تصادف انتخاب می کند و سپس اگر شرطی خاص برقرار باشد، بهترین یا تعدادی از بهترینهای آنها را به عنوان والد برمی گزینند .اگر شرط برقرار نشود، بدترین عضو یا تعدادی از بدترین ها، در تشکیل جمعیت آینده به عنوان والد در نظر گرفته می شوند .

شکل استاندارد این روش، رقابت دوتایی یا باینری است و به شکل زیر میباشد:

1. 2 عضو به تصادف انتخاب می شوند.
2. مقدار r بین 0 و 1 به تصادف تعیین می شود.
3. پارامتر $0 \le K \le 1$ توسط کاربر تعیین می شود (مثلا k = 0 .75).
4. اگر $r < k$ عضو برتر و اگر $r \ge k$ عضو بدتر بین این دو عضو، به عنوان والد انتخاب می شود.
5. دو عضو انتخاب شده برای رقابت، به جمعیت برمی گردند و می توانند دوباره در رقابت شرکت کنند.

روش انتخاب رقابتی میتواند به صورت رقابت n تایی نیز انجام شود.

انتخاب قطع سر

در این روش که توسط گلدبرگ معرفی و ارائه شده، ابتدا یک عدد T که کوچک‌تر از صد می باشد، تعریف شده سپس

کروموزوم‌ها را بر مبنای مقادیر برازندگی مرتب کرده و T درصد،برتر از انتخاب می‌کنیم. حال از هر یک از آنها $\frac{100}{T}$ کپی به نسل بعد انتقال می‌دهیم مثلاً فرض کنید T=20 و تعداد کروموزوم‌ها نیز 20 عدد باشد. بنابراین ٪20 کروموزوم‌های اول یعنی 100/5=4 تای اول را از لیست مرتب شده انتخاب کرده و از هر کدام 5 کپی در نظر بگیرید.

انتخاب قطعی بریندل

این روش که توسط بریندل معرفی و ارائه شده،به این صورت است که احتمال انتخاب برای هر کروموزوم طبق محاسبه می‌شود $\left(P_K = \frac{f_k}{\sum f_i}\right)$ و تعداد مورد انتظار برای هر کروموزوم نیز به صورت $e_k = P_k \times pop - size$ تعریف شده است.

حال هر کروموزوم بر طبق قسمت صحیح مقدار مورد انتظار، به نمونه تخصیص داده می‌شود و سپس جمعیت بر حسب قسمت اعشار تعداد مورد انتظار، مرتب شده و به تعداد مورد نیاز جهت تکمیل جمعیت، به ترتیب از بالای لیست برداشته می‌شود.

انتخاب جایگزینی نسلی اصلاح شده

در این روش که توسط وایت‌لی ارائه شده، ابتدا کروموزوم‌ها بر اساس مقدار برازش منظم شده، سپس به تعداد نوزادان تولید شده از انتهای لیست کروموزوم‌ها حذف می‌گردند، آنگاه نوزادان جایگزین کروموزوم‌های حذف شده می‌شوند. مثلاً اگر 10 کروموزوم وجود داشته باشد، ابتدا آنها را مرتب کرده و بعد اگر قرار باشد 4 نوزاد نیز تولید شود پس از تولید نوزادان ، آنها جایگزین 4 کروموزوم آخر لیست می‌شوند.

انتخاب مسابقه

دراین روش که توسط گلدبرگ ارائه شده، به تعداد Pop-Size مجموعه شامل چند عضو که از قبل مشخص شده، تولید کرده و در هر مجموعه بهترین عضو انتخاب می‌گردد. اندازهٔ مجموعهٔ فوق که به آن اندازه مسابقه گفته می‌شود معمولاً برابر 2 فرض می‌گردد

انتخاب مسابقه تصادفی

این روش که توسط وزل ارائه شده، مانند حالت قبل بوده، با این تفاوت که به جای این که مجموعه به صورت تصادفی انتخاب شود با کمک چرخ رولت انتخاب شده و بهترین آن به عنوان یک عضو از نسل جدید در نظر گرفته می‌شود

ترکیب

در طبیعت بقای نسل یکی از مهمترین فاکتورهاست و تنها عملگر ممکن برای این امر آمیزش است.

در الگوریتمهای ژنتیکی به طبع طبیعت آمیزش وجود دارد.آمیزش با تعویض ژنها، بین دو کروموزوم انجام میگیرد و هر کدام از کروموزومها خصوصیاتی از خود را به فرزندان انتقال میدهند. بدیهی است کروموزومهایی که دارای برازندگی بیشتری هستند شانس بیشتری برای آمیزش دارند.

مهمترین عملگر در الگوریتم ژنتیک، عملگر ترکیب است. ترکیب، فرایندی است که در آن نسل قدیمی کروموزومها با یکدیگر مخلوط و ترکیب میشوند تا نسل تازهای از کروموزومها بوجود بیاید.

جفتهایی که در قسمت انتخاب، به عنوان والد در نظر گرفته شدند، در این قسمت ژنهایشان را با هم مبادله میکنند و اعضایی جدید بوجود میآورند.

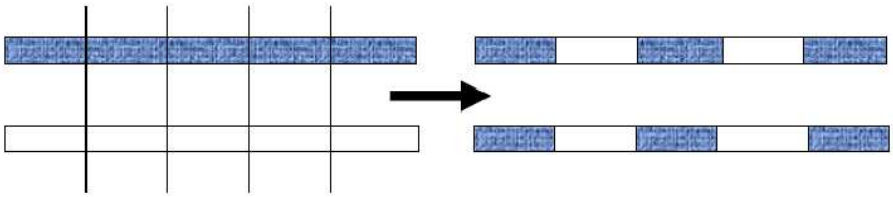
بر اساس مباحث تئوری (که در ادامه بررسی خواهد شد) ترکیب اعضا با تطابق بالا باعث بوجود آمدن اعضایی میشود که از تطابق میانگین، تطابق بیشتری دارند .ترکیب در الگوریتم ژنتیک باعث از بین رفتن پراکندگی یا تنوع ژنتیکی جمعیت میشود .زیرا اجازه میدهد ژنهای خوب یکدیگر را بیابند.

جا به جائی دودوئی (Binary Crossover)

روش های معمول جا به جائی تک نقطه، دو نقطه،چند نقطه و جابجایی یکنواخت می باشد. ساده ترین حالت جا به جا کردن، جا به جائی تک نقطه ای (Single point) است. در جا به جایی تک نقطه ای، ابتدا جفت کروموزوم والد(رشته دودوئی) در نقطه مناسبی در طول رشته بریده شده و سپس قسمت های از نقطه برش، با هم عوض می شوند. بدین ترتیب دو کروموزوم جدید بدست می آید که هر نقطه از ان ها ژنهایی را از کروموزوم های والد به ارث میبرند.

برای جا به جایی چند نقطه ای m موقعیت جا به جا شدن, $k_i \in \{1,2,...,l-1\}$,که k_i نقطه جا به جائی و l طول کروموزوم می باشد، را به صورت تصادفی و بدون تکرار انتخاب می کنیم. سپس جهت ایجاد فرزندی جدید بیت های بین نقاط مشخص شده در والدین با هم عوض می شوند.

به هر بیت از رشته کروموزوم ها یک آلل(Allele) گفته می شود. بخش بین اولین آلل تب اولین نقطه قطع، بین والدین جا به جا نمی شود. این عملیات در شکل 6 نشان داده شده است. فلسفه انجام جا به جایی در چند نقطه و البته حالت های مختلف دیگر عملگر جا به جائی این است که، قسمت هایی از کروموزوم که بیان کننده سهم بسزایی در عملکرد بهتر یک عضو خاص هستند ممکن است در زیر رشته های همسایه یافت نشوند.



شکل 6- جا به جایی چند نقطه

به نظر می رسد نحوه عملکرد عملگر جا به جایی در چند نقطه (Multi crossover) نسبت به روش همگرائی به مقادیر بالاتر برازندگی به پیشرفت و توسعه جست و جو در فضای داده های مربوطه بیشتر کمک می کند، لذا جستجو در دامنه جواب قوی تر می شود.

یانگ عملکرد جا به جایی چند نقطه را مورد بررسی قرار داده و ثابت کرد که عملگر جا به جائی بیشتر، عملکرد الگوریتم ژنتیک را کاهش می دهد.

عملگرهای جا به جائی یک نقطه ای و چند نقطه ای در جائی اثر می کنند که کروموزوم دقیقا در ان نقاط فقط می تواند جدا شود، اما عملگر جا به جائی یکنواخت پتانسیل جا به جا شوندگی را به تمام نقاط یک کروموزوم به صورت یک نواخت نسبت می دهد. به این معنی که احتمال جا به دجا شدن کروموزوم در هر نقطه برابر خواهد بود. یک الگوی بیان کننده عمل جا به جائی (به همان طولی که کروموزوم ها دارند) به صورت تصادفی ایجاد می شود و مقدار تعیین شده در هر بیت از این نمونه نشان می دهد که کدام یک از والدین به عنوان مرجع مقدار دهی برای آن بیت از فرزند خواهد بود. تعداد نقاط برش ثابت نیست ولی به طور متوسط برابر $\frac{l}{2}$ است.

دو کروموزوم اولیه (والدین) و الگوی عملگر جا به جائی و فرزندان حاصل را در نظر بگیرید.

$$P_1 = 1011000111$$
$$P_2 = 0001111000$$
$$\text{Mask} = 0011001100$$
$$O_1 = 0011110100$$
$$O_2 = 1001001011$$

در اینجا مشاهده میشود که فرزند اول O_1 بدین صورت ایجاد شده است که اگر بیت مربوط در الگوی جا به جایی (mask) 1 باشد مقدار آن بیت در O_1 برابر با مقدار بیت متناظر در P_1و همچنین اگر بیت مربوط در الگوی جا به جائی 0 باشد مقدار ان بیت در O_1 برابر با مقدار بیت متناظر درP_2 است.

رشته O_2 با جا به جا کردن P_1 و P_2و در نظر گرفتن همین شیوه ایجاد شده است یعنی مقدار 1 در رشته mask به معنی مقدار بیت متناظر در P_2 برای همان بیت در O_2 است.

مشخصا عملگر جا به جائی یکنواخت همانند عملگر جا به جائی چند نقطه ا ی باعث کاهش خطای همگرائی،ناشی از طول باینری استفاده شده و نوع کد کردن سری پارامترهای داده شده،می شود. این مساله کمک می کند که بر خطای همگرائی موجود در حالت جا به جائی تک نقطه ای در زیر رشته های کوتاه غابه کنیم بدون انکه نیاز به دانستن مقادیر بیت های اعضا در کروموزوم های ارائه شده داشته باشیم.

Spears و Dejong داده اند که چگونه جا به جایی یکنواخت به وسیله احتمال عوض شدن و جا بجا شدن بیتها پارامتری می شود. این پارامتر فوق العاده می تواند بدون توصیف یک همگرائی مربوط به طول رشته های

استفاده شده، در کنترل مقدار تغییر یافته در طول ترکیب بندی مجدد استفاده شود، هنگامی که از عملگر جا بجائی یکنواخت در مقادیر حقیقی استفاده شود به آن ترکیب بندی منفصل گفته می شود.

در مقایسه هائی که بین عملگرهای دودوئی به هر دو صورت تئوری و تجربی انجام شده و نتایج بدست امده نشان می دهد که،هیچ یک از این عملگرها نمی تواند به طور مطلق بهترین بوده و اختلاف در سرعت این روش ها هم نمی تواند بیش از 2٪ باشد.

عملگرجا بجائی دیگری که مطرح می شود به اسم shuffle است. یک نقطه قطع مجزا انتخاب می شود اما قبل از اینکه بیت ها تعویض شوند، در هر دو والد بیتها به صورت تصادفی جا به جا می شوند. بعد از ترکیب بندی مجدد بیت ها در رشته فرزند جایگذاری می شوند. این عملگر نیز خطای همگرائی رشته ها را با جا به جائی تصادفی بیتها در هر جائی که عملگر جا به جائی انجام می شود حذف می کند.

عملگر دیگری نیز عمل جا به جائی را مقید می کند که همیشه اعضای جدیدی ایجاد کند در هر جائی که ممکن باشد. معمولا این عملگر بدین صورت عمل می کند که مکان نقاط قطع را محدود می کند به گونه ای که نقاط قطع تنها جائی اتفاق می اقتند که مقادیر ژن در دو کروموزوم متفاوت است.

جا بجائی حقیقی (real crossover)

در کد کذاری حقیقی که کروموزوم ها به صورت برداری از اعداد حقیقی می باشند. روش های زیادی برای عملگر جا بجائی حقیقی ارائه شده که اکثر ان ها در دو دسته زیر خلاصه می شود:

1-جا بجائی عمومی

2- جا به جائی محاسباتی

عملگر های جا بجائی عمومی با توسعه روش های جا بجائی دودوئی برای کد گذاری حقیقی تهیه می شود که مثال ساده از ان، عملگر جا به جائی ساده می باشد که شامل جا به جائی تک نقطه، دو نقطه و چند نقطه است که مشابه همان حالت دودوئی می باشند با این تفاوت که د راین جا به جای یک بیت دودوئی (0و1) ، یک عدد حقیقی در رشته است. با فرض اینکه $C_1=(C_1^1,...,C_n^1)$ و $C_2=(C_1^2,...,C_n^2)$ دو کروموزومی می

باشند که تحت عمل جا بجائی قرار می گیرند و نقطه $i \in \{1,2,...,n-1\}$ نقطه جا بجائی باشد دو کروموزوم جدید که از اعمال عملگر جابجائی ساده حاصل می شود و به صورت روابط زیر خواهد بود:

$$H1 = \left(C_1^1, C_2^1, .., C_i^1, C_{i+1}^2, .., C_n^2\right)$$

$$H2 = \left(C_1^2, C_2^2, .., C_i^2, C_{i+1}^1, .., C_n^1\right)$$

$Hk = \left(h_1^k, .., h_i^k, .., h_n^k\right)$ جا به جائی محاسباتی بر اساس مفهوم ترکیب خطی بردار ها تهیه شده است. با این فرض که دو فرزند که k = 1, 2 است بر اساس این عملگر تولید شده باشند, در این صورت تحت شرایط مختلف برای λ_1 و λ_2 ضرایب در روابط مذکور، سه نوع مختلف از این عملگر ایجاد می شود:

1- near Crossover که در ان λ_1 و λ_2 هر دو حقیقی هستند.

2- Affine Crossover که در ان $\lambda_1 + \lambda_2 = 1$ است.

3- Convex Crossover که در ان $\lambda_1 + \lambda_2 = 1$ بوده و λ_1 و λ_2 هر دو حقیقی مثبت هستند.

اسامی linear , Affine , Convex از تعوری مجموعه های محدب وام گرفته شده است.

عملگر convex معمولا بیشتر از بقیه کاربرد دارد. این عملگر ها به طور شماتیکی در شکل 9 نشان داده شده است.

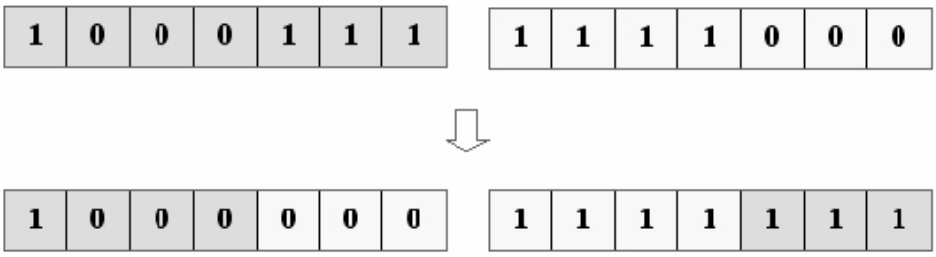
$$h_i^1 = \lambda_1 C_i^2 + \lambda_2 C_i^1 \qquad h_1^1 = \lambda_1 C_i^1 + \lambda_2 C_i^2$$

روشهای پیاده سازی عملگر ترکیب

ترکیب تک نقطه ای

ترکیب تکنقطهای، دو کروموزوم را با انتخاب تصادفی موقعیتی مانندP ، ترکیب میکند P مقداری کمتر یا مساوی طول کروموزومها است .اگر تعداد) طول (ژنها در کروموزومها N ، باشد، از دو کروموزوم والد، دو فرزند به صورت زیر بوجود میآید.

یک فرزند با کپی کردن ژن های (P-1)...1 از کروموزوم والد اول و ژن های P...N از کروموزوم والد دوم، ساخته میشود و فرزند دیگر به طور مشابه، این بار با کپی کردن ژن های (P-1)...1 از والد دوم و ژن های P...N از والد اول، بوجود میآید .در این نوع ترکیب از دو والد، دو فرزند بوجود می آید .به عنوان مثال، این نوع ترکیب در شکل ....نشان داده شده است .در این مثال P=4 میباشد.



شکل 3-3- ترکیب تک نقطهای

لازم به ذکر است اگر P برابر 1 شود یا برابر طول کروموزومها، آنگاه دو والد بدون تغییر وارد جمعیت بعدی میشوند.

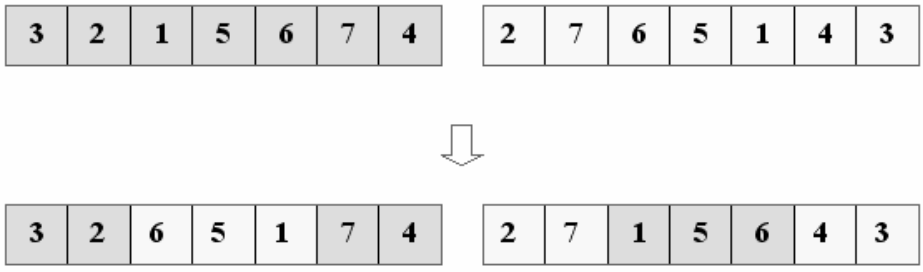
ترکیب دو نقطه ای

در ترکیب دونقطهای، دو موقعیت p1 و p2 به عنوان موقعیتهای ترکیب، به طور تصادفی بین 1 و طول کروموزومها (N) انتخاب میشود .روش ایجاد فرزندان مانند ترکیب تک نقطهای است.

فرزند اول، ژنهای $(P_1-1)...1$ را از والد اول، ژن های $P_1...(P_2-1)$ را از والد دوم و ژن های p2...N را مجددا از والد اول، به ارث میبرد.

فرزند دوم، ژنهای $(P_1-1)...1$ را از والد دوم، ژن های $P_1...(P_2-1)$ را از والد اول و ژن های $P_{2...N}$ را مجددا از والد دوم، بدست میآورد.

در این روش ترکیب نیز، از یک جفت، دو فرزند بوجود میآید .در این روش احتمال اینکه والدها بدون تغییر به جمعیت بعد منتقل شوند، کمتر است .در شکل ....نمونه های از این ترکیب با موقعیتهای ترکیب $P_1=2$, $P_2=5$ نشان داده شده است.



شکل 4-3- ترکیب دونقطه ای

ترکیب n نقطهای

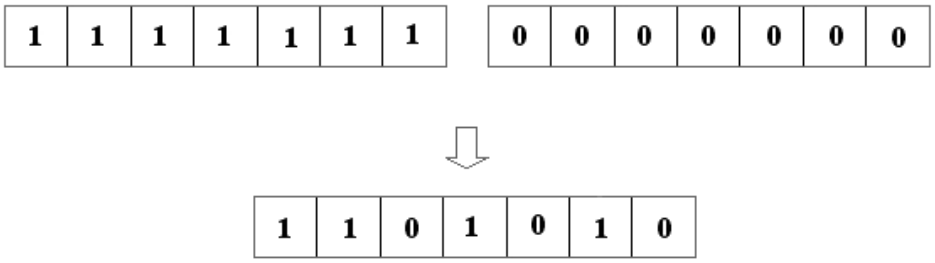
با انتخاب n موقعیت ترکیب، و چیدن ژنها مشابه آنچه در ترکیب تک نقطهای و دو نقطهای گفته شد، ترکیب n نقطهای خواهیم داشت.

ترکیب یکنواخت

در ترکیب یکنواخت، هر ژن کروموزوم جدید به صورت جداگانه انتخاب میشود .هر ژن وابسته به موقعیتش به صورت تصادفی از یکی از دو والد انتخاب میشود .مثلا ژن اول از والد دوم، ژن دوم از والد دوم، ژن سوم از والد اول و تا ژن آخر .برخلاف ترکیبهایی که قبلا ذکر شد، این نوع ترکیب، یک فرزند بوجود می آورد. در واقع در این حالت از یک ماسک استفاده میشود

جمعیت جدیدی که با ترکیب یکنواخت بوجود میآید، دارای تنوع ژنتیکی بیشتری نسبت به ترکیبهای تک نقطه ای و دو نقطه ای میباشد .به همین دلیل این نوع ترکیب در جمعیتهایی که اعضایی کمی دارند اثر بهتری دارد تا جمعیتهایی که تعداد اعضای زیادی دارند.

در جمعیتهای کوچک، ممکن است به تنوع ژنتیکی نیاز باشد تا روش، سریعتر همگرا شود .اما در جمعیتهای بزرگ، معمولا تنوع ژنتیکی لازم، فراهم است .در شکل ....نمونه ای از ترکیب یکنواخت مشاهده می شود



شکل 4-3- ترکیب یکنواخت

ترکیب حسابی

ترکیب حسابی به صورت زیر تعریف میشود:

اگر A و B دو عضو از جمعیت فعلی باشند که به عنوان والد انتخاب شدهاند، از آنها دو فرزند a و b به صورت زیر بوجود میآید:

$$a = \delta A + (1 - \delta) B$$

$$b = \delta B + (1 - \delta) A$$

پارامتر δ مقداری در بازه [0,1] میباشد که در هر ترکیب میتواند مقدار مختلفی داشته باشد.

ترتیب

این روش توسط دیویس معرفی گردیده و به روش O X معروف می‌باشد در این روش دو عدد را به صورت تصادفی به عنوان نقاط برش به دست آورده سپس قسمت مابین را در دو طرف کروموزوم ثابت نگه داشته ولی قسمت‌های دو طرف به این صورت به دست می‌آید که برای نوزاد اول در والد دوم از ابتدای کروموزوم شورع کرده و آنهایی را که در قسمت مابین نوزاد وجود ندارد در جای خالی قرار می‌گیرند

با مثال زیر بحث روشن می‌شود:

والد اول: 476/3598/12

والد دوم: 835/7641/29

برای نوزاد اول قسمت مابین والد یک، بدون تغییر جابجا می‌شود.

نوزاد اول: ---/3598/--

حال قسمت خالی از روی والد دوم پر می‌گردد.

نوزاد اول: 746/3598/12

برای نوزاد دوم نیز به همان صورت عمل می‌شود:

نوزاد دوم: 359/7641/82

چرخه

این روش توسط الیور، اسمیت و هالند معرفی شده و به نام عملگر CX معروف می باشد و به این گونه عمل می‌شود که ابتدا اولین ژن را عیناً از والد اول به نوزاد اول کپی کرده، سپس یک چرخه بین ژن‌هایی که در دو کروموزوم والد اول و دوم وجود دارد ایجاد می‌گردد، نقطه شروع همان ژن فوق می‌باشد. برای ایجاد چرخه باید ابتدا همان ژن را در کروموزوم دوم یافته و مکان آن را در نظر بگیریم. سپس ژن موجود در همان مکان از کروموزوم اول تثبیت کنیم. این عمل تا جایی که چرخه کامل شود ادامه می‌یابد (تا جایی که به نقطه شروع برسد). در این لحظه برای تکمیل نوزاد اول باید ژنهای باقیمانده از کروموزوم دوم را به همان ترتیب نوزاد اول جای‌گذاری نمود.

که مثال زیر بحث را روشن می‌کند.

والد اول: 346578291

والد دوم: 794356281

برای تولید نوزاد اول، ابتدا اولین ژن از کروموزوم والد اول را منتقل کرده و عمل ادامه می‌یابد:

نوزاد اول: 3---57----

حال که چرخه کامل شد عددهای به دست آمده را در والد دوم حذف کرده باقیمانده به ترتیب عبارت است از 946281 که به همین حالت در جاهای خالی نوزاد اول قرار می‌گیرد:

نوزاد اول: 394576281

برای نواد دوم نیز به همان صورت عمل می‌شود.

نوزاد دوم: 7—35---

که با ادامه روند فوق رشته زیر حاصل می‌شود.

نوازد دوم: 746358291

محدب

در این عملگر اگر والد اول P_1 و والد دوم P_2 شود نوزاد اول و دوم به صورت زیر حاصل می‌شود:

نوزاد اول: $c_1 = \lambda_1 p_1 + \lambda_2 p_2$

نوزاد دوم: $c_2 = \lambda_1 p_2 + \lambda_2 p_2$

اگر $\lambda_1 = \lambda_2 = 0.5$ به آن عملگر تقاطعی متوسط می گویند. اگر $\lambda_2 = -0.5$ و $\lambda_1 = 1.5$ باشد به آن نسبت سلبی گفته و در صورتی که λ_2 و λ_1 به صورت تصادفی از بازه $[-d, 1+d]$ انتخاب شود به آن تقاطعی میانه توسعه یافته می گویند

تقاطعی متوسط توسط دیویس و تقاطعی میانه توسعه یافته توسط مولن بین و نسبت سلبی توسط رایت ارائه شده‌اند البته چنگ وجن حالتی را که λ_1, λ_2 دو عدد تصادفی بوده و دارای شرط $\lambda_1 > 1$ و $\lambda_2 > 0$ و $\lambda_1 + \lambda_2 \leq 2$ باشد را تحت عنوان عملگر خطی ارائه نمودند

بخش-نگاشته

این روش که توسط گلدبرگ و لینگل معرفی شده به روش PMX معروف بوده و در حقیقهٔ همان عملگر دو نقطه برش می‌باشد که برای حالت خاصی ارائه شده است

در این روش دو عدد به صورت تصادفی به عنوان نقاط برش به دست آورده، سپس قسمت مابین دو نقطه برش را در دو کروموزوم تعویض کرد و آنگاه قسمت‌های دو طرف طوری مقدارگذاری می‌شوند که در هیچ کدام از دو کروموزوم، تکراری صورت نگیرد.

روش کار با یک مثال روش شده است.

والد اول: 43/5628/791

والد دوم: 65/8349/217

حال برای تولید نوزاد به این صورت عمل می‌شود که قسمت مابین را عوض کرده بعد در والد اول از ابتدای کروموزوم شروع نموده هر عددی را که در قسمت مابین کروموزوم جدید نباشد عیناً نوشته و برای تکراری‌های جای خالی قرار داده می‌شود سپس در والد دوم از ابتدای کروموزوم شروع کرده و هر عددی که در نوزاد جدید نباشد به جای محل خالی، گذاشته می‌شود تا کلیه جاهای خالی پر گردد. برای نوزاد دوم نیز به همین

صورت عمل می‌شود. بنابراین ابتدا قسمت‌های میانی را جابجا کرده دو کروموزوم زیر حاصل می‌شود.

نوزاد اول: --/8349/---

نوزاد دوم: ---/5628/--

سپس جاهای خالی نوزاد اول در صورت تکرای نبودن ژن مورد نظر پر می‌شود.

نوزاد اول: 1-7/8349/--

حال جاهای خالی باقیمانده نوزاد اول پر می‌شود.

نوزاد اول: 65/8349 /721

با توجه به مطالب گفته شده، نوزاد دوم به دست زیر به دست می آید.

نوازد اول: 17-/5628/--

نوزاد دوم: 43/5628/917

احتمال ترکیب

ترکیب لازم نیست در هر نسل اتفاق بیفتد .در واقع ممکن است نسلهایی بدون عملگر ترکیب به نسل های جدید، تبدیل شوند .برای تعیین رخ دادن یا ندادن ترکیب از پارامتری به نام احتمال ترکیبPc استفاده میشود که این پارامتر مقدار بین 0 و 1 است .از آنجا که ترکیب نقشی اساسی در رشد مقدار میانگین تطابق جمعیت دارد، مقدار Pc بین 0/5 تا 0/8 و بیشتر بین 0/7 تا 0/8 درنظر گرفته میشود.

اگر ترکیبی صورت نگیرد، فرزندان دقیقا همانند والدین خواهند بود. اگر ترکیب صورت بگیرد، فرزندان از بخش هایی از کروموزوم های والدین به وجود می ایند. اگر احتمال ترکیب 100٪ باشد ، در این صورت همه فرزندان در نتیجه ترکیب به وجود امده اند. اگر این احتمال 0٪ باشد، کل نسل جدید،در اثر نسخه برداری عینی کروموزوم های نسل قدیم بوجود امده است. (این به این معنی نیست که نسل جدید همانند نسل قدیم است.) ترکیب با این امید صورت می گیرد که کروموزوم های جدید، حاوی بخشهای مناسب کروموزوم های قدیمی است و در نتیجه کروموزوم های جدید بهتر خواهد بود. با این حال، خوب است که برخی از قسمت های نسل قدیم برای نسل بعدی باقی بمانند.

تحلیل مکانیزم جا بجائی

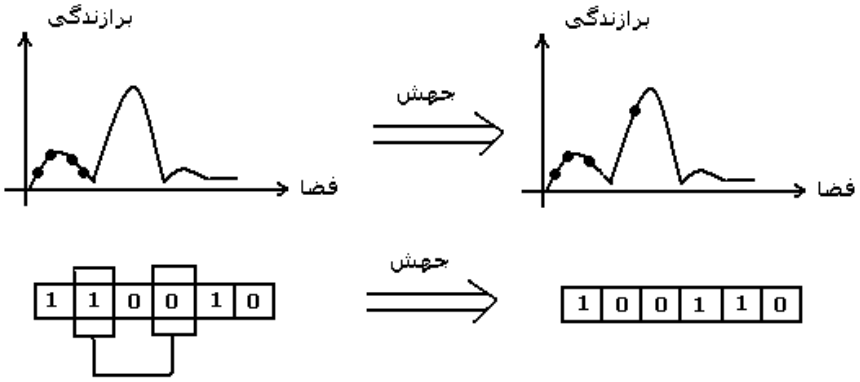
بدلیل اهمیت مرحله Crossover ، بسیاری از تحقیقات الگوریتم های ژنتیکی به روش های Crossover و تحلیل ان ها اختصاص یافته است. در این قسمت تعدادی از مهمترین، این روش ها مورد بحث قرار می گیرد.

معمولا کاربر های الگوریتم های ژنتیکی تعداد نقاط شکست کروموزوم در عمل Crossover را یک یا دو نقطه انتخاب می کنند. در روش Crossover دو نقطه ای را گسترش دهیم،Crossover چند نقطه ای خواهیم داشت، که هر رشته به صورت حلقه ای از بیت ها در نظر گرفته می شود که به K قسمت تعریف شده است، K تعداد نقاط شکست می باشد. در Crossover چند نقطه ای، قسمت های دو کروموزوم یکی در میان با هم تعویض می شوند.

بین اندازه جمعیت و نوع Crossover ارتباط مستقیم وجود دارد. تجربیات نشان می دهد که Crossover تک نقطه ای برای جمعیت های کوچک بسیار مناسب است. اما برای جمعیت های بزرگ تر Crossover دو نقطه ای مناسب تر است. Crossover های با تعداد نقاط شکست ثابت در جمعیت های کوچک امکان جستجوی بیشتری به ما می دهد. این واگرائی ذاتی در جمعیت های بزرگ لزوم جست و جوی بیشتر را کاهش می دهد و بنابر این Crossover دو نقطه ای جواب مناسب را به دست می دهد.

جهش

در طبیعت برخی عوامل مانند تابش اشعه ماورای بنفش باعث به وجود آمدن تغیرات غیر قابل پیش بینی در کروموزومها میشوند.از آنجایی که الگوریتمهای ژنتیکی از قانون تکامل پیروی میکنند در این الگوریتمها نیز عملگر جهش با احتمال کم اعمال میشود.جهش باعث جستجو در فضاهای دست نخورده مسئله میشود.میتوان استنباط کرد که مهمترین وظیفه جهش اجتناب از همگرایی به بهینه محلی است.در اشکال زیر نحوه جهش و کارکرد آن نمایش داده شده است:



به تعبیری دیگر میتوان جهش را مشابه شروع مجدد تصادفی الگوریتم تپه نوردی هنگام گیر افتادن در فلات دانست.

در الگوریتم ژنتیک نیز بعد از اینکه یک عضو در جمعیت جدید بوجود آمد، هر ژن آن با احتمال جهش، جهش مییابد .در جهش ممکن است ژنی از مجموعه ژن های جمعیت حذف شود یا ژنی که تا حال در جمعیت وجود نداشته است به آن اضافه شود .جهش یک ژن به معنای تغییر آن ژن است و وابسته به نوع کدگذاری، روشهای متفاوت جهش استفاده می شود.

همانطور که گفته شد، هر عضو وابسته به احتمال جهش، جهش میابد .احتمال جهش ، Pm ، مقداری است که توسط کاربر تعیین می شود .در الگوریتم استاندارد ژنتیک، بنا به دلایلی که در قسمت های بعد گفته خواهد شد، مقدار این پارامتر، بسیار کوچک، مثل Pm = 0,01 یا حتی Pm= 0,001 در نظر گرفته می شود .اما از آنجا که کمتر از شکل استاندارد این الگوریتم استفاده می شود، به عنوان یک پیشنهاد میتوان Pm را به صورت زیر تخمین زد:

$$p_m = \frac{1}{N}$$

که N تعداد ژنهای کروموزوم است.

در هر حال Pm مقداری بین 0 و 1 است و معمولا عددی کوچک انتخاب می شود .در فرزندی که بوجود آمده است (توسط ترکیب)، به ترتیب مقداری تصادفی بین 0 و 1 به هر ژن اختصاص می یابد .اگر این مقدار اختصاص داده شده از Pm کمتر باشد، ژن جهش می یابد و اگر بیشتر باشد، ژن تغییر نمی کند.

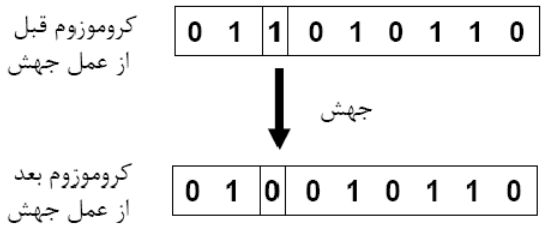
نرخ بالای جهش، باعث تنوع و پراکندگی ژنتیکی در جمعیت میشود که این پراکندگی ممکن است همگرایی را به تاخیر بیندازد .به همین دلیل، پیشنهاد می شود برای جمعیتهای بزرگ یا در نسل های آخر، از

Pm های کوچکتر و برای جمعیت های کوچک یا در نسل های ابتدایی از Pm های بزرگتر استفاده شود.
در ادامه، چند روش جهش بیان می شود.

تقسیم بندی روش های جهش

نوع اول: جهش باینری (Binary Mutation)

در الگوریتم ژنتیک با کد گذاری باینری، این عملگر اغلب با تولید تصادفی یکی از اعداد 0 و 1 جایگزینی ان به جای بیت مورد نظر صورت می گیرد(شکل (7)). اما در برخی کاربرد های ژنتیک، عمل جهش دودوئی در یک بیت با متمم ساختن ان بیت انجام می شود. به این صورت که اگر بیت مورد نظر 0 بوده به بیت 1 و بر عکس تبدیل خواهد شد، که آزمایش ها نشان داده شده است که روش دوم مناسب تر است.



شکل 7- عملگر جهش دودوئی

نوع دوم: جهش حقیقی (Real Mutation)

در کد گذاری حقیقی، عملگر جهش باعث تولید تصادفی یک مقدار جدید در یک موقعیت خاص در کروموزوم می شود. در نتیجه این تغییرات تصادفی در جمعیت کروموزوم ها ، نواحی بیشتری از فضای کاوش بررسی شده است و از همگرائی بی موقع (نا گهانی محلی) الگوریتم جلو گیری می شود. یک مثال از عملگر جهش حقیقی، جهش تصادفی یا یک نواخت می باشد. با این فرض که $C = (c_1,\ldots,c_i,\ldots,c_n)$ یک کروموزوم و C_i یک ژنی باشد که تحت عمل جهش قرار می گیرد انگاه C_i یک مقدار انتخابی تصادفی جدید از محدوده C_i می باشد، که به جای ژن C_i در کروموزوم جدید جایگذاری خواهد شد. مثال دیگری برای این روش عملگر جهش مرزی که در ان یکی از ژن های کروموزوم به طور تصادفی با حد پایین یا بالای محدوده ان ژن، جایگزین می شود($C_i = a_i$ یا $C_i = b_i$)

چند روش برای پیاده سازی عملگر جهش

وارونه سازی بیت

از این نوع جهش هنگامی استفاده میشود که کدگذاری، کدگذاری باینری باشد .در اینجا بیتی که شرایط جهش را دارد اگر 0 باشد به 1 و اگر 1 باشد به 0 تغییر مقدار میدهد .به عنوان نمونه اگر در شکل ...ژن چهارم شرایط جهش را داشته باشد به صورت نشان داده شده، جهش می یابد.

شکل **3-5**- وارونه سازی بیت.

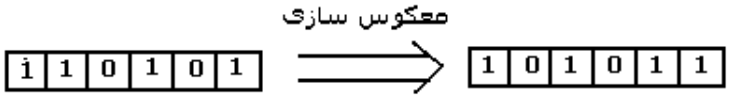
تغییر ترتیب قرارگیری

از این نوع جهش مخصوصا در الگوریتمهایی استفاده می شود که کدگذاری بر اساس مقدار باشد البته در دیگر کدگذاری ها مثل کدگذاری باینری هم میتوان این جهش را بکار برد .در این جهش، محل قرار گیری دو ژنی که می خواهند جهش بیابند در کروموزوم تعویض می شود .در شکل نمونه ای از این جهش، نشان داده شده است.

شکل **3-6**- تغییر ترتیب قرارگیری.

وارون سازی

این عمل در طبیعت بسیار رخ میدهد ولی در الگوریتمهای ژنتیکی به ندرت استفاده میشود و دلیل آن ایجاد تخریب زیاد است.این عملگر کروموزوم را معکوس میکند.



تغییر مقدار

این نوع جهش را نمیتوان برای کدگذاری باینری یا کدگذاری های مشابه که امکان تغییر ژنها وجود ندارد، به کار برد .در این جهش به ژنی که شرایط جهش را دارد مقداری اضافه یا کم می شود .اضافه شدن یا

کم شدن می تواند به تصادف انتخاب شود یا الگوریتم مقید به استفاده از یکی از این دو عمل باشد . مقداری که به ژن افزوده یا از آن کاسته میشود، وابسته به محدوده مقدار ژن است و باز میتواند به تصادف انتخاب شود یا برای الگوریتم تعریف شود .بدیهی است مقدارهای بزرگ پراکندگی ژنتیکی را افزایش می دهند .در شکل ...نمونهای از این جهش نشان داده شده است .این جهش خصوصا برای کدگذاری هایی که در آنها، ژنها به صورت اعداد حقیقی هستند، مناسب است.

(1.29,5.68,2.86, 4.11, 5.55)□ (1.29,5.68,2.73, 4.22, 5.55)

شکل **7-3**- تغییر مقدار.

همانطور که در شکل ...دیده میشود، ژنهای دوم و سوم کروموزوم جهش یافته اند .از شکل پیداست که مقدار انتخابی برای افزودن یا کاستن میتواند از ژنی به ژن دیگر متفاوت باشد.

احتمال جهش

اگر مرحله جهش صورت نگیرد، فرزندان بلا فاصله بعد از ترکیب و بدون هیچ تغییری بوجود می ایند(یا مستقیما نسخه برداری می شوند عمل ترکیب هم صورت نگرفته است). اگر تغییر صورت بگیرد، یک یا بیش از یک قسمت از کروموزوم تغییر می کند. اگر احتمال تغییر 100٪ باشد، یعنی همه کروموزوم های تتغییر کرده اند و اگر 0٪ باشد، هیچ چیز تغییر نکرده است.

به طور کلی جهش از قرار گرفتن GA در اکسترمم های محلی جلو گیری می کند. جهش نباید زیاد صورت بگیرد زیرا در این صورت بهGA به جستجوی کاملا تصادفی تبدیل خواهد شد.

GA پارامترهای دیگری نیز دارد. از پارامترهای مهم دیگر تعداد جمعیت و احتمال ترکیب می باشد.

محک اختتام اجرای الگوریتم ژنتیک

برای اینکه تشخیص دهیم چه موقع الگوریتم از اجرا متوقف شود، از شیو ه های مختلفی می توان استفاده کرد . به عنوان نمونه می توان همگرا شدن کل جمعیت را در نظر گرفت و یا اینکه فاصله ارزیابی (برازندگی) بهترین فرد جمعیت از متوسط ارزیابی ها (برازندگی ها) را در نظر گرفت که در این حالت

باید از حد مشخصی کوچک تر با شد، یا مقدار تابع هدف از حد مشخصی بیشتر باشد یا می توان تعداد نسل های مشخصی را به عنوان محک اختتام در نظر گرفت.

به طور کلی تحت شرایط زیر می توان الگوریتم را متوقف کرد

1- بدست آوردن جواب نهائی مورد نظر بعد از چند تکرار کم و یا قابل قبول بودن جواب به ازای خطای خاص.

2- اگز با پیشروی الگوریتم هیچ نوع بهبودی مشاهده نشد خواه الگوریتم جواب دلخواه را پیدا کرده باشد و یا اینکه در مینیمم محلی گیر کرده باشد.

3- اگر مقدار میانگین تابع هدف به ازای تعدادی تکرار به مقدار خاصی رسیده باشد.

4- الگوریتم به تعداد ثابتی از نسل ها رسیده باشد .

5- بیشترین درجه برازش فرزندان حاصل شود یا دیگر نتایج بهتری حاصل نشود.

6- بازرسی دستی.

7- ترکیبهای بالا

استدلال همگرایی الگوریتم ژنتیک

در این قسمت قضیه طرحواره 1 بیان میشود .قبل از پرداختن به خود قضیه نیاز به تعریفها و بررسیهایی است که در ادامه انجام میشود.

طرحواره

یک طرحواره الگویی است که با 1 ها ، 0 ها و * ها ساخته میشود.* مقداری است که میتواند 0 یا 1 می باشد. به عنوان مثال H=1**0*0 یک طرحواره است که میتواند به 8 شکل(نمونه)نوشته شود .یکی از این نمونه ها عبارتست از101010 .

بیتهای مخالف * در طرحواره، بیتهای مشخصه نام دارند .بیتهای مشخصه دو ویژگی دارند 1) مقدارشان مشخص است و2) مکانشان در طرحواره تعیین شده است .تعداد بیتهای مشخصه، مرتبه طرحواره o(H)

عبارت است از بیشترین فاصله بین دو بیت مشخصه مثلا در طرحواره $\delta(H)=5-1=4$,0**01 و در طرحواره $\delta(H)=5-3=2$,**1*1 .

با این تعریف ها بحث را ادامه میدهیم.

تاثیر انتخاب

فرض کنید m(H,t) تعداد نمونههای طرحواره H در زمان t در جمعیت P(t) باشد. باشد. F(H) را به عنوان مقدار میانگین تطابق در نمونههای طرحواره H در نظر میگیریم .در الگوریتم استاندارد ژنتیک احتمال انتخاب یک عضو برابر است با:

$$P_I = \frac{f_i}{\sum f_i}$$

بعد از تشکیل یک جمعیت جدید n عضوی ،p(t+1) ، از جمعیت p(t) میتوانیم بحث را به صورت زیر ادامه دهیم.

برای هر نمونه h_i از طرحواره H ، تعداد $nph_i = \frac{nf_{h_i}}{\sum f_j}$ کپی از h_i در p(t+1) وجود دارد. بنابراین

$$m(H,t+1) = n\frac{f_{h_1}}{\sum f_j} + n\frac{f_{h_2}}{\sum f_j} + \ldots + n\frac{f_{h_k}}{\sum f_j}$$

که $h_1, h_2, \ldots, h_k$ نمونه های H در p(t) میباشند .سپس

$$m(H,t+1) = n\frac{f_{h1} + f_{h_2} + \ldots + f_{h_k}}{\sum f_j}$$

$$= n\frac{m(H,t)}{m(h,t)} n\frac{f_{h1} + f_{h_2} + \ldots + f_{h_k}}{\sum f_j}$$

$$= nm(H,t)\frac{f_{h1} + f_{h_2} + \ldots + f_{h_k}}{m(H,t)}\frac{1}{\sum f_j}$$

$$= m(H,t)n\frac{f(H)}{\sum f_j}$$

از آنجا که $\bar{f} = \frac{\sum f_i}{n}$ میتوان نوشت:

$$m(H,t+1)=m(H,t)\frac{f(H)}{\bar{f}}$$

تعداد نمونه های طرحوارههایی که تطابقی بالاتر از تطابق میانگین جمعیت دارند، در جمعیت جدید افزایش می یابند .اگر فقط انتخاب را در نظر بگیریم، وابسته به مقدار تطابق، تعداد نمونه های مربوط به یک طرحواره در نسل های بعد یا کاهش می یابد و یا بیشتر می شود.

اکنون فرض کنید طرحواره ای مانند H داریم که تطابقش به مقدار $c\bar{f}$ از تطابق میانگین بیشتر است . c یک مقدار ثابت میباشد .بنابراین

$$m(H,t+1)=m(H,t)\frac{f(H)}{\bar{f}}$$

$$=m(H,t)=\frac{\bar{f}+c\bar{f}}{\bar{f}}$$

$$=(1+c)m(H,t)$$

اگر نقطه شروع را در t=0 در نظر بگیریم:

$$M(H,t+1)=m|(H,t)(1+c)^t$$

از این رابطه نتیجه می گیریم:

لم 1: تعداد طرحواره هایی که مقدار تطابقی بالاتر از تطابق میانگین داشته باشند در نسل بعد به صورت نمایی افزایش و تعداد طرحواره هایی که مقدار تطابقی کمتر از تطابق میانگین داشته باشند در نسل بعد به صورت نمایی کاهش مییابد.

تاثیر ترکیب

اگرچه طبق لم 1 طرحواره های بد از جمعیت حذف می شوند اما طرحواره جدیدی به جمعیت اضافه نمی شود .برای ایجاد طرحواره های جدید، به عبارت دیگر کروموزوم ها و ژنهای جدید از عملگر ترکیب در الگوریتم ژنتیک استفاده می شود .در اینجا ساده ترین شکل ترکیب در نظر گرفته میشود.

در این ترکیب، یک جفت به تصادف انتخاب می شوند .نقطه ای باز به تصادف در رشته های جفت تعیین می شود و زیر رشته های وابسته به این نقطه در دو عضو تعویض می شوند ترکیب تک نقطه ای که در قسمت عناصر الگوریتم ژنتیک به صورت کامل به آن پرداخته شده است.

در ابتدا، تاثیر ترکیب روی یک مثال بررسی می شود .عضو A را به صورت A=0111000 در نظر می گیریم .دو طرحواره از طرحواره های مرتبط باA را به شکل

$$H_1 = *1****0(\delta(H_1) = \delta_1 = 5)$$
$$H_1 = ***10**(\delta(H_2) = \delta_2 = 1)$$

تعریف می کنیم.

فرض کنید A به عنوان یکی از جفت ها برای ترکیب انتخاب می شود و نقطه ترکیب مکان بین 3 و 4 می باشد یا به عبارتی:

A=011-1000
H1=*1*-***0
H2=***-10**

در این مثال همانطور که دیده می شود از آنجا که نقطه ترکیب باعث از هم گسیختگی مقدارهای مشخصه طرحواره اول شده است، این طرحواره از بین می رود اما طرحواره دوم که آرایش مقدارهای ثابتش بر هم نخورده، زنده میماند.

بنابراین، تاثیر ترکیب روی یک طرحواره به طول مشخصه طرحواره بستگی دارد .حال این مساله به صورت عمومی بیان میشود .به طور کلی در ترکیب تک نقطه ای، L-1 موقعیت ترکیب برای رشتهای با طول L وجود دارد .اگر $\delta(H)$ طول مشخصه طرحواره H باشد، داریم:

برای از هم گسیختگی کاندیدا است $p(\frac{\delta(H)}{L-1}H)$

یک طرحواره که در یکی از جفتها به شکلی است که باید از هم گسیخته شود، باز هم می تواند زنده بماند اگر جفت دیگر شامل این طرحواره باشد .احتمال این اتفاق برابر است با:

$$m(H,t)\frac{f(H)}{|n\bar{f}} =$$

حال اگر چنین نباشد، یعنی طرحواره از هم گسیخته شود و در جفت دیگر نیز وجود نداشته باشد، احتمال این رخداد برابر است با:

$$\frac{\delta(H)}{L-1}\left(1-m(H,t)\frac{f(H)}{n\bar{f}}\right) =$$

البته یک طرحواره مانند H میتواند با ترکیب دو جفت که هیچ یک شامل این طرحواره نبوده اند بوجود بیاید .بنابراین

از هم گسیخته می شود $p(\le \frac{\delta(H)}{L-1}\left(1-m(H,t)\frac{f(H)}{n\bar{f}}\right)\|H)$

روابط بالا در صورتی معتبر است که تک تک اعضای جمعیت برای ترکیب انتخاب شوند اما معمولا در الگوریتم ژنتیک هر عضو با احتمالی مثل $p_0 < 1$ انتخاب می شود و ممکن است تمام اعضا برای ترکیب برگزیده نشوند .از این رو احتمال واقعی از هم گسیختگی طرحواره H به این صورت است که این طرحواره در ترکیب گسیخته شود(وابسته به نقطه ترکیب)و همچنین رشته ای(عضوی) انتخاب شود که شامل این طرحواره باشد .یعنی:

$$p(\le p_0 \frac{\delta(H)}{L-1}\left(1-m(H,t)\frac{f(H)}{n\bar{f}}\right)\|H)$$

با توجه به رابطه بالا، احتمال اینکه یک طرحواره، تغییر نکند عبارتست از:

$$p(\le 1- p_0 \frac{\delta(H)}{L-1}\left(1-m(H,t)\frac{f(H)}{n\bar{f}}\right)\|H)$$

در انتها، میتوان نتیجه گرفت تعداد نمونه های طرحواره H در نسل جدید بعد از انتخاب و ترکیب عبارتست از:

$$m(H,t+1) \ge m(H,t)\frac{f(H)}{\bar{f}}\left(1- p_0 \frac{\delta(H)}{L-1}\left(1-m(H,t)\frac{f(H)}{n\bar{f}}\right)\right)$$

این بررسی به صورت زیر بیان میشود:

لم 2: یک طرحواره با یک نرخ نمایی تکثیر می شود اما تنها در صورتی این اتفاق میافتد که طرحواره طول مشخصه کوچکی داشته باشد و مقدار تطابقش بیشتر از تطابق میانگین باشد.

با توجه به این مطالب در انتهای اجرای الگوریتم ژنتیک، میتوان انتظار داشت:

در جفت وجود دارد $p(H) = m(H,t)\frac{f(H)}{n\bar{f}} \approx 1$

بنابراین:

$$m(H,t+1) \ge m(H,t)\frac{f(H)}{\bar{f}}$$

به بیان دیگر طرحواره با حضور ترکیب تکثیر می شود و الگوریتم نیز همگرا می گردد اما تنها راه گریز از اپتیمم محلی، جهش است.

تاثیر جهش

جهش به تمام بیتهای یک عضو با احتمال $p_m << 1$ اعمال می گردد .طرحواره H در خلال جهش تنها در صورتی زنده خواهد ماند که هیچکدام از مقدارهای مشخصهاش، تغییر نکنند .احتمال اینکه یک بیت جهش نیابد عبارتست از:

$$1 - p_m$$

از این رو احتمال عدم تغییر طرحواره H در عملگر جهش برابر است با:

$$p(H) = (1 - p_m)^{0(H)}$$

برای p_m های به اندازه کافی کوچک میتوان با استفاده از بسط تیلور نوشت:

$$p(H) \approx (p_m)^{0(H)}$$

از این بررسی نتیجه میگیریم:

لم 3: طرحوارههایی که مرتبه بالا دارند نسبت به طرحواره هایی که مرتبه پایین دارند بیشتر تحت تاثیر جهش قرار میگیرند.

اگر طرحواره ها دارای طول های مشخصه کوچک باشند، به احتمال زیاد مرتبه پایینی هم دارند و تقریبا با جهش از هم گسیخته نمی شوند.

قضیه طرحواره

مباحث بالا میتواند به صورت قضیه زیر بیان شود که این قضیه، قضیه بنیادی الگوریتم ژنتیک می باشد .

قضیه 1(قضیه طرحواره)

فرض کنید H .یک طرحواره باشد m(H,t) تعداد نمونه های این طرحواره در جمعیت p(t) در زمان (نسل) t است . فرض کنید p_0 و p_m به ترتیب احتمال ترکیب و احتمال جهش باشد .با این فرضیات تعداد نمونههای مورد انتظار طرحواره h در جمعیت جدید، P(t+1) با وجود ترکیب جهش عبارتست از:

$$m(H,t+1) \geq m(H,t)\frac{f(H)}{\bar{f}}\left(1 - p_0\frac{f(H)}{\bar{f}}\left(1 - p_0\frac{\delta(H)}{L-1}\left(1 - m(H,t)\frac{f(h)}{n\bar{f}}\right)\right)\right)(-p_m)^{0(H)}$$

به بیان دیگر، نمونههای طرحوارههایی که طول مشخصه کوچک دارند، دارای مرتبه پایین هستند و مقدار تطابقشان بیشتر از مقدار تطابق میانگین است، به صورت نمایی افزایش مییابند.

پردازش مفید

در جمعیتی که n عضو به طول L دارد ، بین 2^L و $\min(n2^L,3^L)$ نمونه میتواند مورد پردازش قرار بگیرد .در عمل اما این تعداد نمونه پردازش نمیشود چون ترکیب مایل به نابود کردن نمونه های نامناسب است .در ادامه هدف بررسی این مساله است چه تعداد نمونه به صورت مفید مورد پردازش قرار میگیرد.

جمعیتی را با n عضو (رشته)باینری به طول L در نظر میگیریم .در اینجا بحث را به نمونه هایی محدود میکنیم که احتمالشان از مقدار ثابتی مانند p_0 بیشتر است .فرض کنید $p_m << 1, p_0 = 1$ بنا بر مطالب ذکر شده، احتمال زنده ماندن طرحواره ای مثل H در رابطه زیر صدق میکند.

$$p(H) \geq 1 - \delta(H)/(L-1)$$

از آنجا که ما فقط مواردی را بررسی میکنیم که

$$p(H) \geq p_s$$

باید داشته باشیم:

$$1 - \frac{\delta(H)}{L-1} \geq P_s$$

$$\Rightarrow \frac{\delta(H)}{L-1} \leq 1 - p_s$$

$$\Rightarrow \delta(H) \leq (1-p_s)(L-1)$$

که یک حد بالا را برای طول مشخصه طرحوارههایی که مورد بررسی قرار میگیرند، مشخص می کند .این حد بالا را با L_s باشد در رشتهای به طول L برابر با $(L-L_s)2^{ls}$ است .درنتیجه تعداد نمونه های زنده که احتمال بزرگتر از p_s دارند در جمعیتی با تعداد اعضای n برابر است با $n_s = n(L-L_s)2^{l_s}$ برای یک تخمین محافظه کارانه تر ، n را برابر $n = 2^{(L_s+1)/2}$ در نظر میگیریم .با این تعداد اعضا انتظار میرود یک نمونه، یا نمونه های کمی با طول مشخصه بزرگتر یا مساوی $(L_s+1)/2$ داشته باشیم.

از جایی که نمونه ها به صورت Binomially از بین می روند، نصف نمونه ها دارای مرتبه بیشتر از $L_s+1)/2$ و نصف دیگر دارای مرتبه کمتر از مقدار یاد شده می باشند .اگر فقط آنهایی را در نظر بگیریم که مرتبه بیشتری دارند، میتوان یک حد پایین برای تعداد نمونه ها به صورت زیر تخمین زد.

$$n_s \geq n(L-L_s)2^{l_s-1}$$

حالا

$$n = 2^{(l_s+1)/2}$$
$$\Rightarrow n^2 = \left|2^{L_S+1}\right.$$
$$\Rightarrow 2^{L_S-1} = \frac{N^2}{4}$$

از این رو

$$n_{s\geq} \frac{L-L_s}{4} n^3 = 0(n^3)$$

و می توان نتیجه گرفت:

علیرغم نابود شدن نمونههای مرتبه بالا و طولانی توسط ترکیب و جهش، الگوریتم ژنتیک به طور ذاتی تعداد زیادی از نمونه ها را پردازش می کند وقتی که در حال پردازش تعداد نسبتا کمی از اعضا می باشد.

انواع الگوریتم های ژنتیکی

الگوریتم های ژنتیک که نمونه اولیه ان توسط هالند در سال 1975 ارائه شد، تکامل طبیعی را در سطح ژن و کروموزوم شبیه سازی می کنند. عملکرد غالب در تولید نسل جدید، پیوند کروموزوم هاست، گر چه جهش در ژن ها نیز به عنوان یک عملکرد ثانوی به کار می رود.

تا کنون سه شاخه اصلی در این روش اصلی در این روش ارائه شده است:

1- الگوریتم ژنتیک سری

2- الگوریتم ژنتیک موازی

3- برنامه ریزی ژنتیک

الگوریتم ژنتیکی سری

الگوریتم ژنتیک سری همان الگوریتم ژنتیک معمولی است که در مقابل نوع موازی سری نام گرفته است.

تکامل یک پروسه بهینه سازی مبتنی بر تغییرات تصادفی تدریجی نمونه های مختلف در یک جمعیت و انتخاب احسن ان ها است. با مدل سازی این پروسه می توان یک تکنیک بهینه سازی اماری را به دست اورد که امروزه

در مسائل پیچیده مختلف و بخصوص مسائل طراحی ، کارائی خود را نشان داده است. در الگوریتم ژنتیکبه عنوان یکی از الگوریتم های تکاملی ، اثر کد های ژنتیکی در ترکیب و انتقال اطلاعات و هم چنین فرایند انتخاب طبیعی بر اساس سازگاری موجود با شرایط زیست محیطی مدل سازی است. در این الگوریتم نمونه هایی که در پروسه تکاملی قرار می گیرند، جواب های مختلف در فضاهای جواب هستند. متناظر هر جواب(نقطه در فضای جواب) ،یک نمونه ژنتیکی(genotype) به صورت یک رشته از کاراکتر ها(ژن ها) ، نسبت داده میشود. الگوریتم ژنتیک در هر تکرار محاسباتی (نسل) روی جمعیتی از رشته ها عمل می کند.تغییرات تصادفی روی مجموعه نمونه ها، از طریق اعمال مدل های ایده ال فرایند های ژنتیکی روی رشته ها انجام می شود. اما انتخاب طبیعی بر اساس نمود رفتاری(phenotype) هر رشته انجام می شود. بدین مفهوم که رشته ها رمز گشائی می شوند و جواب های مختلف از نظر عملکرد بر اساس تابع هدف ارزیابی شده و انتخاب، بر مبنای این ارز یابی و تصادف انجام می شود.

الگوریتم ژنتیکی موازی

تا کنون دو مدل اصلی در الگوریتم ژنتیک موازی مطرح گشته است. یکی مدل جزیره ای (island model) و دیگری مدل همسایگی(find Grained Model) در مدل جزیره ای چندین زیر جمعیت مجزا مطابق با الگوریتم ژنتیک معمولی تکامل می یابد و هر از چند گاهی زیر جمعیت های همسایه، بهترین کروموزوم یکدیگر را معاوضه می کنند. در مدل همسایگی یک مدل منفرد تکامل می یابد. هر کروموزوم این جمعیت در یک سلول از یک شبکه مشبک قرار دارد و الگوریتم ژنتیک سری، به صورت مجزا به هر سلول و همسایگانش که بر حسب شعاع همسایگی مشخص می شوند، اعمال می گردد. شبکه به صورت ترویدtriode) در نظر گرفته می شود تا از اثرات مرزی اجتناب گردد.

مقایسه ای بین رفتار این این الگوریتم با الگوریتم های معمولی نشان می دهد که مدل همسایگی به خاطر مکانیزم انتخاب محلی که از فشار انتخاب می کاهد، کاوش دقیق تری را در فضای جستجوفراهم می سازد.

از این جهت در مسائل ساده تر بدون بهبودی در عملکرد روش ، تنها بار محاسباتی اضافه تر تحمل می گردد. ولی مسائل مشکل تر از این طریقه جستجو سود خواهد برد.

شعاع همسایگی مناسب نیز به مسئله مورد حل بستگی دارد ولی حتی همسایگی های کوچک به شعاع یک یا دو، انتخابی مقاوم و اطمینان از رفتاری خوب را فراهم می سازند.

برنامه ریزی ژنتیکی

برنامه ریزی ژنتیک شاخه جدیدی از الگوریتم ژنتیک است که برای تکامل ساختاری های درختی بکار میرود. در برنامه ریزی ژنتیک کروموزوم ها دارای ساختمانی درختی متشکل از گره ها و شاخه ها هستند.

گره ها بر حسب مسئله مورد حل می تواند به گروه های متفاوت تقسیم بندی گردد. شاخه ها نیز معرف ارتباط بین گره ها می باشند. طول درخت فقط توسط قدرت محاسباتی محدود می گردد و خود الگوریتم تاثیری در ان ندارد.

قدرت این روش در تعین پارامتر های ستختاری نسبت به الگوریتم ژنتیک معمولی ضعیفتر می باشد. یک راه جبران این کاستی استفاده از روش مرکب برنامه ریزی- الگوریتم ژنتیک می باشد. در این کروموزوم از دو مورد هر کدام جهش و پیوند مر بوطه به صورت مجزا به کار گرفته می شود.

با توجه به روش های مختلفی که در پیاده سازی الگوریتم ژنتیک بکار برده می شود:

- روش سری از دقت بالا تری نسبت به روش محاطی بر خوردار می باشد.
- سرعت انجام عملیات در ازای محک اختتام های بالاتر بیشتر از روش محاطی است.

بالا بودن دقت سرعت عملیات برای روش سری در خصوص مواردی که تعداد واحد های تولیدی زیاد می باشند، بارز تر می باشد. پس به طور کلی می توان چنین نتیجه گرفت که روش سری از لحاظ دقت و سرعت انجام عملیات به خصوص برای مواردی که تعداد پارامتر ها زیاد باشند مناسب تر و بهتر است.

البته الگوریتم ژنتیک به روش های دیگری ا زجمله الگوریتم ژنتیک کلاسیک، مقاوم و تر کیبی نیز بیان شده است.الگوریتم ژنتیک کلاسیک شامل روش رمز کنندگی به رشته های بیتی و عملگر های انتخاب، تبادل وجهش هستند. اگر چه اثبات شده است که با وجود این روش ها و عملگرها، الگوریتم ژنتیک قادر به یافتن حل های به طور تقریب بهینه است، لیکن برای همگرایی نیاز به گذشتن تعداد نسل از جمعیت اولیه می باشد و به همین جهت این الگوریتم ها را به روش های دیگری نظیر به نخبه گزینی، روش برازندگی و.....مجهز می کنند تا کارائی ان ها از نظر سرعت و دقت افزایش دهند. به این صورت راه برای ابداع الگوریتم ژنتیک هیبریدی هموار می شود.

الگوریتم ژنتیک مقاوم، الگوریتمی است برای طیف وسیعی از مسائل، عمل کرد خوبی را نشان می دهد.

گاهی اوقات هدف اصلی از تحقیق درباره الگوریتم ژنتیک توسعه یک شکل مقاومی از این الگوریتم بوده است. واضح است که الگوریتمی مقاوم است که روش ها و عملگرهای به کار رفته در ان کمتر به شکل مساله مورد

نظر وابسته می باشد. یکی از مهم ترین اجزای الگوریتم ژنتیکی که تا حد زیادی مقاوم بودن ان را تعیین می کند، روشهای رمز کننده به فضای رشته های بیتی انتقال داد. سپس با استفاده از عملگر های متناسب با این نوع رشته ها، مساله مورد نظر را حل نمود. ولی در صورتی که بخواهیم از روش های رمز کننده دیگر نظیر رمز کردن به اعداد واقعی (دهدهی) استفاده کنیم ممکن است به سختی بتوان طیف وسیعی از مسائل نظیر مسائل بهینه سازی ترکیبی را با الگوریتم های ژنتیک دارنده دارنده این روش ها حل نمود. بنابراین برای داشتن الگوریتم های ژنتیک مقاوم بایستی اجرای این الگوریتم ها و به خصوص روش رمز کننده انها تا حد ممکن مقاوم شود.

الگوریتم ژنتیک ترکیبی، الگوریتمی است که علاوه بر داشتن روش های موجود، در الگوریتم های متداول و در روش های موجود، در الگوریتم های بهینه سازی متداول نیز استفاده می شود.

انگیزه ابداع این الگوریتم ژنتیکی این است که اگر چه الگوریتم های ژنتیکی متداول مقاوم هستند، ولی در حالت کلی از یک الگوریتم بهبنه سازی که روی یک حفره ویژه استفاده میشود ممکن است موفق تر نباشند. اما با ترکیب یک الگوریتم ژنتیکی متداول با الگوریتم هایی که به طور رایج استفاده می شود، می توان به الگوریتم هایی که بهتر از دو الگوریتم والد است، دست یافت. به طور معمول ترکیب نمودن مستلزم به کار گیری روش های نمایش دهنده و بهینه سازی است.

مقایسه الگوریتم ژنتیک با سیستم‌های طبیعی

سیستمهای طبیعی	الگوریتم ژنتیک
کروموزوم بسته‌های ژنی هستند که اطلاعات وراثتی را از نسلی به نسل دیگر عیناً انتقال می‌یابند	پاسخ‌های ممکن مساله به صورت رشته‌های عددی رمزگذاری شده است.
محیط شرایط محیطی را که جمعیت در آن قرار دارد.	تابع برازش مساله به صورت یک رابطهٔ ریاضی درآمده که تابع برازش می‌نامند.
اصل انتخاب طبیعی معیار بقای موجود زنده و تکثیر آن، سازش با محیط است.	تکثیر هر رشته جمعیت را به عنوان متغیر تابع برازش در نظر گرفته و مقدار تابع برازش هر رشته محاسبه می‌شود، متناسب با مقدار تابع برازش، رشته‌های جمعیت جدید انتخاب می‌شود.
تقاطع در نتیجه تقاطع یا تبادل قسمتی از کروموزوم‌ها مبادله ژنهای پیوسته صورت می‌گیرد	تقاطع (Crossover) رشته‌های جمعیت به صورت دو به دو مزدوج می‌شوند. زوج رشته‌ها از یک نقطه قطع می‌شوند. نیم بخشهای بین دو رشته تعویض می‌شوند.
جهش جانشین شدن ژنی به جای ژن دیگر در طول زنجیره DNA یا تغییرات ایجاد شده در ژن	جهش (Mutation) یک بیت از رشته عددی به صورت تصادفی انتخاب میشود و دچار تغییر می‌گردد.
ایجاد نسلهای جدید و تکامل موجودات	تکرار مراحل فوق بعد از مرحلهٔ تکثیر

نقاط قوت الگوریتم های ژنتیک

اولین و مهمترین نقطه قوت این الگوریتم ها این است که الگوریتم های ژنتیک ذاتاً موازی اند .اکثر الگوریتم های دیگر موازی نیستند و فقط می توانند فضای مسئله مورد نظر را در یک جهت در یک لحظه جستجو کنند

واگر راه حل پیدا شده یک جواب بهینه محلی باشدویا زیر مجموعه ای از جواب اصلی باشد باید تمام کارهایی که تا به حال انجام شده را کنار گذاشت ودوباره از اول شروع کرد.از آنجایی که GA چندین نقطه شروع دارد،در یک لحظه می تواند فضای مسئله را از چندجهت مختلف جستجو کند. اگر یکی به نتیجه نرسید سایر راه ها ادامه می یابند و منابع بیشتری را در اختیار شان قرار می گیرد.در نظر بگیرید: همه 8 عدد رشته باینری یک فضای جستجو را تشکیل می دهند،که می تواند به صورت ******** نشان داده شود.رشته 01101010 یکی از اعضای این فضاست.همچنین عضوی از فضاهای ********0و*******01و0 ******0و*1*1*1*0و 0**01*01 والی آخر باشد.

به دلیل موازی بودن واین که چندین رشته در یک لحظه مورد ارزیابی قرار می گیرند GA ها برای مسائلی که فضای راه حل بزرگی دارند بسیار مفید است .اکثر مسائلی که این گونه اند به عنوان "غیر خطی" شناخته شده اند.در یک مسئله خطی،Fitness هر عنصر مستقل است،پس هر تغییری در یک قسمت بر تغییر وپیشرفت کل سیستم تاثیر مستقیم دارد.می دانیم که تعداد کمی از مسائل دنیای واقعی به صورت خطی اند.در مسائل غیر خطی تغییر در یک قسمت ممکن است تاثیری ناهماهنگ بر کل سیستم ویا تغییر در چند عنصر تاثیر فراوانی بر سیستم بگذارد. خوشبختانه موازی بودن GA باعث حل این مسئله می شود ودر مدت کمی مشکل حل می شود.مثلاً برای حل یک مسئله خطی 1000 رقمی 2000 امکان حل وجود دارد ولی برای یک غیر خطی 1000 رقمی 2^{1000} امکان .

یکی از نقاط قوت الگوریتم های ژنتیک که در ابتدا یک کمبود به نظر می رسد این است که :GA ها هیچ چیزی در مورد مسائلی که حل می کنند نمی دانندو اصطلاحاً به آنهاBlind Watchmakers می گوییم . آنها تغییرات تصادفی را در راه حل های کاندیدشان می دهند وسپس از تابع برازش برای سنجش این که آیا آن تغییرات پیشرفتی ایجاد کرده اند یا نه، استفاده می کنند.مزیت این تکنیک این است که به GA اجازه می دهند یا با ذهنی باز شروع به حل کنند.از آنجایی که تصمیمات آن اساساً تصادفی است،بر اساس تئوری همه راه حلهای ممکن به روی مسئله باز است،ولی مسائلی که محدود به اطلاعات هستند باید از راه قیاس تصمیم بگیرند ودر این صورت بسیاری از راه حلهای نو وجدید را از دست می دهند.

یکی دیگر از مزایای الگوریتم ژنتیک این است که آنها می توانند چندین پارامتر را همزمان تغییردهند.بسیاری ازمسائل واقعی نمی توانند محدود به یک ویژگی شوند تا آن ویژگی ماکسیمم شود یا مینیمم و باید چند جانبه

در نظر گرفته شوند.GAها در حل این گونه مسائل بسیار مفیدند،و در حقیقت قابلیت موازی کار کردن آنها این خاصیت را به آنها می بخشد.و ممکن است برای یک مسئله 2 یا چند راه حل پیدا شود،که هر کدام با در نظر گرفتن یک پارامتر خاص به جواب رسیده اند.

به طور خلاصه مزایای الگوریتم ژنتیک را می توان در موارد زیر برشمرد

1. با متغیرهای پیوسته وهم گسسته میتواند عمل بهینه سازی را انجام دهد.
2. نیازی به محاسبه مشتق توابع ندارد.
3. بطور همزمان میتواند تمامی ناحیه جستجو شونده وسیع تابع هزینه را جستجو کند.
4. قادر به بهینه سازی مسائل با تعداد متغیرها ی زیاد میباشد.
5. قابل اجرا از طریق کامپیوترهای موازی است.
6. توابع هزینه ای که بسیار پیچیده باشند نیز از این طریق قابل بهینه سازی میباشند و الگوریتم در اکسترمم محلی به دام نمی افتد.
7. قادر است تا چند جواب بهینه را بطور همزمان بدست آورد نه فقط یک جواب.
8. الگوریتم های ژنتیک بر روی مجموعه ای از راه حل ها اعمال می شوند و نه بر روی یک راه حل خاص.
9. قادر است تا متغیرها را کد بندی نموده و بهینه سازی را با متغیر ها ی کدبندی شده انجام دهد. کد بندی سرعت همگرایی الگوریتم را افزایش میدهد.
10. الگوریتم توانایی کار کردن با داده های عددی تولید شده و داده های تجربی را علاوه بر توابع تحلیلی دارد.
11. فرآیند ارائه شده توسط الگوریتم های ژنتیک بر روی فضایی از مجموعه نمایندگان یا همان فضای کروموزوم ها اعمال می گردد و نه بر روی خود فضای راه حلها.
12. الگوریتم های ژنتیک از قوانین انتقالی احتمالی بجای قوانین انتقالی قطعی استفاده می کنند، بدین معنا که حرکت آن در هر نقطه از الگوریتم کاملا احتمالی بوده و بر اساس قطعیت صورت نمی پذیرد. این امر از مزایای مهم این روش بوده و از افتادن سیستم در کمنینه محلی جلوگیری می نماید. البته میزان احتمال به گونه ای است که احتمال حرکت به سمت هدف مساله بیشتر از احتمال حرکت آن به سمت مخالف جواب می باشد.

13. تنها ملاک ارزیابی و سنجش میزان شایستگی هر راه حل توسط الگوریتم های ژنتیک، مقدار تابع شایستگی آن در فضای کروموزوم ها می باشد و نه معیارهای مورد نظر در سطح فضای راه حل ها
14. برای حل برخی از مسائلی از رده NP-Hard نیز استفاده می شود.
15. این الگوریتم بیشتر در مسائل بهینه سازی و امثالهم بکار می رود

NP- hard Problems

نمونه ای از مسائلی که نمی توان آن ها را به روش سنتی حل کرد مسائل NP هستند.

کارهای زیادی وجود دارند که به الگوریتم های سریع Polynomial (چند جمله ای) را در مورد ان ها به کار برد. همچنین برخی از مسائل را نمی توان به روش الگوریتم حل کرد.

اما مسائل مهم زیادی نیز وجود دارند که یافتن راه حل در ان ها بسیار دشوار است. اما اگر راه حل را داشته باشیم، بررسی ان اسان می شود. این واقعیت منجر به مسائل NP- complete problems شد. NP معرف Nondeterministic (چند جمله ای های غیر جبری) و به این معناست که امکان این وجود که راه حل را حدس زد و سپس ان را بررسی کرد.

برای سهولت کار، بررسی مسائل NP- complete ، محدود به مسائلی است که پاسخ مس تواند بله یا خیر باشد. به دلیل وجود کارهایی با نتایج پیچیده، دسته دیگری از مسائل با نام NP-hard معرفی شده اند. این دسته مانند مسائل NP-complete محدود نیستند.

یکی از ویژگی های مسائل NP ان است که یک الگوریتم ساده را (که ممکن است در نگاه اول بدیهی به نظر برسد) می توان برای یافتن راه حل های مفید به کار برد. اما به طور کلی، این روش، روش های ممکن زیادی را فراهم می کند و بررسی کردن تمام راه حلها، فرایند بسیار کندی خواهد بود.

امروزه، هیچکس نمی داند که ایا الگورتم سریعتری برای یاقتن جواب دقیق در مسائل NP وجود دارد یا خیر. و یافتن چنین الگوریتمی وظیفه مهمی است که به عهده محققان می باشد. امروزه اکثر مردم تصور می کنند که چنین الگوریتمی وجود ندارد و بنابر این به دنبال روش دیگری (جایگزین) هستند. و نمونه ای از روش جایگزین، الگوریتم ژنتیکی است.

محدودیتهای GAها

یک مشکل چگونگی نوشتن عملگر Fitness است که منجر به بهترین راه حل برای مسئله شود.اگر این کارکرد برازش به خوبی و قوی انتخاب نشود ممکن است باعث شود که راه حلی برای مسئله پیدا نکنیم یا مسئله ای دیگر را به اشتباه حل کنیم. به علاوه برای انتخاب تابع مناسب برای Fitness ،پارامترهای دیگری مثل اندازه جمعیت،نرخ جهش وCrossover ،قدرت ونوع انتخاب هم باید مورد توجه قرار گیرند.

مشکل دیگر،که آن را نارس می نامیم این است که اگر یک ژنوم که فاصله اش با سایر ژنوم های نسل اش زیاد باشد(خیلی بهتر از بقیه باشد)و خیلی زود دیده شود(ایجاد شود)ممکن است محدودیت ایجاد کند و راه حل را به سوی جواب بهینه محلی سوق دهد.این اتفاق معمولاً در جمعیت های کم اتفاق می افتد.روش های Rank Scaling, tournament selection بر این مشکل غلبه می کنند

استراتژی برخورد با محدودیت‌ها

بحث دیگری که در اجرای الگوریتم ژنتیک وجود دارد چگونگی برخورد با محدودیت‌های مسأله می‌باشد، زیرا عملگرهای ژنتیک مورد استفاده در الگوریتم باعث تولید کروموزوم‌های غیر موجه می‌شود. میکالویچ چند تکنیک معمول جهت مواجهه با محدودیت‌ها تقسیم بندی نموده است که در ادامه به چند تا از آنها اشاره می‌شود.

استراتژی اصلاح عملگرهای ژنتیک

یک روش برای جلوگیری از تولید کروموزوم غیر موجه این است که عملگر ژنتیکی طوری تعریف گردد که پس از عمل بر روی کروموزم‌ها، کروموزم تولید شده نیز موجه باشد. در این حالت یکسری مشکلات وجود دارد. مثلاً پیدا کردن عملگری که دارای شرط فوق باشد بسیار دشوار بوده و از مسأله‌ای به مسأله دیگر متفاوت می‌باشد.

استراتژی ردی

در این روش پس از تولید هر کروموزوم آنرا از نظر موجه بودن تست کرده و در صورت غیر موجه بودن حذف می‌گردد. این روش بسیار ساده و کارا می باشد.

استراتژی اصلاحی

در این روش به جای اینکه کروموزوم غیر موجه حذف گردد تبدیل به یک کروموزوم موجه می‌شود. این روش نیز مانند روش اول به مسأله وابسته بوده و یافتن فرآیند اصلاح گاهی بسیار پیچیده می‌باشد .

استراتژی جریمه‌ای

در این روش بر خلاف سه روش قبل که از ورود جواب‌های غیر موجه جلوگیری می کردند، جواب غیر موجه با احتمال کم امکان حضور می‌یابند. سه روش فوق دارای این عیب بودند که به هیچ نقطه‌ای بیرون از فضای موجه توجه نمی‌کردند، اما در بعضی مسائل بهینه سازی، جواب‌های غیر موجه درصد زیادی از جمعیت را اشغال می‌کنند. در چنین شرایطی اگر جستجو فقط درناحیه موجه انجام گیرد شاید یافتن جواب موجه خیلی وقت‌گیر و مشکل باشد

استراتژی جریمه‌ای از متداول‌ترین تکنیک های مورد استفاده برای سر و کار داشتن با جواب‌های غیر موجه می‌باشد که در آبن ابتدا محدودیت‌های مسأله در نظر گرفته نمی‌شوند پس برای هر تخلف از محدودیت‌ها یک جریمه اختصاص داده می‌شود که این جریمه در تابع هدف قرار می گیرد.

مسأله اصلی چگونگی انتخاب یک مقدار مناسب برای مقدار جریمه می‌باشد تا در حل مسائل به ما کمک نماید.

نکته‌ای که در روش جریمه وجود دارد این است که یک جواب غیر موجه به سادگی حذف نمی‌شود زیرا ممکن است در ژنهای آن اطلاعات مفیدی وجود داشته باشد که با اندکی تغییر به جواب بهینه تبدیل شود

بهبود الگوریتم ژنتیک

برای بهبود دادن GA می توانیم تغییرات زیر را اعمال کنیم

1- استفاده از بهینه گر محلی.

2- تغییر پارامترهائی از قبیل تغییر جمعیت اولیه، آهنگ جهش و کسر ادغام.

3- تغییر GA باینری به پیوسته و بالعکس.

چند نمونه از کاربرد های الگوریتم های ژنتیک

نرم‌افزار شناسایی چهره (شناسایی چهره با استفاده از تصویر ثبت شده . در این روش، شناسایی چهره براساس فاصله اجزای چهره و ویژگی‌های محلی و هندسی صورت می‌گیرد که تغییرات ناشی از گیم، تغییرات نور و افزایش سن کمتین تأثیر را خواهد داشت .همچنین گراف‌ها برای چهره‌های جدید با استفاده از الگویتم‌های ژنتیک ساخته شده و با استفاده از یک تابع تشابه، قابل مقایسه با یکدیگر هستند که این امر تأثیر به‌سزایی در افزایش سرعت شناسایی خواهد داشت).

توپولوژی های شبکه های کامپیوتی توزیع شده.

بهینه سازی ساختار ملکولی شِمیایی (شیمی)

مهندسی برق برای ساخت آنتنهای Crooked-Wire Genetic Antenna

مهندسی نرم افزار

بازی های کامپیوتری

مهندسی مواد

مهندسی سیستم

رباتیک(Robotics)

تشخیص الگوو استخراج داده(Data mining)

حل مسئله فروشنده دوره گرد

آموزش شبکه های عصبی مصنوعی

یاددهی رفتار به رباتها با GA .

یادگیری قوانین فازی با استفاده از الگویتم های ژنتیک.

یک مثال ساده

مثال: فرض کنید تابع $f(x) = -x^2 + 6x - 3$ را داریم . می خواهیم برای $0 \le x \le 15$ و $x \in N$ ماکزیمم تابع را بیابیم.

مراحل حل مساله به روش الگوریتم ژنتیک به شرح زیر است :

1.کدکردن (Encoding)

برای اینکه ما جوابهای ممکن یعنی $0 \le x \le 15$ را کدگذاری کنیم یکی از روشهایی که در پیش داریم تبدیل هر عدد به یک عدد باینری متناظر است. چون بیشترین جواب ممکن ما 15 است و متناظر باینری آن 1111 چهار بیتی است لذا تمام جواب های ما (کروموزوم ها) دارای طولی برابر $l=4$ می باشند.

2. تابع ارزش (Evaluation)

تابعی که مقداری برای fitness حساب می کند مستقیما از خود تابع f(x) بدست می آید.

یعنی اگر به ازای یک a و b : $f(b) > f(a)$ باشد آنگاه b دارای fitness بیشتری است. یا اینکه b از a بهتر است.

3. انتخاب (Selection)

فرض کنیم که در آن جمعیت اولیه با m فرد که $m < n$ دوعدد 3 و8 دارای fitness بیشتری نسبت به جمعیت خود بودند و احتمال انتخاب بیشتری داشتند. بعد از انتخاب این دو عدد به عنوان parent الگوریتم وارد مراحل تولید نسل جدید می شود.

$3 \rightarrow 0\,0\,1\,1$ $\quad f(3) = 6$

$8 \rightarrow 1\,0\,0\,0$ $\quad f(8) = -19$

4. ترکیب (Crossover)

فرض کنیم که ترکیب عمل زیر را انجام دهد.

$0\,0\,1\,1$
$1\,0\,0\,0$
$\xrightarrow{\text{point} = 3}$
$0\,0\,1\,0$
$1\,0\,0\,1$

5. جهش (mutation)

و نیز عملگر جهش :

$0\,0\,1\,0$
$1\,0\,0\,1$
$\xrightarrow{\text{point} = 2}$
$0\,1\,1\,0 = (6)$
$1\,1\,0\,1 = (13)$

حال کروموزوم های متولد شده جزو نسل جدید به حساب می آیند و به جمعیت اولیه افزوده می شوند وجوابهای با fitness پایین حذف می شوند و الگوریتم دوباره با n فرد به کار خود ادامه می دهد.

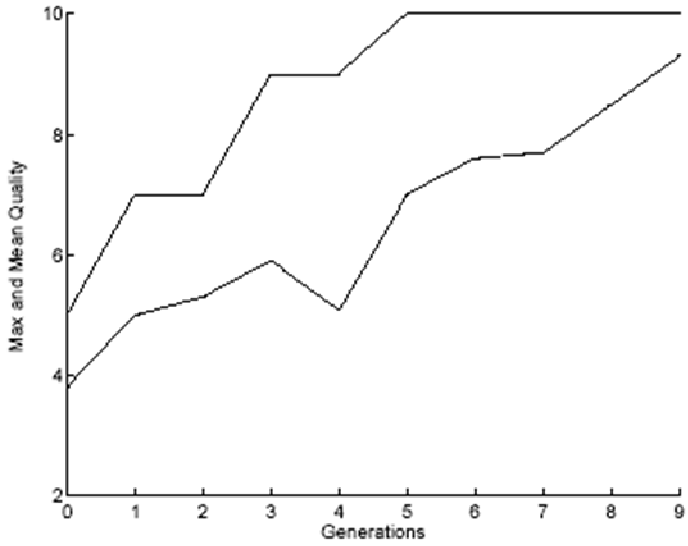
یک نکته لازم به تذکر است که ممکن است بهترین جواب (برای مثال x = 6) طی مراحل بعد به نحوی از بین برود. برای جلوگیری از این امر بعد از هر بار انجام الگوریتم بهترین جواب را در جایی کنار می گذاریم (far) تا همیشه بهترین جواب حاصل از هر بار اجرای الگوریتم را داشته باشیم.

معیارهای مختلفی را می توان برای توقف الگوریتم در نظر گرفت و معمولا چند معیار برای توقف استفاده می شود تا احتمالهای مختلف وقوع پیشامدها در طی اجرای الگوریتم حساب شوند. یک معیار می تواند این باشد که بهترین جواب را بعد از اجرای تعداد مشخصی بار از الگوریتم تغییر ندهد. یا معیار دیگر اینکه میانگین fitness جوابهای موجود در جمعیت جاری همان fitness بهترین جواب یا بسیار نزدیک به آن باشد.

یا اینکه می توانیم از پیش قرارداد کنیم که الگوریتم به تعداد مشخصی اجرا شود. معمولا روتین های توقف مختلف است و بستگی به پیچیدگی و چگونگی مساله دارد.

در اینجا ما برای تعریف تابع evaluation از خود تابع استفاده کردیم و لزومی برای نسبت دهی یک مقدار به عنوان quality نبود.

اگر ما این مراحل را به گونه ای دنبال کنیم که بعد از هر بار اجرا جواب های بد حذف و جوابهای نسل جدید که احتمالا fitness بالایی دارند جایگزین آنها شوند به مرور فاصله fitness بین min و Max کم می شود و میانگین جواب ها به بهترین جواب میل می کند و این نشان می دهد که ما به طور کلی به بهترین جواب نزدیک می شویم.



سوالی که در اینجا مطرح می شود اینست که اگر در مثال قبل به جای فضای گسسته فضای پیوسته [0 , 15] مطرح می شد الگوریتم به چه شکلی انجام می گرفت؟

در این حالت ما یک ε تعریف می کنیم تا هر بیت جواب در مراحلی مثل mutation به اندازه ε تغییر کند. بدیهی است ما هر چقدر این ε را کوچکتر فرض کنیم میزان محاسبات و تکرار الگوریتم بالا می رود و به همین دلیل زمان اتمام الگوریتم بالا می رود. عامل دیگری که در زمان موثر است n اولیه در جمعیت اولیه است که با زیاد فرض کردن n زمان اتمام بالا می رود.

مسائل حل شده

الگوریتم ژنتیک و حل مساله TSP

مقدمه :

مساله فروشنده دوره گرد (The Traveling Salesman Problem) به این شکل است که می خواهیم در یک تعداد شهر (n) دوری پیدا کنیم که از هر شهر دقیقا یک بار عبور کند و در پایان به شهر آغازین بازگردد به طوری که طول دور مینیمم باشد.

هر چند منشا TSP دقیقا مشخص نیست ولی قطعا در حدود سال 1931 بوده است. اولین نمونه شبیه به این مساله توسط Euler در سال 1759 مطرح شد و به این صورت بود که یک مهره اسب می بایست روی برد شطرنج حرکت کند و از هر خانه دقیقا یک بار عبور کند.

این مساله را می توان به صورت ریاضی هم شبیه سازی کرد. به این ترتیب که ما در یک گراف وزن دار G(v,e) دوری فراگیر (اویلری) با مینیمم مجموع وزنهای یالهای گذرنده می خواهیم بیابیم.

به روش ریاضی مساله با یافتن تعداد جایگشت های n شی متمایز وسپس ارزیابی هر حالت بررسی می شود.

تعداد جایگشتها n! است. برای یافتن مینیمم دورها نیز به حداکثر n! محاسبه احتیاج داریم. ولی اگر n را زیاد فرض کنیم تعداد محاسبات بسیار بالا خواهد بود به همین دلیل گفته می شود که الگوریتم حل مساله در زمان چند جمله ای نیست.(None-Polynomial)

حال می خواهیم چند الگوریتم ارائه شده برای حل مساله TSP را مطرح کنیم. تا به امروز الگوریتمی بدست نیامده است که این مساله را در زمان چندجمله ای (Polynomial) حل کند. به همین دلیل ما بهینگی را فدای زمان می کنیم تا بتوانیم در یک زمان معقول به یک جواب خوب برسیم.

یکی از الگوریتمهای ارائه شده الگوریتم حریصانه است (Greedy). به این شکل که الگوریتم یک لیست از همه یالها در گراف ایجاد می کند و سپس آنها را از کمترین هزینه به بیشترین هزینه مرتب می کند. سپس ابتدا یالی با کمترین هزینه را انتخاب می کند و چک می کند که این یال سبب نشود که دوری در گراف ایجاد شود که از همه رئوس عبور نکند.

روش دیگری نیز هست که شبیه به Greedy است که به نام Nearset Neighbor مطرح شده است.

به این شکل که یک نقطه آغازین به صورت تصادفی انتخاب می کنیم وبعد به نزدیکترین راس بعدی می رویم و از آن راس نیز همین کار را ادامه می دهیم. البته باز هم شرط می کنیم که در هر مرحله دوری که اویلری نباشد ایجاد نشود.

این روشها همیشه راه حل خوبی ارائه نمی دهد چون اغلب آخرین یال اضافه شده وزن نسبتا زیادی دارد.

الگوریتم بعدی A minimum spanning tree میباشد.

ابتدا ما یک درخت پوشای مینیمم با n-1 یال می یابیم. سپس می توانیم یک دور به وسیله رفتار وچگونگی یالها در درخت پوشایمان ایجاد کنیم (مثل یالهای دو جهتی). بعد از یک شهر که فقط به یک شهر دیگر متصل شده است شروع می کنیم و با دنبال کردن یالهای طی نشده برای شهرهای جدید ادامه می دهیم. اگر یال طی نشده ای وجود نداشت به یال قبلی بر می گردیم و این کار را ادامه می دهیم تا به شهر ابتدایی بازگردیم. با این کار ما یک کران بالا برای TSP خواهیم داشت.

قابل توجه است که ما بعضی از شهرها را بیش از یک بار بازدید کردیم. وقتی که نیاز داریم به عقب حرکت می کنیم ولی در عوض به شهر بازدید نشده بعدی می رویم. بعد از اینکه همه شهرها بازدید شدند به شهر شروع برمی گردیم.

حل مسئله TSP به وسیله الگوریتم ژنتیک:

مراحل الگوریتم ژنتیک برای مسئله به صورت زیر است

1. Encoding

برای encoding می توانیم یک ماتریس مجاورت گراف ایجاد کنیم که شامل 1 در مکان i و j است اگر یک یال از راس i به راس j وجود داشته باشد و در غیر این صورت 0 است.

حال می توانیم از این ماتریس همان گونه که هست استفاده کنیم یا به این صورت که سطرهای ماتریس را به هم الحاق کنیم و رشته ای طولانی از 0 و 1 ها ایجاد کنیم ولی توجه داریم که این رشته نمایش باینری عدد نیست.

به این صورت اولین متد encoding ساخته می شود.

برای مثال ماتریس مقابل :

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

دوری را نشان می دهد که از شهر1 به شهر 3 و از شهر 3 به شهر 2 و از شهر 2 به شهر 1 می رود.

متد بعدی که مورد بررسی قرار می گیرد متدی است که یک رشته از اعداد که معرف شماره شهر است را می سازد.

اولین شیوه ساخت این متد به این صورت است که برای مثال رشته v = 1234 این مفهوم را می رساند که ابتدا از شهر 1 شروع کرده سپس به شهر 2 می رویم و از آنجا به شهر 3 و بعد به شهر 4 حرکت می کنیم. در پایان هم به شهر اول بر می گردیم. توجه کنید که دو رشته 1234 و 2341 معادلند.

شیوه دوم این است که اگر رشته v = 3124 را داشته باشیم

به این معنی است که دور از شهر 1 به شهر 3 و از شهر 3 به شهر 2 و از شهر 2 به شهر 4 و در پایان از شهر 4 به شهر 2 می رویم.

قابل توجه است که هر رشته ممکن در اینجا یک دور مجاز را نشان نمی دهد. مثلا رشته 3412 نشان می دهد که ما از شهر 1 به شهر 3 می رویم و به شهر 1 بازمی گردیم واز شهر 2 به شهر 4 می رویم و به شهر 2 باز می گردیم که یک رشته بی ربط است.

2. Crossover

متد اول Partially Matched Crossover (PMX) می باشد.

اگر دو رشته مقابل را داشته باشیم :

1 2 3 4 | 5 6 7 | 8
8 5 2 1 | 5 6 7 | 7

و یک crossover دو نقطه ای انجام دهیم خواهیم داشت :

v1 = 1 2 3 4 | 3 6 4 | 8
v2 = 8 5 2 1 | 5 6 7 | 7

که به طور بدیهی غیر مجاز هستند چون v1 شهر 5 یا 7 را بازدید نمی کند و شهر های 3 و 4 را دو بار بازدید می کند. به طور مشابه v2 شهر های 3 و 4 را نمی بیند و شهر های 5 و 7 را دو بار می بیند.

PMX بدون استفاده از ابزاری این مشکل را به این گونه حل می کند که تعویض های 5 ↔ 3 و 6 ↔ 6 و 7 ↔ 4 را انجام می دهد سپس این تعویض ها را عینا روی ژن های خارج از نقاط crossover تکرار می کند.

و به این ترتیب رشته های زیر ساخته می شود :

1 2 5 7 3 6 4 8
8 3 2 1 5 6 7 4

ولی ما در این روش لزوما یک دور مجاز تولید نمی کنیم به همین دلیل نیاز داریم تا با یک روتین بهتر برای crossover راه حل بهتری پیدا کنیم و دورهایی که مجازند را تولید کنیم.

متد دوم Cycle Crossover (CX) می باشد.

لازم به ذکر است که این متد crossover روی اولین شیوه encoding عمل می کند یعنی رشته 1234 به این مفهوم است که به ترتیب از 1 به 2 و به 3 و به 4 می رویم و در آخر به 1 بر می گردیم.

فرض کنید دو رشته زیر را داریم:

1 2 3 4 5 6 7 8
8 5 2 1 3 6 4 7

ما برای تولید یک رشته جدید v1 به این شکل عمل می کنیم :

اولین ژن را از یکی از parent ها انتخاب می کنیم:

v1 = 1 - - - - - - -

باید هر عنصر را از یکی از parent ها برداریم و آنرا در موقعیتی که قبلا بوده قرار دهیم.

از آنجاییکه اولین موقعیت توسط 1 اشغال شده است عدد 8 از رشته دوم نمی تواند به آن محل برود پس مجبوریم 8 را هم از رشته اول برداریم:

v1 = 1 - - - - - - 8

و به همین شکل 7 و 4 را نیز از رشته اول برداشته و در محل خود قرار دهیم :

v1 = 1 - 4 - - - 7 8

حال با پر کردن بقیه محل ها با عناصر آن محلها از رشته دوم v1 را می سازیم.

این متد همیشه یک کروموزوم مجاز می سازد.

البته ممکن است فرزند درست شده همان parent باشد ولی این مشکل نیست چون نشانگر اینست که fitness, parent بالایی دارد و باز هم می تواند یک انتخاب باشد.

متد سوم Order Crossover می باشد که خیلی به PMX شبیه است.

همان دو نقطه را در نظر می گیرد ولی به جای اصلاح کروموزومها با تعویض تکرارها به طور ساده تری بقیه ژن ها را مرتب می کند تا یک دور مجاز بدهد.

مثلا اگر دو رشته زیر را داشته باشیم :

1 3 5 | 7 6 2 | 4 8
5 6 3 | 8 2 1 | 4 7

با تعویض کردن ژنهای بین دو نقطه بدست می آید :

v1 = - - - | 8 2 1| - -
v2 = - - - | 7 6 2| - -

سپس با شروع از دومین نقطه crossover ژنها را از کروموزوم parent ثبت می کنیم.

4 8 1 3 5 7 6 2
4 7 5 6 3 8 2 1

سپس ژن هایی را که بین نقاط crossover بوده اند را حذف می کنیم. یعنی 8 و 2 و 1 را از لیست v1 و 7 و 6 و 2 را از لیست v2 حذف می کنیم تا رشته های زیر بدست آیند :

4 3 5 7 6
4 5 3 8 1

و با جایگزینی از دومین نقطه crossover با کروموزومهای فرزند رشته های نهایی زیر بدست می آیند:

v1 = 5 7 6 8 2 1 4 3
v2 = 3 8 1 7 6 2 4 5

متد چهارم Crossover Metrix می باشد. همان crossover یک یا دو نقطه ای می باشد.

اگر ماتریس های زیر را داشته باشیم :

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

نقاط crossover را بعد از اولین ستون و بعد از دومین ستون انتخاب می کنیم. crossover کردن ستون ها نتیجه می دهد:

$$A' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

$$B' = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

حالا ما چندین 1 در بعضی سطرها داریم و بعضی از سطرها اصلا 1 ندارند. این را بوسیله انتقال یکی از 1 ها از سطری که چندین 1 دارد به سطری که هیچ یکی ندارد درست می کنیم.

این انتخاب بین سطرهای شامل چند 1 تصادفی است.

$$A'' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$B'' = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

حال با ملاحظه ماتریس اول می بینیم که a ↔ a و b → c → b . پس ما دو دور مختلف ایجاد کرده ایم ولی این را بوسیله برش و متصل کردن درست می کنیم.

یال a به a را و همچنین یکی از یالهای بین b و c را میبریم و a را به b و c وصل می کنیم.

متد پنجم Modified Order Crossover (MOX) شبیه crossover یک نقطه ای است.

یک نقطه crossover تصادفی در parent انتخاب می کنیم و به طور معمول ژنهای قبل از نقطه را همان طور که هستند رها می کنیم. سپس ژنهای بعد از نقطه crossover را به ترتیبی که در دومین کروموزوم parent ظاهر شده اند دوباره مرتب می کنیم.

اکر دو کروموزوم زیر را داشته باشیم :

1 2 3 | 4 5 6
3 6 4 | 2 1 5

بدست خواهیم آورد :

v1 = 1 2 3 | 6 4 5
v2 = 3 6 4 | 1 2 5

crossover تا حالا تمرکز روی موقعیت شهر در دور را جستجو کرده در حالیکه در واقع یالها مهمترین قسمتهای سفر فروشنده دوره گرد هستند زیرا آنها وزنها را معین می کنند بنابراین چیزیکه ما واقعا می خواهیم این است که با یالها بیشتر از موقعیت هر شهر سروکار داشته باشیم.

Grefenstette (1981) یک روتین crossover اختراع کرده که هر راس را از یکی از آنهایی که برای راس جاری در یکی از parentها لازم است بر می دارد.

این را بوسیله ایجاد یک لیست یال برای هر راس انجام می دهیم.

کروموزومهای:

v1 = 1 2 3 4 5 6
v2 = 3 6 4 2 1 5

ابتدا یکی از نخستین راسها را از یکی از parentها یعنی1 یا 3 در این مثال انتخاب می کنیم آن را که کمترین عدد از راسهای لازم را دارد انتخاب می کنیم و یا اگر آنها عدد یکسانی دارندتصادفی یکی را انتخاب می کنیم. سپس مشاهده می کنیم که راسهایی با راس 1 تلاقی می کنند. چون این همان راسی است که اول انتخاب کردیم دوباره راسی با کمترین عدد از راسهای لازم که قبلا انتخاب نشده اند انتخاب می کنیم سپس راس 2 را انتخاب می کنیم این فرایند انتخاب راسهای فرضی را ادامه می دهیم اگر به حالتی برخورد کنیم که نتوانیم راسی را که قبلا انتخاب نشده را انتخاب کنیم یک گره که قبلا انتخاب نشده است را تصادفی انتخاب می کنیم. این به آن معنی است که ما می خواهیم راسی را بدست آوریم که در راس جاری ما در یکی از parent ها روی نداده است اما متاسفانه این اجتناب ناپذیر است بنا براین کروموزوهای parent ما می توانند فرزند تولید

کنند. توجه داشته باشید که ما در تمام مدت در توانایی انتخاب راس هایی که در یکی از parent ها لازم بودند موفق بودیم. ما فقط یک فرزند از این crossover بدست می آوریم تا بسیاری از crossover ها را دوبار انجام دهیم تا یک نسل جدید ایجاد کنیم.

همچنین عملگرهای crossover ای داریم که از اطلاعات مکاشفه ای استفاده می کند.

Heuristic Crossover یک راس تصادفی برای شروع انتخاب می کند و سپس به دو یالی که راس جاری را در کروموزومهای parent رها می کند توجه می کند و کوتاهترین یالی را که یک دور را نشان نمی دهد بر می دارد. اگر هر دو یال یک دور را نشان دهند به طور تصادفی یالی را که این کار را انجام ندهد انتخاب می کنیم.

3. Mutation:

ابتدا عملگر دو گزینه ای (2 - opt) را نشان می دهیم. دو یال (a,b) و (c,d) را از دورمان انتخاب می کنیم و چک می کنیم که آیا می توانیم این 4 راس را با یک روش متفاوت به هم وصل کنیم تا کمترین وزن را به ما بدهد یا خیر. برای انجام این کار چک می کنیم که اگر Cab +Ccd > Cac+ Cdb باشد یالهای (a,b) و (c,d) را با یالهای (a,c) و (d , b) عوض کنیم. توجه کنید که فرض کرده ایم که a و b و c و d با ترتیب مشخصی در دور ظاهر شده اند حتی اگر b و c متصل نباشند.

همچنین یک عملگر سه گزینه ای (3- opt) داریم که به جای دو یال سه یال تصادفی را نشان می دهد. اگر یالهای (a, b) و (c, d) و (e, f) را داشته باشیم چک می کنیم که اگر

Cab + Ccd + Cef > Cac + Cbe + Cdf باشد یالهای (a,b) و (c,d) و (e,f) را با یالهای (a,c) و (b,e) و (d,f) عوض کنیم.

عملگر (or-opt) شبیه (2-opt) است.یک مجموعه از راس های متصل را تصادفی انتخاب می کنیم و چک می کنیم که آیا این رشته می تواند بین دو راس دیگر اضافه شود تا وزن تقلیل یابد یا خیر.

ما می توانیم این را بوسیله پیدا کردن مجموع وزن های یالهای اضافه شده و مجموع وزن های یالهای حذف شده محاسبه کنیم. اگر وزن یالهای حذف شده بیشتر بود تعویض انجام گیرد.

سه عملگر Mutation دیگر نیز وجود دارند که یک شهر انتخابی تصادفی را به یک مکان انتخاب شده تصادفی اضافه می کند.

همچنین ما هنگامیکه دو شهر تصادفی را انتخاب می کنیم و آنها را تعویض می کنیم یک mutaion دو جانبه داریم.

مقایسه روشهای مختلف الگوریتم ژنتیک برای TSP:

تا به حال فرمهای مختلفی از رمزگذاری ها encoding و عملگرهای crossover و mutation را در حل مساله TSP به روش الگوریتم ژنتیک دیدیم. این حالتها می توانند با هم ترکیب شوند و منجر به رسیدن به راه حل های مختلفی برای TSP به روش الگوریتم ژنتیک شوند. ولی از آنجایی که متدهای crossover روی encoding های خاصی عمل می کنند در نتیجه الگوریتمهای ژنتیک خیلی متفاوتی برای جستجو نداریم.

حال به بررسی الگوریتمهای ژنتیک محض یعنی بدون استفاده از Heuristic Information می پردازیم.

فرض کنید که PMX crossover را انتخاب کرده ایم و هیچ عملگری را برای mutation اتخاذ نکرده ایم. با این شرایط در 33 شهر به جوابی می رسیم که طول آن 10 درصد از جواب بهینه بیشتر است. و برای 100 شهر این میزان به 210 درصد می رسد. اگر در یک مساله که از 30 شهر تشکیل شده است اگر از PMX استفاده کنیم بهترین طول 498 و اگر از Order Crossover استفاده کنیم این میزان به 425 کاهش می یابد. در حالی که Cycle Crossover نتیجه ای برابر 517 می دهد.از آنجایی که می دانیم در این مساله خاص(30 شهر) بهترین جواب طولی برابر 420 دارد به نظر می رسد که Order Crossover جوابی بهتر از بقیه بر می گرداند.

حال به بررسی Matrix Crossover می پردازیم. اگر از یک crossover دو نقطه ای استفاده کنیم مشاهده می کنیم که برای 30 و 50 و 75 و 100 و 318 دورهایی با طول 420 و 426 و 535 و 629 و 42154 را ارائه می کند. که همه این جوابها کمتر از 2 درصد بیشتر از جواب بهینه هستند. پس احتمالا استفاده از یالها بسیار امیدوار کننده تر از استفاده از راسها به عنوان متغیر است.توجه کنید که به هر حال نمایش ماتریس فضای بیشتری را برای ذخیره کردن نسبت به نمایش به صورت عدد صحیح و crossover ساده می خواهد و در ضمن محاسبات crossover و mutation در ماتریس پیچیده تر و زمانبرتر است.

همچنین روش دیگری که تست شده اینست که ما از (2-opt) برای mutation استفاده کنیم و از crossover استفاده نکنیم. این روش نیز جواب خوبی ارائه می دهد ولی جواب قبلی بهتر از این روش است. در ضمن برای وقتی که n را زیاد فرض می کنیم این روش جوابی مناسب ارائه نمی دهد.

Heuristic Algorithm نیز به جواب خوبی می رسد. Heuristic Algorithm وقتی که با (2–opt) mutation ترکیب می شود بهترین جواب را در مقایسه با متدهایی که تا به حال گفتیم برمی گرداند. به طوری که این جواب بسیار نزدیک به مقدار بهترین جواب است. البته این روش فضای زیادی را اشغال می کند و نیز وزن هر یال باید در جایی ذخیره شود.

در نتیجه می بینیم که الگوریتم ژنتیک وقتی که از نمایش ماتریس برای encoding و از Matrix Crossover یا Heuristic Crossover استفاده می کند بهترین جواب رابرمی گرداند و بهتر از دیگر روشها کار می کند.در هر دو روش crossover بالا استفاده از mutation(2–opt)کیفیت الگوریتم را افزایش می دهد.

نتیجه گیری:

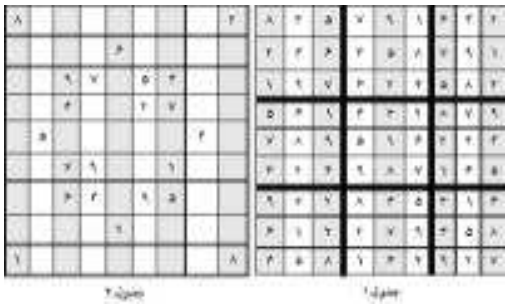
الگوریتمهای ژنیتیک به نظر می رسد که یک جواب خوب برای TSP پیدا می کند. در حالیکه کارایی الگوریتم به میزان زیادی به نحوه encode کردن و نیز crossover و mutation بستگی دارد. به نظر می رسد که استفاده از نمایش Matrix و Heuristic Information بهتر از بقیه روشها کار می کند و جواب قابل قبولی را برمی گرداند که به جواب واقعی بسیار نزدیک است.

احتمالا الگوریتم ژنتیک روش بهتری نسبت به دیگر روشها برای TSP است ولی هنوز جواب بهتری نسبت به دیگر روشهای موجود پیدا نکرده است. ولی این را نیز می دانیم که بهترین الگوریتمهای غیر ژنتیکی ارائه شده برای TSP در حالتهای خاصی از الگوریتمهای ژنتیک ارائه شده است. پس ما امیدواریم که با ارائه روتین های بهتری برای Encoding و Crossover و Mutation راه حلهای مناسب تری برای مسئله فروشنده دوره گرد ارائه شود.

الگوریتم ژنتیک و حل معمای سودوکو (Sodocu)

صورت مسئله

احتمالاً همه شما با جداول سودوکو (Sodocu) آشنایید، جداول بازی با اعدادی که اکنون در اکثر روزنامه‌ها و مجله‌ها در قسمت سرگرمی فضایی را به خود اختصاص داده‌اند یا شاید در اطرافتان افرادی باشند که به صورت جدی به حل مداوم این‌گونه جداول می‌پردازند. جدول سودوکو در حقیقت یک جدول 9*9 است که بعضی از خانه‌های آن با اعدادی بین 1 تا 9 پرشده است. شما می‌توانید در هر خانه خالی یکی از اعداد 1 تا 9 را قرار دهید اما باید این کار را طوری انجام دهید که سه شرط زیر برقرار باشد.



1-درهرسطر هیچ عدد تکراری وجود نداشته باشد.

2-درهرستون هیچ عدد تکراری نباید وجود داشته باشد.

3-در هر جدول 3*3 طبق شکل نیز نباید هیچ عدد تکراری وجود داشته باشد.

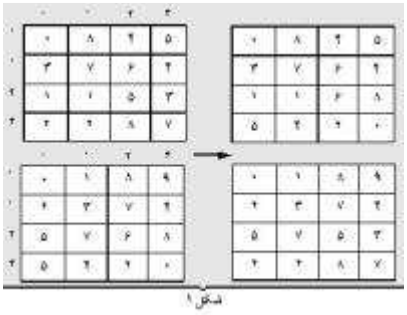
در جدول 1 یکی از جداول حل شده سودوکو را می‌بینید که شروط مورد نظر در آن صادق است، اما در حقیقت شکل آغازین مسئله مانند جدول 2 است که باید حل شود. اگر به حل یک نمونه جدول سودوکو تمایل دارید، می‌توانید جدول 2 را حل کنید.

مسئله مورد نظر در اینجا بدین‌صورت است که فرض کنید یک جدول سودوکوی خالی دارید و از شما می‌خواهند مثلاً پنجاه راه‌حل معرفی کنید. یعنی به پنجاه جدول که خاصیت سودوکو داشته باشد. توجه کنید که در مسئله ما همه خانه‌ها خالی هستند. برای این‌که زمان اجرای این برنامه کمتر شود و امکان رسیدن به جواب در خانه نیز مهیا باشد، در صورت مسئله مورد نظر تغییر کوچکی داده شده است؛ به این صورت که فرض می شود در هر خانه غیر از اعداد 1 تا 9 عدد صفر را نیز می‌توان قرار داد .

اگر بخواهیم بیان پیشرفته‌تری از صورت مسئله داشته باشیم، در حقیقت این مسئله، پیدا کردن حالت‌های مناسبی است در بین همه حالات ممکن برای جدول مزبور ما. در هر خانه جدول ده عدد مختلف می‌تواند قرار بگیرد. از طرفی دارای 81 خانه هستیم . پس مقدار حالات ممکن برای مسئله ما برابر 81^{10} است و از میان همه این

حالات ما می‌خواهیم به دنبال حالت‌هایی بگردیم که شرایط مورد نظر ما را داشته باشد. بدیهی است پیمایش همه این حالات، حتی برای کامپیوترهای امروزی که ما از آن‌ها استفاده می‌کنیم و در مدت زمان طبیعی، امری محال است.

حل مسئله



با توجه به این‌که تعداد حالات مسئله بسیار زیاد است، پیمایش کل فضای درخت حالت مسئله غیرممکن است. از طرفی ما نیازمند پنجاه جواب درست هستیم و مطمئناً جواب‌های مسئله ما در فضای درخت حالت مسئله پراکنده هستند و در نهایت این شرایط این ایده را به وجود می‌آورد که برای حل مسئله از الگوریتم‌های ژنتیک استفاده کنیم.

تعیین کروموزم

قدم اول در حل مسئله تعیین ساختار کروموزم است .همان‌طور که می‌دانیم هر کروموزم در الگوریتم ژنتیک، معادل یک وضعیت از حالات ممکن برای فضای حالت مسئله است .

در مسئله ما نیز جدول سودوکو را در قالب یک آرایه یک‌بعدی می‌توانیم تصور کنیم که اعداد متناظر با هر خانه به ترتیب در کنار هم قرار گرفته‌اند و در مراحل بعد با تعیین یک نقطه شکست در این آرایه، می‌توانیم عمل ترکیب را برای به دست آوردن حالات جدید انجام دهیم، اما در همین ابتدا می‌توانیم از اندکی ذکاوت برنامه‌نویسی خود استفاده کنیم و به جای این‌که جدول سودوکو را یک آرایه یک بعدی با 81 خانه در نظر بگیریم، همان آرایه دو بعدی 9*9 در نظر بگیریم و فقط شکست را ترکیبی از سطر و ستون فرض کنیم .

مثلاً اگر دو کروموزم به شکل فرضی زیر باشند، با این فرض که نقطه شکست در سطر بعد از ستون 1 باشد، حاصل ترکیب به فرم شکل 1 خواهد بود. (برای واضح بودن شکل فرض کردیم هر جدول 4*4 خانه است).

ساخت جمعیت آغازین یا نسل اول

همان‌طور که می‌دانیم، جمعیت آغازین را می‌توانیم به صورت تصادفی ایجاد کنیم. یعنی در هر خانه از یک نمونه از جدول مورد نظر یک عدد تصادفی بین صفر تا نُه قرار دهیم. در اینجا ذکر چند نکته لازم است: اول این‌که، در کد حل مسئله تعداد کروموزم‌های جمعیت آغازین را برابر پنجاه فرض شده است و از این به بعد از هر نسل پنجاه عنصر انتخاب می‌شوند و به عنوان عناصر سازنده نسل بعدی مورد استفاده قرار می‌گیرند .

از طرف دیگر، اگر بتوانیم جمعیت آغازین خود را به گونه‌ای انتخاب کنیم که در عین تصادفی بودن قسمت‌هایی از شروط‌ها را نیز در خود داشته باشد، مطمئناً بسیار سریع‌تر به جواب می‌رسیم؛ چراکه هر چه جمعیت آغازین بهتر باشد، احتمال دسترسی سریع به جواب بیشتر است.

با توجه به این ایده می‌توانیم جمعیت آغازین را به‌گونه‌ای طراحی کنیم که با این‌که در هر خانه یک عدد تصادفی قرار داشته باشد، در هر سطر هیچ عدد تکراری نداشته باشیم. (کد)

```
'''Generate first nation''
''''''''''''''''''''''''''
        For ZZ = 0 To 49
            For RR = 0 To 8
                For II = 0 To 8
                    FlagB = True
                    While FlagB = True
                        TempA = TrueRandom.Rand(10)
                        FlagA = True
                        For LL = 0 To 8
                            If M1(ZZ, RR, LL) = TempA
                               Then FlagA = False
                        Next LL
                        If FlagA = True Then
                            M1(ZZ, RR, II) = TempA
                            FlagB = False
                        End If
                    End While
                Next II
            Next RR
        Next ZZ
.
.
```

در اینجا ذکر یک نکته لازم است که با توجه به این‌که نیازمند تعداد بسیار زیادی عدد بین 0 تا 9 به صورت تصادفی هستیم و با توجه به این‌که این اعداد در کسر بسیار کوچکی از زمان برای کامپیوتر تولید می‌شوند، نمی‌توانیم از توابعی مانند RND که عدد تصادفی ایجاد می‌کنند، استفاده کنیم؛ چراکه هسته یا سید تولید عدد تصادفی در این توابع در بهترین حالت ساعت (زمان) اجرای تابع است که برای ما مناسب نیست .(کد)

```
Public Class TrueRandom
    Public Shared Function Rand(ByVal MaxNum As Integer) As Integer
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''generate  a random number between 0 and MaxNum-1 ''
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        Dim rnd(20) As Byte
        Dim num As Long
        Dim generator As New RNGCryptoServiceProvider
        generator.GetBytes(rnd)
        For x As Integer = 0 To 20
            num += CInt(rnd(x))
        Next x
        Return CInt(num Mod MaxNum)
    End Function
End Class
```

ساخت تابع از ارزش

در ادامه باید تابعی ایجاد کنیم که بتوانیم توسط آن به هر نمونه عددی متناسب با میزان خوب بودن آن را اختصاص دهیم. این تابع را که RANK می‌نامیم، بدین‌صورت تعریف می‌کنیم: به ازای هر سطر یا ستون یا هر مربع 3 *3 که شرط جدول سودوکو را داشته باشد، یک واحد به جواب تابع RANK اضافه می‌کنیم .

بدین‌صورت تابع RANK تابعی خواهد بود که دارای جواب بین صفر تا 27 باشد. البته بدیهی است که اگر نمونه‌ای دارای RANK برابر 27 باشد، بدین معنی است که معادل یکی از المان‌های جواب نهایی سودوکو مسئله ما است .(کد)

```
ublic Function Rank(ByRef M1(,,) As Integer, ByRef M1rank() As Integer, ByVal
BB As Integer)
'''this function is rank function and rank M1 that fill the result in M1rank'
   For ZZ = 0 To BB - 1
     Rank = 0
     '''''''vertical ranking for table''''''''''
     For RR = 0 To 8
        FlagA = True
        For II = 0 To 8
          For LL = 0 To 8
           If M1(ZZ,RR,II)= M1(ZZ,RR,LL) And II <> LL Then FlagA = False
          Next LL

        Next II
        If FlagA = True Then Rank = Rank + 1
     Next RR
```

باید توجه داشت که خروجی تابع RANK از نظر برنامه‌نویسی آرایه‌ای است یک‌بعدی به طول تعداد نمونه‌هایی که قرار است ارزشیابی شوند. مثلاً اگر بخواهیم جمعیت پنجاه‌گانه اولیه را ارزشیابی کنیم، خروجی تابع ما آرایه‌ای یک بعدی با طول پنجاه است که ارزش نمونه صفر در خانه شماره صفر و به همین ترتیب تا ارزش نمونه 49 در خانه شماره 49 قرار می‌گیرد(n= 50)

ترکیب نمونه‌ها و ساختن جواب جدید

در ادامه باید هر بار دو نمونه را انتخاب کنیم و از ترکیب هر دو نمونه دو جواب جدید به دست آوردیم باید توجه داشت که نباید یک نمونه با خودش ترکیب شود؛ چراکه دو جواب عیناً مثل خودش تولید می‌کند. در نهایت اگر بخواهیم به طور خلاصه بیان کنیم، با انتخاب دو عدد تصادفی بین صفر تا 49 دو نمونه را انتخاب می‌کنیم.

توجه داشته باشید که باید این عدد تصادفی به گونه‌ای باشد که نمونه‌ای که دارای مقدار تابع ارزش بیشتری است، به همان نسبت نیز دارای شانس انتخاب بیشتری باشد. (کد زیر)

```
Public Function selectinst(ByVal M1rank() As Integer, ByRef num1 As nteger)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''this functionm select random instance according to its rank chance''
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        Max = 0
        For II = 0 To 49
            Max = Max + M1rank(II)
        Next II
        TempNum = TrueRandom.Rand(Max) + 1
        num1 = -1
        CalcNum = 0
        II = -1
        While TempNum > CalcNum
            II = II + 1
            num1 = num1 + 1
            CalcNum = CalcNum + M1rank(II)
        End While
        Return (num1)
    End Function
.
.
.
```

پس از این انتخاب دو نمونه با انتخاب دو عدد تصادفی بین صفر تا هشت شماره سطر و ستون نقطه شکست را به دست می‌آوریم و عمل ترکیب را انجام می‌دهیم و دو جواب جدید به دست می‌آوریم، اما صرف عمل ترکیب ما را به جواب نهایی رهنمون نخواهد کرد، بلکه باید از عمل جهش نیز استفاده کنیم .

بنابراین به ازای هر جواب به دست آمده است این اعمال را انجام دهیم. ابتدا یک عدد تصادفی بین یک یا صفر انتخاب می‌کنیم. اگر عدد ما صفر بود، عمل جهش را انجام نمی‌دهیم، اما اگر یک بود، دو عدد تصادفی دیگر بین صفر تا هشت به نشانه شماره سطر و ستون خانه‌ای که باید در آن جهش انجام شود و یک عدد تصادفی بین صفر تا نُه به عنوان مقدار جدید آن خانه به دست می‌آوریم و عمل جهش را در آن انجام می‌دهیم. بدین‌ترتیب هر جواب ممکن است با احتمال پنجاه درصد در یکی از ژن‌های خود دچار جهش بشود.

مطالب ذکر شده به صورت زیر کد می شود .تابعی که آرایه مجموعه آغازین (مجموعه پنجاه‌تایی) و مجموعه جواب بعدی (مجموعه ده هزارتایی) و دو عدد تصادفی نقطه شکست و آدرس ذخیره کردن جواب‌ها در مجموعه بعدی را دریافت می‌کند دو عمل ترکیب و جهش را انجام می‌دهد و در نهایت آرایه‌ای ده‌هزارتایی به عنوان جواب برمی‌گرداند. (کد)

```
Public Function produce(ByRef M1(,,) As Integer, ByRef M2(,,) As Integer,
ByVal num1 As Integer, ByVal num2 As Integer, ByVal GenerationArrayPointer2
As Integer)
        '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''this function compund two result and make new answer'''
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   TempRow = TrueRandom.Rand(9)
   TempCol = TrueRandom.Rand(9)

   For II = 0 To TempRow - 1
       For RR = 0 To 8
           M2(GenerationArrayPointer2, II, RR) = M1(num1, II, RR)
           M2(GenerationArrayPointer2 + 1, II, RR) = M1(num2, II, RR)
            Next RR
   Next II
   For RR = 0 To TempCol
       M2(GenerationArrayPointer2,TempRow,RR)= M1(num1,TempRow,RR)
       M2(GenerationArrayPointer2+1,TempRow,RR)= M1(num2,TempRow,RR)
   Next RR
   For RR=TempCol To 8
       M2(GenerationArrayPointer2,TempRow,RR)= M1(num2,TempRow,RR)
       M2(GenerationArrayPointer2 + 1, TempRow, RR) = M1(num1, TempRow, RR)
   Next RR
.
.
.
```

ارزشیابی مجموعه جواب

اکنون دارای یک آرایه با ده هزار عنصر به عنوان جواب هستیم که مطمئناً بعضی از این جواب‌ها نسبت به بقیه بهتر هستند و همان‌طور که می‌دانیم، برای ساختن نسل بعد، باید از جواب‌هایی استفاده کنیم که از نظر ارزش، دارای رتبه بهتری باشند. بدین‌ترتیب باید در ادامه جواب‌های خود را ارزشیابی کنیم .این عمل دقیقاً همان تابع RANKاست، اما این‌بار مجموعه ده هزارتایی را ارزشیابی می‌نماید و خروجی را در یک آرایه یک بعدی ده‌هزارتایی می‌ریزد.

ساختن نسل بعد

در اینجا باید از میان ده هزار جواب به دست آمده، جواب‌های برتر را به عنوان والدهای نسل بعدی انتخاب کنیم و بقیه را دور بریزیم، که در آن از تکنیک نخبه‌گرایی استفاده شده است .

در این مرحلهبه این صورت عمل می کنیم که اگر عناصر آرایه ارزش مجموعه والد همه دارای یک مقدار بودند، عمل نخبه‌گرایی را انجام نمی‌دهیم، در غیراین‌صورت، از میان والد نسل قبلی ده والدی را انتخاب می‌کنیم که دارای مقدار تابع ارزش بیشتری بودند و در مجموعه والد نسل بعد قرار می‌دهیم. (کد)

```
  GenerationArrayPointer = 0
FlagE = False
XX = M1rank(0)
For RR = 0 To 49
    If XX <> M1rank(RR) Then FlagE = True
Next RR
If FlagE = True Then
    While GenerationArrayPointer < 10
        Big = -1
        Loc = -1
        For RR = 0 To 49
            If Big <= M1rank(RR) Then
                Big = M1rank(RR)
                Loc = RR
            End If
        Next RR
       M1rank(Loc) = -1
.
.
.
.
```

با توجه به این‌که مجموعه والدها برابر پنجاه عضو است، اگر عمل نخبه‌گرایی انجام شود، باید چهل عضو باقیمانده را از میان مجموعه جواب (مجموعه ده‌هزارتایی) انتخاب کنیم، اما اگر نخبه‌گرایی انجام نشود، هر پنجاه عضو از میان مجموعه جواب انتخاب می‌شوند. معیار انتخاب نیز که البته واضح است: عضوهایی انتخاب می‌شوند که بیشترین امتیاز را از تابع ارزش دریافت کرده باشند. (کد)

```
While GenerationArrayPointer < WW
    Big = -1
    Loc = -1
    FlagC = True
    For RR = 0 To qqq - 1
        If Big < M2rank(RR) Then
            Big = M2rank(RR)
            Loc = RR
        End If
    Next RR
    M2rank(Loc) = -1
    FlagC = False

    For ttt = 0 To GenerationArrayPointer - 1
        comp(M1, ttt, M2, Loc, FlagD)
.
.
.
```

بقیه مراحل مشخص است که تکرار مراحل قبل می باشد، یعنی ارزشیابی مجموعه والد، انتخاب دو والد تصادفی متناسب با تابع ارزششان و ساختن جواب جدید و تکرار این مرحله تا به دست آوردن نسل بعدی و در نهایت انتخاب والدهای بعدی از میان نسلی به دست آمده و تکرار دوباره.....

بهینه سازی چینش حروف فارسی بر روی صفحه کلید با استفاده از الگوریتم های ژنتیکی

چکیده :

به دست آوردن چینش بهینه ی حروف فارسی بر روی صفحه کلید در دراز مدت برای کسانی که با تایپ کردن متون فارسی درگیر هستند، بسیار مفید خواهد بود. یک الگوریتم تکاملی می تواند با توجه به یک تابع تناسب که میزان راحتی تایپ کردن متون فارسی را برای یک چینش ارائه می دهد، در فضای چینش های حروف فارسی بر روی صفحه کلید جستجو کرده و چینش بهینه را به دست اورد. در اینجا، یک الگوریتم ژنتیک به دنبال یافتن بهترین جایگشت از 32 حروف فارسی بعلاوه حروف حمزه "ء" بر روی 33 کلید اصلی صفحه کلید است. تابع تناسب برای هر جایگشت یا چینش، با توجه به عوامل راحتی در تایپ کردن، هزینه ای را به ان چینش اختصاص می دهد. این عوامل به مواردی چون استفاده متناوب از دست ها در تایپ متون، استفاده نکردن از یک دست برای تایپ دو حروف متوالی، و میزان سختی تایپ هر دو در چینش مربوطه بر روی صفحه کلید بر می گردد. در بررسی هایی که با استفاده از متون چندین سایت فارسی زبان انجام شد، مشخص شد که چینش بهینه حاصل از این الگوریتم ژنتیک، بهتر از چینش فعلی عمل می کند.

مقدمه

از ان زمانی که کامپیوتر به وجود آمد تا کنون، صفحه کلید به عنوان اصلی ترین رابط میان انسان و کامپیوتر معرفی شده است. قبل از ان نیز از صفحه کلید در ماشین های تایپ استفاده می شد. از همان ابتدائی که صفحه کلید مستطیل شکل توسط کریستوفر لاتام شلز طراحی شد و در ما شین های تایپ مورد استفاده قرار گرفت، تا کنون که بیش از130 سال از ان می گذرد، این طراحی به صورت پیوسته مورد انتقاد منتقدین قرار داشته است. علاوه بر اینکه طراحی فیزیکی این صفحه کلید مورد انتقاد قرار گرفته است، از نحوه ی چینش حروف بر روی این صفحه کلید مستطیل شکل نیز انتقاد شده است. محققان برای ایجاد چینش بهینه ی حروف بر روی این صفحه کلید الگوریتم های مختلفی ارائه کرده اند. اما این الگوریتم ها اکثرا برای زبان انگلیسی ارائه شده اند و متاسفانه در مورد زبان فارسی کار جدیدی صورت نگرفته است و همان چینشی که در ابتدا استفادهی از زبان فارسی در کامپیوتر ارئه شد(چینش فعلی) ، مورد استفاده قرار می گیرد.

محققان مختلفی از الگوریتم های پردازش تکاملی و روش های مشابه برای حل مساله چینش حروف بر روی صفحه کلید استفاده کرده اند. اولین محققی که در این زمینه شروع به کار کرده گلاور است. وی صفحه کلید را مجموعه ای از اجزای کوچکتر در نظر گرفته و یک الگوریتم ژنتیک طراحی کرد که جمعیت(Population) آن از کد کردن جایگشت های مختلف این اجزاء بر روی صفحه کلید ایجاد می شد. برای بدست آوردن جایگشت بهینه نیز یک سری عملگر ژنتیکی مناسب با جمعیت طراحی کرد.

افراد بعدی که روشی برای حل این مساله ارائه دادند، لایت و آندر سون بودند که از انیلینگ شبیه ساری شده (Simulated Annealing) برای جستجوی فضای چینش های مختلف حروف بر روی سه ردیف صفحه کلید که حروف انگلیسی بر روی ان ها قرار دارد (ده،نه، وهفت کلید در ردیف های بالا تا پایین) ، استفاده کردند. تابع تناسب (Fitness Function) ارائه شده توسط این افراد از زمان تایپ شدن و فرکانس تکرار حروف استفاده می کرد.

کلاوسر یک استراتژی تکاملی برای حل این مساله طراحی کرد که از یک عملگر استفاده می کرد و تنها جای دو حروف را عوض می کرد. وی این الگوریتم را برای حروف انگلیسی بعلاوه چهار علامت نشانگذاری در سه ردیف ده کلیدی به کار برد. تابع تناسب استفاده شده میزان جا به جایی انگشتان را از موقعیت پایه ای آنان (شکل 1) محاسبه کرده و الگوریتم به سمت کمینه کردن مجموع جا بجا کردن های انگشتان دست برای یک متن ثابت به ازای چینش های مختلف حرکت می کند.

شکل 1-موقعیت پایه ای انگشتان تایپیست بر روی صفحه کلید

واگنر و همکارانش از بهینه سازی کلونی مورچه ها (Ant Colony Optimization) برای جست و جو در فضای چینش های حروف بر روی صفحه کلید استفاده کردند. تابع تناسب انها عواملی نظیر فرکانس فشردن

کلید های مختلف, استفاده از دست های مختلف برای تایپ دو حروف متوالی, و استفاده از دو انگشت برای تایپ حروف متوالی را در نظر می گیرد.

دشوال و دب از یک الگوریتم ژنتیک حالت دائمی (Steady State) برای حل مساله چینش حروف هندی بر روی یک صفحه کلید چهار ردیفی استفاده کردند و از عملگر های جهش (Mutation) و باز ترکیبی (Crossover) متناسب با جمعیت موجود در این الگوریتم که جایگشت های حروف بر روی کلید های این صفحه است, استفاده کردند.

گوتی ,بورگ,و جالسترم نیز از یک الگوریتم ژنتیک برای حل این مساله استفاده کردند. انها نیز از فاکتور های در نظر گرفته شده توسط واگنر و همکارانش در تابع تناسب خود استفاده کردند. انها برای اصلاح نسل جامع الگوریتم ژنتیک خود از توابع باز ترکیبی و جهش استفاده کردند. تابع جهش تنها جای دو حروف را در یک کروموزوم با هم عوض می کند و تابع باز ترکیبی دو کروموزوم والد را گرفته و آن حروفی را جایگاهشان یکی است, تغییر نداده و مابقی را به صورت تصادفی تغییر داده و دو کروموزوم جدید فرزند ایجاد می کند.

در بخش های بعدی به ترتیب , مسئله ای را که الگوریتم ژنتیکی بکار رفته به آن ارجاع می دهد, مورد بحث قرار خواهد گرفت, توابع و پارامتر های الگوریتم ژنتیک مورد استفاده بررسی می شوند, و آزمایش هایی که توسط این الگوریتم انجام شده است, بیان شده و نتایج آن مورد بررسی قرار خواهد گرفت.

مساله

همانند دیگر الگوریتم هایی که فضای چینش های مختلف را جست و جو می کنند, در این مساله نیز هندسه ی صفحه کلید ثابت است و ما می خواهیم که تعداد33 نشانه که متشکل از 32 حروف الفبای فارسی بعلاوه ی حروف حمزه"ء" است را بر روی سه ردیف صفحه کلید که به ترتیب دارای 11,12 و10 کلید هستند,قرار دهیم. هدف این مساله بدست آوردن چینشی از این نشانه ها بر روی این کلید ها است, به طوری که این چینش طوری باشد که کاربر هنگام استفاده از کلید برای تایپ حروف فارسی, احساس راحتی بیشتری نسبت به کار با بقیه چینش ها داشته باشد.

الگوریتم ژنتیک

برای حل مساله بیان شده در قسمت قبلی, از یک الگوریتم ژنتیک استفاده شده است. تابع تناسب موجود در این الگوریتم ژنتیک, میزان راحتی یا سختی استفاده از یک چینش را محاسبه می کند. در هر نسل, عملگر های ژنتیکی بر روی جمعیت موجود که چینش های مختلفی از حروف فارسی بر روی صفحه کلید هستند, اعمال می شوند و جامعه به سمتی سوق داده می شود که مقدار تابع تناسب بازای اعضای آن به کمینه مقدار خود برسند. میزان تناسب هر عضو از جامعه که در واقع یک چینش ارسی بر روی صفحه کلید هستند, با اعمال تابع تناسب بر متنی که از مطالب چند سایت جبری فارسی زبانان تهیه شده است,به دست می آید.

جمعیت

اعضای تشکیل دهنده ی جمعیت در این مساله جایگشت های مختلف حروف فارسی روی صفحه کلید یا به عبارت دقیق تر, چینش ها هستند.

هر عضو جمعیت در این مساله را می توان به صورت برداری از حروف فارسی در نظر گرفت که هر اندیس آن متناظر با یک کلید از صفحه کلید است. مثلا هر بردار با طول 33 که شامل حروف فارسی بعلاوه ی حروف حمزه "ء" باشد را میتوان به عنوان یک کروموزوم (یک عضو از جمعیت) در نظر گرفت که حرف iام از این بردار, متناظر با کلیدی از صفحه کلید است که بر چسب شماره ی i بر روی آن زده شده است. در شکل 2- الف اندیس های متناظر هر کروموزوم بر روی صفحه کلید نشان داده شده است و در شکل 2- ب یک کروموزوم از جمعیت را که متناظر با چینش فعلی صفحه کلید است, نشان داده شده است.

شکل 2- اندیس های ژن های هر کروموزوم بر روی صفحه کلید

۱	۲	۳	۴	۵	۶	_	۳۳
ض	ص	ث	ق	ف	غ	_	پ

شکل 3- ساختار یک کروموزوم از جمعیت که متناظر با چینش کنونی حروف فارسی بر روی صفحه کلید است.

در حالت کلی, ما به دنبال یک عضو از فضای تمامی اعضای ممکن جمعیت هستیم که با توجه به تابع تناسبی که در بخش بعدی تعریف می شود, هزینه کمتری نسبت به مابقی داشته باشد. نکته قابل توجه این است که تعداد چینش های مختلف 33! یا 8/8×1036 عدد است و این فضایی است که الگوریتم ژنتیک باید آن را به دنبال چینش بهینه جستجو کند.

تابع تناسب

تعیین راحتی و سختیه کار کردن با چینش حروف بر روی صفحه کلید یک مساله پیچیده ارگومونیک است. لذا می توان به کاری که محققان متخصص در این زمینه انجام داده اند, استناد کرد. نورمن و رومل هارت چهار هدف را برای طراحی کارای یک صفحه کلید ارائه کرده اند: برابری کاری که دو دست انجام می دهند, بیشترین تایپ حروف به صورت متناوب با دو دست, کمترین تکرار تایپ دو حروف متوالی با یک انگشت, و بیشترین تایپ حروف بر روی کلید های پایه ای(کلید های ردیف وسط). بنابراین,تناسب یک کروموزوم با شبیه سازی تایپ کردن یک متن که این کروموزوم معرف چینش صفحه کلید آن است, ارزیابی می شود که عوامل اندازه گیری را می توان به صورت هزینه تعریف کرد. برای دو هدف اول می توان فاکتور اندازه گیری زیر را معرفی کرد:

Chand: هزینه ی مربوط به استفاده از یک دست برای تایپ کردن دو حروف پشت سر هم.

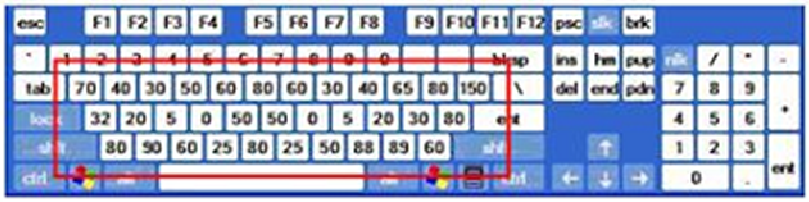
این فاکتور هر دو هدف اول را می پوشاند. هدف دوم که مستقیما براورده می شود. اما در مورد هدف اول, چون استفاده متناوب از دو دست برای تایپ دو حرف متوالی در نهایت بار کلی تایپ یک متن را به صورت برابر بین دو دست تقسیم می کند, لذا هدف اول نیز پوشانده می شود. برای هدف سوم, فاکتور اندازه گیری زیر معرفی می شود:

Cfinger : هزینه ی مربوط به استفاده از یک انگشت برای تایپ کردن دو حروف پشت سر هم.

اما فاکتور اندازه گیری دیگری در نظر گرفته شده است که نه تنها هدف چهارم را پوشش می دهد, بلکه عوامل دیگری نیز دز نظر دارد. در این فاکتور تنها به فرکانس فشردن کلید های پایه ای توجه نمی شود, بلکه استفاده از انگشتان مختلف دست و میزان راحتی استفاده از آنها, و میزان جا به جا شدن انگشتان بر روی صفحه کلید در

نظر گرفته می شود. شکل 3 هزینه ی مربوط به فشرد شدن هر کلید از صفحه کلید را نشان می دهد که این اعداد از محاسبه با تایپیست ها و پیانیست های حرفه ای بدست آمده است. البته لازم به ذکر است که این اعداد برای یک فرد راست دست بیان شده اند. بنابر انچه که بیان شد, فاکتور سوم به صورت زیر بیان می شود:

Cergonomic: هزینه ی مربوط به تایپ کردن یک حرف با توجه به موقعیت آن حرف بر روی صفحه کلید.



شکل 3- هزینه مربوط به فشردن هر کلید از صفحه کلید

تابع تناسب برای هر کروموزوم از مجموع این سه فاکتور برای تمامی حروفی که در متن مورد استفاده برای آزمایش وجود دارند,بدست می آید:

$$\sum_{w_i \in W} \sum_{l_j \in w_i} [C_{hand}(l_j, l_{j-1}) + C_{finger}(l_j, l_{j-1}) + C_{ergonomic}(l_j)]$$

که در آن, W مجموعه ی تمامی کلمات موجود در متن مورد استفاده برای آزمایش است, w_i کلمه ی i ام از مجموعه ی W است, حرف l_j ام از کلمه ی w_i است, Cergonomic تابعی است که هزینه ی مربوط به تایپ حرف l_j ا با توجه به مقادیر مشخص شده در شکل 3 و چینش مربوطه بر می گرداند, Cfinger تابعی است که اگر دو حرف l_j و l_{j-1} با توجه به چینش مربوطه توسط یک انگشت تایپ می شوند, یک مقدار ثابت (متوسط اعداد نشان داده شده در شکل 3) را بر می گرداند و در غیر این صورت مقدار خروجی آن صفر است, Chand تابعی است که اگر دو حرف l_j و l_{j-1} با توجه به چینش مربوطه تو سط یک دست تایپ شوند, یک مقدار ثابت (یک چهارم مقدار ثابت ذکر شده برای تابع Cfinger) را بر می گرداند و در غیر این صورت مقدار خروجی آن صفر است. در دو تابع Cfinger و Chand اگر حرف l_j اولین حرف کلمه w_i باشد. میزان خروجی برای این دو تابع صفر است.

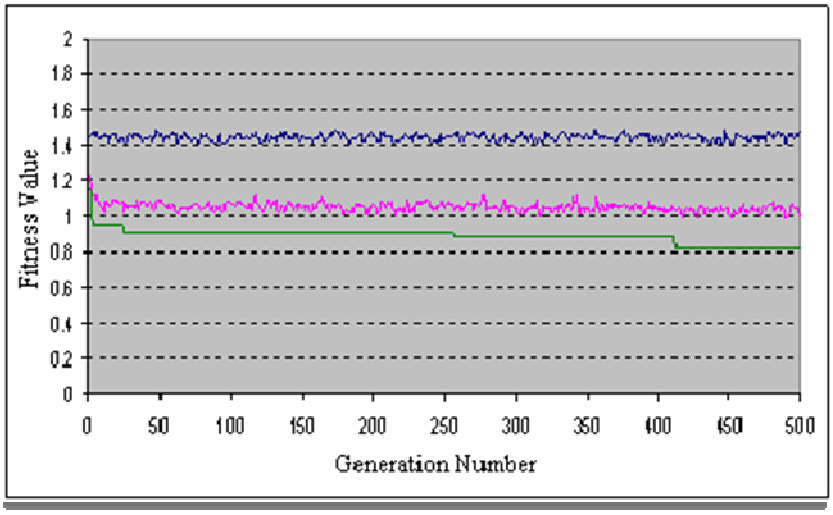
عملگر های ژنتیکی

برای اینکه جمعیت موجود در یک الگوریتم ژنتیک را به سمتی سوق داد که تابع تناسب مینیمم شود, باید از عملگر های ژنتیکی استفاده کرد. در اینجا تنها از عملگر جهش استفاده شده است. دلیل عدم استفاده از عملگر باز ترکیبی این است که ساختار اعضای جمعیت طوری است که ترکیب کردن دو کروموزوم والد (به طوری که کروموزوم های فرزند ژن های مشترک والدینشان را داشته باشند و ما بقی را به صورت تصادفی عوض کنند) هزینه ی زمانی بالائی دارد.

در صورتی که می توان عملگر جهش را طوری طراحی کرد که هم هزینه ی زما نی بسیار کم تری داشته باشد و هم جای خالی عملگر باز ترکیبی را پر کند. ما عملگر جهش را به دو صورت برای اعضای مختلف جامعه به کار می بریم. در این مساله ما یک جامعه ی نخبگان انتخاب می کنیم که اعضای آن٪ α تراز اول جمعیت را از دید تابع تناسب تشکیل می دهند. عملگر جهش برای هر عضو از جامعه ی نخبگان, تنها محتویات چهار زوج ژن را به صورت تصادفی جا بجا می کند. در حالیکه برای افراد عادی جامعه این تعداد به 12 جا بجایی افزایش می یابند. این تبعیض بین نخبگان جامعه و اعضای عادی, باعث می شود که نسل نخبگان کمتر دستخوش تغییر گردد و تغییرات زیادی بر روی اعضای عادی جامعه ایجاد شود. لذا در گذار نسل ها امکان ایجاد افراد نخبه از افراد عادی با توجه به تعداد تغییرات, بالا است. همچنین افراد نخبه ی نسل های قبلی نیز دارای تغییرات کمی هستند و لذا در انها نیز امکان اصلاح نسل وجود دارد. لازم به ذکر است که تعداد اندکی از اعضای جامعه که بهترین تناسب ها را دارند, بدون تغییر به نسل بعدی راه پیدا می کنند.

کارایی

الگوریتم ژنتیک توضیع داده شده در بخش قبل را پارامتر های زیر اجرا کردیم : تعداد اعضای جمعیت 100 کروموزوم که در نسل اول به صورت تصادفی تولید شده اند, α که درصد تشکیل دهنده جامعه نخبگان است, 10٪ کل جمعیت , تعداد اعضایی که به صورت مستقیم و بدون اینکه عملگر های ژنتیکی بر روی ان اعمال شود, به نسل بعدی می روند, 3 عضو و تعداد کل نسل ها 500 نسل. نمودار اصلاح نسل بر اساس میزان تناسب که با استفاده از تابع تناسب بدست می آید, در شکل چهار نشان داده شده است. در این نمودار, میزان تناسب اعضا نسبت به تناسبی که چینش کنونی حروف فارسی بر روی صفحه کلید دارد نرمال شده اند. منحنی های نشان داده شده در این شکل, به ترتیب از بالا به پایین, متوسط مقادیر تناسب همه ی اعضای جامعه, متوسط مقادیر تناسب جامعه ی نخبگان, و بهترین تناسب در طی 500 نسل هستند.



شکل 4- نمودار های تناسب اعضای جامعه در طی نسل های مختلف.منحنی نشان داده شده به ترتیب از بالا به پایین, متوسط مقادیر تناسب همه ی اعضای جامعه, متوسط مقادیر تناسب همه ی اعضای جامعه, متوسط مقادیر جامعه ی نخبگان,و بهترین تناسب هستند.

بهترین چینشی که در نهایت این الگوریتم ژنتیک برای حروف فارسی ارائه داد, هزینه اش با توجه به تابع تناسب,815 /0هزینه ی چینش کنونی حروف فارسی است که رقم قابل ملا حظه ای است. این چینش در شکل 5 نشان داده شده است.

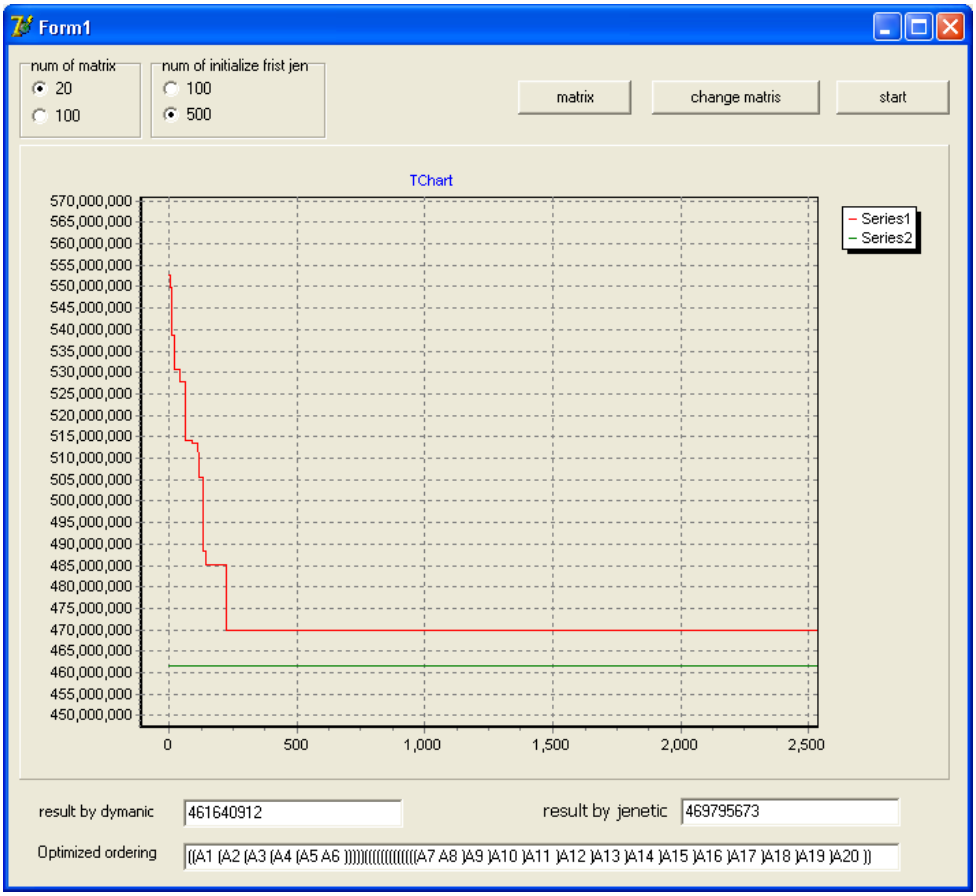
شکل 5- چینش ارائه شده توسط الگوریتم ژنتیک برای حروف فارسی بر روی صفحه کلید

برای صفحه کلید های مستطیل شکل کنونی, یک چینش مناسب حروف باید از جابجایی های زیاد انگشتان دست بر روی صفحه کلید جلو گیری کند. همچنین باید فاکتور های ارگونومیک دیگری را نیز برای راحتی کاربر در نظر داشته باشد.از جمله ی این فاکتور ها می توان به تایپ نکردن دو حروف متوالی با یک انگشت و حتی با یک دست و تقسیم بار تایپ حروف به صورت مساوی بین دو دست اشاره کرد.

الگوریتم ژنتیک ما در این مساله از بین فضای بسیار بزرگ چینش های مختلف حروف فارسی که از جایگشت 32 حرف الفبای فارسی به علاوه حرف "ء" بر روی 33 کلید صفحه کلید تشکیل شده است, به دنبال چینش بهینه می گردد. بعد از اجرای این الگوریتم, با توجه به فاکتور های ذکر شده که در تابع تناسب گنجانده شده است, چینش بهینه ای ارائه شد که هزینه ی آن بر اساس تابع تناسب به مراتب کمتر از هزینه ی چینشی است که هم اکنون بر روی صفحه کلید وجود دارد.

حل مسئله ضرب ماتریس ها به وسیله الگوریتم ژنتیک

توضیح حل این مسئله در این رساله آورده نمی شود .در اینجا نمونه ای از یک مرتبه اجرای برنامه به ازای 20 ماتریس و تعداد جمعیت اولیه 500 را با شرط 10000 تکرار مشاهده می کنید که هزینه الگوریتم ژنتیک به هزینه کمینه آن که از مرتبه $O(n^3)$ است نزدیک میشود.



Glossary

Adaptation

اقتباس، تطبیق،

در زیست‌شناسی به تغییرات ژنتیکی برای رسیدن به یک منظور خاص گفته می‌شود، به نحوی که مستلزم اضافه شدن رشته‌هایی به ژنهای اولیه بشود. در صورت عدم اضافه شدن چیزی، به‌جای فرآیند تطبیق، فرآیند کپی صورت گرفته است.

مثال: تطبیق چشم در تاریکی و روشنایی برای تنظیم میزان نور وارده به آن

Allel

حالتهای مختلف یک ژن برای قرار گرفتن در یک Locus

هر یک از مقادیر متفاوتی که یک ژن از یک کروموزوم در یک موقعیت خاص می‌تواند دارا باشد.

Answer

جواب،راه‌حل بهینه،یک یا چند solution مساله که دارای بالاترین برازش باشند.

Choromosome

کروموزوم ، هریک از راه حلهای کاندیدا به عنوان جواب بهینه، مجموعه‌ای از ژنها با فیلد های مختلف که به عنوان افراد در استخر ژنی اولیه حضور دارند.

Crossover

برش،تقاطع،همبُری، یکی از عملگرهای مهم الگوریتمهای ژنتیک

در الگوریتمهای ژنتیک به معنی Sexual Recombination نیز می‌باشد.

Elitism

نخبه گرایی،بهینه‌گزینی

چون ممکن است بدلیل ماهیت تصادفی الگوریتمهای ژنتیک، بهترین افراد یک نسل از بین رفته و در نسل بعد حضور نداشته‌باشند، برای جلوگیری از این مساله بهترین افراد را بدون هیچ گونه تغییری در نسل بعدی کپی می‌نمائیم تا کارآیی را افزایش دهیم.

Epistasis

خصوصیات و ویژگیهای یک ژن که در یک نژاد دیگر مخفی می‌شود.

Evolution

فرآیند تغییرات در طول زمان که با تغییر در آرایش ژنتیکی نوع حاصی همراه باشد.

میزان نرخ تغییرات allel در یک نسل نسبت به نسلهای دیگر

Feature

خصیصه، ویژگی خاصی که قابل کپی برداری یا انتقال به دیگر ژنها یا کروموزوم ها باشد.

Fitness_

شایستگی، برازش، قابلیت یک کروموزوم خاص از یک ژنوتیپ برای تولید مثل

Fitness_Function

تابع برازش، تعیین میزان شایستگی یک راه حل

بررسی مقدار بهینگی یک راه حل، objective function

Gene

بلوکی از DNA که دارای ساختار پروتئینی خاصی است و شکل کلی ارگانیسم را می‌سازد.

هر یک از بیتهای یک رشته کروموزومی

مثلاً ژن رنگ چشم یک جانورداری این مشخصات است:

دارای شماره 10 است (Locus آن در Position شماره 10 است.)

Allel Value آن آبی است. (یا مشکی، سبز، ...)

Genome

راه حلهای مختلف یک مساله شامل data ودستورالعمل که معمولاً به شکل رشته بیان می‌شود. دنباله کامل DNA از یک مجموعه از کروموزوم ها، یک مجموعه از پارامترها که راه حل مساله را تعریف می‌نماید.

Genotype

فضای جوابهای کد شده شامل تمامی رشته‌ها

نوع و معرف نماینده یک جنس(از موجودات دارای صفات مشابه ارثی)

با مطالعه DNA ژنوتیپ به طور کامل مشخص می‌شود.

ژنوتیپ=فنوتیپ کد شده

Heredity

انتقال کاراکترها و صفات از والدین به فرزندان با کپی ژنهای مختلف حین آمیزش

Implicit_Parallelism

توازی مجازی، مسائلی با ماهیتهای متفاوت در شاخه‌های مختلف علوم که بعد از کد شدن بسیار به یکدیگر شبیه شده و ممکن است دارای جوابهای یکسانی نیز باشند.

Individual

شخص، فرد، منحصر به فرد،

Locus

موقعیت مکانی یک ژن روی یک کروموزوم که می‌تواند بوسیله هرکدام از آللهای موجود اشغال شود.مثلاً در رشته 11101 عدد 0 در موقعیت مکانی چهارم از سمت چپ قرار دارد.

Mutation

جهش، یکی از عملگرهای الگوریتمهای ژنتیک

با استفاده از این عملگر تضمین می‌شود که هیچ جوابی از دید الگوریتم مخفی نمی‌ماند.

Natural_Selection

مکانیزم اصلی تکامل و انتخاب در طبیعت که بوسیله چارلز داروین توضیح داده شده است.

Offspring

زاد و ولد، فرزند، اولاد، مبدا، منشا

Phenotype

فضای اولیه جوابها با همه محیط آن که نشاندهنده رفتار فیزیکی واقعی است.

با مطالعه رفتار یک ارگانیسم ،فنوتیپ بطور کامل مشخص می‌شود . فنوتیپ=ژنوتیپ دیکد شده

Premature Convergence

همگرایی زودرس، همگرایی سریع به جواب بدون تضمین بهینگی بدین معنی که به احتمال قوی به یک نقطه بهینه محلی رسیده‌ایم.

فلسفه استفاده از عملگر جهش برای اجتناب از این حالت است.

Recombination

زیست شناسی:پدیده لقاح میان تخمک و اسپرم

شیمی: ترکیب گازهای هیدروژن و اکسیژن در یک باتری در طول عمل شارژ شدن آن که باعث تولید آب می‌شود.

Selection

زیست شناسی:آلل‌های خاصی از یک نوع که برای اعمال بعدی مانند: آمیزش، جهش و برش انتخاب می‌شوند.معمولترین نوع‌های ان در طبیعت شامل حالتهای زیر است:

- Sexual
- Ecological
- Stabilizing
- Disruptive
- Directional

Reproduction

تولید یک چیز از خودش به نحویکه به چیز اولیه اضافه یا از آن کم نشده باشد ، کپی برداری،

این عمل به دو شیوه جنسی (اشتراک مواد جنسی از دو والد مختلف) و غیر جنسی (تولید کپی از والد)صورت می‌پذیرد.

Simulated Annealing(SA)

یک روش اکتشافی بر اساس احتمالات درمسایل بهینه سازی در فضای جستجوی بسیار بزرگ توسط Kirkpatrick، Glatt و Vecchi در سال 1983 مطرح شد.ایده اصلی آن از فرآیند Annealing در مهندسی مواد (متالوژی)حاصل شده است.که به معنی گداختن و سردکردن کنترل شده مواد، برای افزایش اندازه کریستالهای تشکیل دهندهوکاهش شکنندگی آنان است.

در استفاده از SA ، هر نقطه دلخواه از فضای جستجو با یک حالت فیزیکی سیستم، معادل گرفته می‌شود.

می‌خواهیم تابع E(s) که انرژی درونی سیستم را بیان می‌کند مینیمم بشود. هدف ما شروع از یک وضعیت کاملاً دلخواه اولیه و رسیدن به حالت حداقل انرژی درونی است.

Solution

راه‌حل،یک جواب محتمل مساله

بهترین راه حل که دارای بالاترین برازش می‌باشد، جواب مساله نامیده می‌شود.

Species

نوع، گونه، نوعی از ارگانیسم، انواع

این کلمه هم به معنای مفرد بکار می‌رود و هم نشانه جمع است. در شیمی: مولکول، یون، اتم، رادیکال آزاد

Stochastic_process

فرآیند تصادفی،اتفاقی،غیر قطعی،random ،گرفته شده از واژه یونانی < stochos >،فرآیندی که حالت بعدی آن، بطور کامل از روی محیط و حالت قبلی، قابل تعیین نیست.

به عنوان مثال می‌توان از سریهای زمانی، سیگنالهای صوتی و تصویری،موارد بسیاری در علم پزشکی مانند نوار قلب، نوار مغز، فشار خون، حرارت بدن و ... نام برد.

Trajectory

خط سیر، مسیر گلوله، گذرگاه، مسیر مناسب و با بیشترین قابلیت احتمال عبور

منابع و مراجع

1. عباس راستگو, رضا شفیعی زاده- طراحی و ترکیب بندی مکانیزم ها به کمک الگوریتم ژنتیک.
2. محمد جوادی – الگوریتم ژنتیک – انتشارات دانشگاه امام حسین (ع)
3. سایت برنامه نویس www.barnamenevis.com
4. مجله علم و کامپیوتر www.ccwmagazine.com
5. www.wikipedia.com
6. www.talkorigins.org
7. www.gpwiki.org
8. پاورپوینت Koza
9. www.smi.stanford.edu/people/koza
10. دانشکده کامپیوتر دانشگاه McGill کانادا
11. www.cgm.cs.mcgill.ca
12. www.sharifthinktank.com
13. www.itna.com
14. Guided operators for a hipper-heuristic Genetic Algorithm
    www.cs.nott.ac.ukIT university of Nottingham